# Surface Reconstruction with Triangular B-splines

Ying He and Hong Qin
*Department of Computer Science*
*State University of New York at Stony Brook*
*Stony Brook, NY, 11794-4400, USA*
{*yhe*|*qin*}*@cs.sunysb.edu*

## Abstract

*This paper presents a novel modeling technique for reconstructing a triangular $B$-spline surface from a set of scanned 3D points. Unlike existing surface reconstruction methods based on tensor-product $B$-splines which primarily generate a network of patches and then enforce certain continuity (usually, $G^1$ or $C^1$) between adjacent patches, our algorithm can avoid the complicated procedures of surface trimming and patching. In our framework, the user simply specifies the degree $n$ of the triangular $B$-spline surface and fitting error tolerance $\epsilon$. The surface reconstruction procedure generates a single triangular $B$-spline patch that has $C^{n-1}$ continuity over smooth regions and $C^0$ on sharp features. More importantly, all the knots and control points are determined by minimizing a linear combination of interpolation and fairness functionals. Examples are presented which demonstrate the effectiveness of the technique for real data sets.*

## 1. Introduction

The challenging problem of reconstructing a surface from a large set of scattered sample points arises in a variety of applications including reverse engineering, geometric modeling and processing, graphics, vision, medical image segmentation, etc. In terms of the underlying surface representation, existing approaches fall into three categories: polygonal meshes, splines and zero-set surfaces. Among them, spline-based algorithms have been widely studied and employed since they are well suited for further processing in CAD/CAM systems.

Tensor-product $B$-splines and NURBS are currently the industrial standard for surface representation. However, due to their rectangular structures, they exhibit two major difficulties in scattered data fitting:

- A single $B$-spline patch can represent only surfaces of simple topological type. Thus, a surface of arbitrary topological type must be defined as a network of B-spline patches. It is challenging to enforce a certain degree of continuity between adjacent patches while at the same time fit the patch network to the points.
- Although it is desirable in principle to have surfaces that are as smooth as possible, in practice it is necessary to be able to model discontinuities like sharp edges or corners as well. However, duplicating knots in tensor-product $B$-spline will produce a discontinuity curve across the whole patch.

Triangular B-splines, or DMS splines, introduced by Dahmen, Micchelli and Seidel [1], have numerous positive characteristics that make them appropriate for surface reconstruction, such as their automatic smoothness properties, the ability to define a surface over arbitrary triangulations (which can be adapted to the local density of sampled data) and model sharp features between any desired adjacent knots [11].

Note that, the existing triangular $B$-spline-based approaches [3, 4, 7, 8, 11, 13] only deal with fixed knots. The number and positions of knots are determined by the distribution of parameters before fitting and are not allowed to be changed during fitting. The main reason for using splines with fixed knots is efficiency, since the basis functions can be precomputed. However, the fitting quality can, in principle, be improved if time-varying knots are allowed instead of fixed ones.

The main contribution of this paper is the development of a new algorithm based on triangular $B$-spline for surface reconstruction. This approach has the following features:

1. It generates a single triangular $B$-spline patch that has a user-specified continuity over both smooth regions and sharp features.
2. It can handle parametric domains with arbitrary topology with ease. Therefore, there is no need for surface trimming and patching.

3. The knots are first placed heuristically according to the features, and then are refined by minimizing a linear combination of interpolation and fairness functionals.

The rest of this paper is organized as follows: Section 2 reviews the related work on triangular B-splines and surface reconstruction methods. In Section 3, we present several theoretical results on triangular B-splines, such as the derivatives with respect to parameters and knots. Section 4 details our algorithm. Experimental results on several real data are demonstrated in Section 5. Finally, we conclude the paper in Section 6.

## 2. Previous Work

### 2.1. Triangular $B$-splines

The theoretical foundation of triangular $B$-splines lies in the simplex spline of approximation theory. Dahmen et al. [1] propose triangular $B$-splines from the point of view of blossoming. Fong and Seidel [3, 4] present the first prototype implementation of triangular $B$-splines and show several useful properties, such as affine invariance, convex hull, locality and smoothness. Greiner and Seidel [7] demonstrate the practical feasibility of multivariate $B$-spline algorithms in graphics and shape design. Pfeifle and Seidel [10] present an efficient algorithm to evaluate quadratic triangular $B$-splines, and they also demonstrate the fitting of a triangular B-spline surface to scattered functional data through the use of least squares and optimization techniques [11]. Han and Medioni [8] employ triangular NURBS for modeling and visualizing sparse, noisy data that may contain unspecified discontinuity edges and functions. Qin and Terzopolous [13] present dynamic triangular NURBS, a free-form shape model that demonstrates the convenience of interaction within a physics-based framework. Franssen et al. [5] propose an efficient evaluation algorithm, which works for triangular $B$-spline surfaces of arbitrary degree. He et al. [9] derive the formula to evaluate the directional derivatives of triangular $B$-spline with respect to knots.

### 2.2. Surface reconstruction with splines

There has been considerable work on surface reconstruction with splines. In order to handle objects with arbitrary topology, the reconstructed surface is usually represented as the collection of several patches. The patches are trimmed near the boundaries, which results in gaps between neighboring patches. The main effort goes into filling these gaps with properly chosen blending surfaces. For instance, Eck and Hoppe's method yields a $G^1$ tensor product B-spline surface [2] and Wagner et al's approach guarantees $C^2$-continuous result [16].

## 3. Triangular $B$-splines with free knots

### 3.1. Definition

The construction of the triangular $B$-spline scheme in [3, 4, 5, 7, 11] is as follows: let points $\mathbf{t}_i \in \mathbb{R}^2$, $i \in \mathbb{N}$, be given and define a triangulation

$$T = \{\Delta(I) = [\mathbf{t}_{i_0}, \mathbf{t}_{i_1}, \mathbf{t}_{i_2}] : I = (i_0, i_1, i_2) \in \mathcal{I} \subset \mathbb{N}^3\}$$

of a bounded region $D \subseteq \mathbb{R}^2$, where every triangle is oriented counter-clockwise (or clockwise). Next, with every vertex $\mathbf{t}_i$ of $T$ we associate a cloud of knots $\mathbf{t}_{i,0}, \ldots, \mathbf{t}_{i,n}$ such that $\mathbf{t}_{i,0} = \mathbf{t}_i$ and for every triangle $I = [\mathbf{t}_{i_0}, \mathbf{t}_{i_1}, \mathbf{t}_{i_2}] \in T$,

1. all the triangles $[\mathbf{t}_{i_0,\beta_0}, \mathbf{t}_{i_1,\beta_1}, \mathbf{t}_{i_2,\beta_2}]$ with $\beta = (\beta_0, \beta_1, \beta_2)$ and $|\beta| = \beta_0 + \beta_1 + \beta_2 \leq n$ are non-degenerate.
2. the set

   $$\Omega_n^I = interior(\cap_{|\beta| \leq n} X_\beta^I), X_\beta^I = [\mathbf{t}_{i_0,\beta_0}, \mathbf{t}_{i_1,\beta_1} \mathbf{t}_{i_2,\beta_2}]$$

   satisfies
   $$\Omega_n^I \neq \emptyset \tag{1}$$
3. if $I$ has a boundary edge, say, $(\mathbf{t}_i, \mathbf{t}_j)$, the entire area $[\mathbf{t}_{i,0}, \ldots, \mathbf{t}_{i,n}, \mathbf{t}_{j,0}, \ldots, \mathbf{t}_{j,n})$ must lie outside of $D$.

Then the triangular $B$-spline basis function $N_\beta^I$, $|\beta| = n$, is defined by means of simplex splines $M(\mathbf{u}|V_\beta^I)$ as

$$N(\mathbf{u}|V_\beta^I) = |d_\beta^I| M(\mathbf{u}|V_\beta^I)$$

where $V_\beta^I = \{\mathbf{t}_{i_0,0}, \ldots, \mathbf{t}_{i_0,\beta_0}, \ldots, \mathbf{t}_{i_2,0}, \ldots, \mathbf{t}_{i_2,\beta_2}\}$ and

$$d_\beta^I = d(X_\beta^I) = det \begin{pmatrix} 1 & 1 & 1 \\ \mathbf{t}_{i_0,\beta_0} & \mathbf{t}_{i_1,\beta_1} & \mathbf{t}_{i_2,\beta_2} \end{pmatrix}$$

is twice the area of triangle $X_\beta^I$.

Assuming (1), these $B$-spline basis functions can be shown to be all non-negative and to form a partition of unity. Hence, any triangular $B$-spline surface

$$\mathbf{F}(\mathbf{u}) = \sum_{I \in \mathcal{I}} \sum_{|\beta| = n} \mathbf{c}_{I,\beta} N(\mathbf{u}|V_\beta^I), \mathbf{c}_{I,\beta} \in \mathbb{R}^3 \tag{2}$$

lies in the convex hull of its control points.

This surface is globally $C^{n-1}$ if all the sets $X_\beta^I$, $|\beta| \leq n$ are affinely independent. In general, if at most $\mu$ knots within a domain triangle $\Delta(I)$ are collinear, $2 \leq \mu \leq n+2$, then $\mathbf{F}(\mathbf{u})$ is $C^{n+1-\mu}$-continuous everywhere.

The directional derivative of a degree $n$ simplex spline along a given direction $\mathbf{v} \in \mathbb{R}^2$ for a parameter value $\mathbf{u} \in \mathbb{R}^2$ is given as

$$\mathcal{D}_\mathbf{v} M(\mathbf{u}|V) = n \sum_{j=0}^{2} \mu_j(\mathbf{v}) M(\mathbf{u}|V \setminus \{\mathbf{t}_j\}),$$

where $\mathbf{v} = \sum_{j=0}^{2} \mu_j(\mathbf{v})\mathbf{t}_j$ and $\sum_{j=0}^{2} \mu_j(\mathbf{v}) = 0$. Thus, the directional derivative of a surface $\mathbf{F}$ at a parameter value $\mathbf{u}$ along the direction $\mathbf{v}$ has the expression

$$\mathcal{D}_{\mathbf{v}}\mathbf{F}(\mathbf{u}) = \sum_{I\in\mathcal{I}} \sum_{|\beta|=n} \mathbf{c}_{I,\beta} |d_\beta^I| \mathcal{D}_{\mathbf{v}} M(\mathbf{u}|V_\beta^I).$$

### 3.2. Shared control points

For a general triangular $B$-spline surface, each triangle $I$ has its own set of control points $\mathbf{c}_{I,\beta}$. However, in this paper we consider a more restricted class of surfaces by sharing respective control points along common boundaries of two adjacent triangles in the parametric triangulation.

Triangular $B$-splines with shared control points have several useful properties:

1. A degree $n$ surface can be evaluated with the efficiency of a degree $n-1$ surface [4], i.e.,

$$\mathbf{F}(\mathbf{u}) = \sum_{I\in\mathcal{I}} \sum_{|\beta|=n-1} \mathbf{c}_{I,\beta}^{(1)}(\mathbf{u}) N(\mathbf{u}|V_\beta^I), \qquad (3)$$

where

$$\mathbf{c}_{I,\beta}^{(1)}(\mathbf{u}) = \sum_{j=0}^{2} \mathbf{c}_{I,\beta+e^j} \lambda_j(\mathbf{u}|X_\beta^I),$$

$\lambda_j(\mathbf{u}|X_\beta^I)$ is the $j-th$ barycentric coordinate with respect to $X_\beta^I$ and $e^j = (\delta_{j,i})_{i=0}^2$, $j = 0, 1, 2$ are the coordinate vectors. This also implies that the last knot $\mathbf{t}_{i,n}$ associated to vertex $\mathbf{t}_i$ does not contribute to the shape.

2. The directional derivative can be written in the form of a degree $n-1$ surface [12], i.e.,

$$\mathcal{D}_{\mathbf{v}}\mathbf{F}(\mathbf{u}) = n \sum_{I\in\mathcal{I}} \sum_{|\beta|=n-1} \mathbf{c}_{I,\beta}^{(2)}(\mathbf{v}) N(\mathbf{u}|V_\beta^I), \qquad (4)$$

where

$$\mathbf{c}_{I,\beta}^{(2)}(\mathbf{v}) = \sum_{j=0}^{2} \mathbf{c}_{I,\beta+e^j} \mu_j(\mathbf{v}|X_\beta^I).$$

Equations (3) and (4) can significantly improve the software system for rendering a triangular $B$-spline surface, since the value $\mathbf{F}(\mathbf{u})$ and normal $\mathbf{F}_u(\mathbf{u}) \times \mathbf{F}_v(\mathbf{u})$ of a parameter $\mathbf{u}$ can be evaluated simultaneously. Hence, in the rest of this paper, we only consider triangular $B$-splines with shared control points.

### 3.3. Directional derivative with respect to a knot

Let us use $\mathcal{D}_{\mathbf{t}_l,\mathbf{v}}$ to denote the directional derivative with respect to a knot $\mathbf{t}_l$ along the direction $\mathbf{v}$, i.e.,

$$\mathcal{D}_{\mathbf{t}_l,\mathbf{v}} M(\mathbf{u}|\mathbf{t}_0,\ldots,\mathbf{t}_n) = \lim_{\varepsilon\to 0} \frac{M(\mathbf{u}|\mathbf{t}_0,\ldots,\mathbf{t}_l+\varepsilon\mathbf{v},\ldots,\mathbf{t}_n) - M(\mathbf{u}|\mathbf{t}_0,\ldots,\mathbf{t}_l,\ldots,\mathbf{t}_n)}{\varepsilon}.$$

The directional derivative of a triangular $B$-spline surface $\mathbf{F}$ with respect to knot $\mathbf{t}_{s,l}$, along the direction $\mathbf{v}$ is [9]

$$\mathcal{D}_{\mathbf{t}_{s,l},\mathbf{v}}\mathbf{F}(\mathbf{u}) = \mathcal{D}_{\mathbf{v}}\mathbf{G}(\mathbf{u}) + \mathbf{H}(\mathbf{u},\mathbf{v}), \qquad (5)$$

where

$$\mathbf{G}(\mathbf{u}) = -\frac{1}{n+1} \sum_{I\in\mathcal{I}, i_j=s} \sum_{|\beta|=n+1} \mathbf{c}_{I,\beta-e^j} N(\mathbf{u}|\widehat{V}_\beta^I),$$

$$\mathbf{H}(\mathbf{u},\mathbf{v}) = \sum_{I\in\mathcal{I}, i_j=s} \sum_{|\beta|=n, \beta_j=l} \mu_j(\mathbf{v}|X_\beta^I) \mathbf{c}_{I,\beta} N(\mathbf{u}|V_\beta^I),$$

and

$$\widehat{V}_\beta^I = \{\ldots, \mathbf{t}_{s,0}, \ldots, \mathbf{t}_{s,l-1}, \mathbf{t}_{s,l}, \mathbf{t}_{s,l}, \mathbf{t}_{s,l+1}, \ldots, \mathbf{t}_{s,n}, \ldots, \}.$$

Note that Equation (5) holds for a general triangular $B$-spline. According to Equation (4), for a triangular $B$-spline with shared control points, $\mathcal{D}_{\mathbf{v}}\mathbf{G}(\mathbf{u})$ can also be simplified as follows:

$$\mathcal{D}_{\mathbf{v}}\mathbf{G}(\mathbf{u}) = -\sum_{I\in\mathcal{I}, i_j=s} \sum_{|\beta|=n} \mathbf{c}_{I,\beta-e^j}^{(2)}(\mathbf{v}) N(\mathbf{u}|\widehat{V}_\beta^I).$$

### 3.4. Evaluation

Equation (5) shows that the computation of derivatives with respect to a knot relies only on the evaluation of two triangular B-splines, one with the same knot configuration but different control points, another with different knots but the same control points. Thus, it is straightforward to develop the evaluation algorithm for derivatives based on existing evaluation routines for triangular $B$-splines [5]. However, this is not efficient in practice. The reason is that the evaluation of $\mathbf{F}(\mathbf{u})$ and $\mathcal{D}_{\mathbf{t}_{s,l},\mathbf{v}}\mathbf{F}(\mathbf{u})$ share many simplex splines of lower degree. The evaluation process will be accelerated if every simplex spline of degree $i$, $i = 0, \ldots, n$, is computed only once. In our implementation, we treat the evaluation of $\mathbf{F}(\mathbf{u})$, $\mathcal{D}_{\mathbf{v}}\mathbf{F}(\mathbf{u})$ and $\mathcal{D}_{\mathbf{t}_{s,l},\mathbf{v}}\mathbf{F}(\mathbf{u})$ simultaneously.

Note that a simplex spline is given by the following recursive equation:

$$M(\mathbf{u}|V) = \begin{cases} \frac{\chi[\mathbf{t}_0,\mathbf{t}_1,\mathbf{t}_2)}{|d(V)|} & |V| = 3 \\ \sum_{j=0}^{2} \lambda_j(\mathbf{u}|W) M(\mathbf{u}|V \setminus \{w_i\}) & |V| > 3 \end{cases},$$

where $\chi_{[\mathbf{t}_0,\mathbf{t}_1,\mathbf{t}_2)}(\mathbf{u})$ is the characteristic function on $[\mathbf{t}_0, \mathbf{t}_1, \mathbf{t}_2)$. The elements in $W = \{w_0, w_1, w_2\} \subset V$ can be chosen arbitrarily from $V$. $W$ is called the *split set* for $V$.

The evaluation problem of re-using partial results depends on an efficient way to index and search all of the relevant basis functions of simplex splines. In a related work, Franssen et al. [5] present a directed graph data structure that makes searching related basis functions (to be evaluated) superfluous. In this paper, we further extend Franssen's idea to accommodate triangular $B$-splines with free knots. We present our improved data structure as follows:

```
class SimplexSpline {
public:
  // degree of this simplex spline
  int degree;
  // degree+3 knots
  double2* knots;
  // used to index a SimplexSpline
  int* knot_indices;
  // pointers to lower degree SimplexSplines
  SimplexSpline* M[3];
  // the last evaluation point
  double2 lastpoint;
  // value of last evaluation point
  double lastvalue;
  // pointer to the parametric domain
  Domain* dm;
    ...};

typedef SortedList<SimplexSpline>
        SimplexSplineList;

class Domain {
public:
  int degree;
  vector<double2> knots;
  vector<SimplexSplineList> ssl;
    ...};
```

The class *Domain* maintains an array of knots and a directed graph, in which every node represents a *SimplexSpline*. Each *SimplexSpline* of degree $i > 0$ has three outgoing edges that connect it with three different *SimplexSpline*s of degree $i - 1$. These three *SimplexSpline*s are determined by choosing a split set $W$ and unfolding the above recurrence. This directed graph is also organized in $n + 1$ layers, in which the $i$-th layer is a sorted list of *SimplexSpline*s of degree $i$. When constructing a new *SimplexSpline*, the procedure first (binary) searches the corresponding layer. If it has already been created, the procedure simply returns its pointer; otherwise, it inserts a new *SimplexSpline* into the list and unfolds it recursively until all the paths reach existing nodes or layer 0. This data structure has several advantages:

1. This graph is built only once during preprocessing and then can be used to evaluate the simplex splines at arbitrary locations.
2. Whenever we compute the value of a simplex spline at a parametric location, we store this value in the node of the simplex spline in the graph. When, during evaluation of the same point, we encounter the same simplex spline through another incoming edge and just use the stored value.
3. It can deal with evaluation of a point, normal, and derivatives with respect to knots in the same fashion. The only difference is that the evaluation of derivatives of knots starts from layer $n$ and others from layer $n - 1$.
4. If the position of a knot $\mathbf{t}_{s,l}$ is changed, we collect the simplex splines on the top layer that contains $\mathbf{t}_{s,l}$, and then update simplex splines of lower degree recursively through their outgoing edges.

## 4. Surface Reconstruction with Triangular $B$-splines

### 4.1. Problem statement

The problem of reconstructing smooth surfaces from discrete scattered data arises in many fields of science and engineering and has now been studied thoroughly for nearly 40 years. The problem can typically be stated as follows: given a set $P = \{\mathbf{p}_i\}_{i=1}^m$ of points $\mathbf{p}_i \in \mathbb{R}^3$, find a parametric surface $\mathbf{F} : \mathbb{R}^2 \to \mathbb{R}^3$ that approximates $P$.

To find a proper parametric domain $\Omega \subset \mathbb{R}^2$, parameterization is usually the first step. Parameterizing a point cloud $P$ is the task of finding a set of parameter points $\psi(\mathbf{p}_i) \in \Omega$, one for each point $\mathbf{p}_i \in P$. Then, we consider the following problem:

$$\min E(\mathbf{F}) = E_{dist}(\mathbf{F}) + \lambda \cdot E_{fair}(\mathbf{F}), \qquad (6)$$

where

$$E_{dist}(\mathbf{F}) = \sum_{i=1}^{m} \|\mathbf{p}_i - \mathbf{F}(\mathbf{u}_i)\|^2,$$

$\mathbf{u}_i = (u_i, v_i)^T$ is the parameter of point $\mathbf{p}_i$ and $E_{fair}(\mathbf{F})$ is a fairness functional with the smoothing factor $\lambda \geq 0$.

The commonly-used fairness functionals, such as simplified membrane energy and thin-plate energy, require integration, which is usually computational expensive. In this paper, we use a simple, yet effective, fairness functional

$$E_{fair}(\mathbf{F}) = \sum_{i=1}^{m} (\mathbf{n}_i \cdot \mathbf{F}_u(\mathbf{u}_i))^2 + (\mathbf{n}_i \cdot \mathbf{F}_v(\mathbf{u}_i))^2 \qquad (7)$$

where $\mathbf{n}_i$ is the normal of point $\mathbf{p}_i$. Note that these normals can be estimated either from initial scans during the shape

acquisition phase or by local least-squares fitting to $P$. This fairness functional can be seen as an approximation of surface normals.

Our triangular $B$-spline surface reconstruction algorithm consists of three steps: 1) constructing an initial domain triangulation, 2) fitting with triangular $B$-spline and 3) refining the domain triangulation adaptively.

### 4.2. Constructing an initial domain triangulation

Suppose the parameters $\mathbf{u}_i$ have been obtained by existing parameterization methods. A principle in constructing such a triangulation is that areas with dense parameter points should have more triangles and *vice versa*. Furthermore, placing primary knots along feature lines is also helpful in sharp feature recovery. Based on these observations, we construct the initial domain in three steps:

1. *Feature detection.* The detected and reconstructed features enable the splitting of the whole parametric domain into simpler sub-domains such that each subdomain represents a smooth surface patch.

2. *Domain partition.* First, uniformly sample the boundary of the domain and feature lines. Next, simplify the original mesh with the user-specified number of triangles. (In our implementation, *QSlim* [6] is used for this purpose.) Finally, map the simplified mesh to the parametric domain.

3. *Constrained Delaunay triangulation.* Set the boundary and feature constraints, and perform constrained Delaunay triangulation to the vertices generated by the above step. Refine the triangulation by removing triangles with small area.

### 4.3. Fitting with triangular $B$-splines

Although it is possible to treat control points and knots simultaneously during the optimization process, we prefer to handle them separately. There are several reasons for doing so: 1) Solving the sub-problem of control points is much easier and faster than the knots sub-problem; 2) The control points are more useful than the knots to construct an initial surface; and 3) This helps to reduce the complexity of the problem.

If only the control points are treated as variables in Equation (6), it falls into a very special category of nonlinear programming, i.e., unconstrained convex quadratic programming. For example, $E_{dist}$ has the following form:

$$E_{dist} = \frac{1}{2}\mathbf{x}^T Q\mathbf{x} + \mathbf{c}^T\mathbf{x} + f,$$

where $\mathbf{x} = (\dots, \mathbf{c}_{I,\beta}, \dots)^T$,

$$Q = \begin{pmatrix} & \vdots & \\ \dots & 2\sum_{i=1}^{m} N_{I,\beta}(u_i, v_i)N_{I',\beta'}(u_i, v_i) & \dots \\ & \vdots & \end{pmatrix},$$

$$\mathbf{c} = (\dots, -2\sum_{i=1}^{m} \mathbf{d}_i N^I_\beta(u_i, v_i), \dots)^T,$$

and $f = \sum_{i=1}^{m} \|\mathbf{d}_i\|^2$. $E_{fair}$ is also a quadratic function in the unknown of control points, and can be written in a similar fashion.

Note that, $Q$ is a semi-positive definite, symmetric and sparse matrix. For a typical triangular $B$-spline with 500 triangles in the domain, more than $99\%$ elements in $Q$ are zeros. Interior-point methods can solve this problem very efficiently.

When considering the knots as free variables in Equation (6), we need also to pay attention to the positions of knots. We classify the knots into two categories: the primary knots $\{\mathbf{t}_{s,0}|s \in \mathbb{N}\}$ and the sub-knots $\{\mathbf{t}_{s,l}|s \in \mathbb{N}, 1 \le l < n\}$.

The knots are subject to two kinds of constraints:

1. Domain constraint: the primary knots must yield a valid triangulation in $\Omega$ and the sub-knots must satisfy Equation (1). The sub-knots on the boundary must lie outside of $\Omega$. Traas's scheme [15] can guarantee Equation (1): for every vertex $\mathbf{t}_i$, place all the sub-knots $\mathbf{t}_{i,j}$, $j = 1, \dots, n$ within a circle $c_i$ where $c_i$ does not intersect with any middle line of triangles associated with $\mathbf{t}_i$ (See Figure 1(a)).

2. Feature constraint: the primary knots lie on a sharp feature curve and the sub-knots lie on an edge between adjacent primary knots (See Figure 1(b)).

The domain constraint is necessary for all free knots splines. Feature constraint is useful to model discontinuities such as boundaries, sharp edges and corners. Therefore, Equation (6) is a typical large-scale constrained nonlinear programming problem. In our first implementation, we treat all the free knots as variables in Equation (6) and solve it using a general nonlinear programming package. However, the performance is unsatisfactory even after we improve the evaluation of the objective function and its gradient. Observe that not all the knots have the same contribution to the objective function. Therefore, there is no need to optimize a knot if it can only change the shape slightly. Hence, we develop the procedure *OptimizeKnot*$(\mathbf{t}_s)$, which only finds the optimal positions of knots associated with vertex $\mathbf{t}_s$.

Let us use $T(\mathbf{t}_s, k)$ to denote the $k$-ring $(k \ge 1)$ neighboring triangles surrounding $\mathbf{t}_s$. Let $P(\mathbf{t}_s) \subset P$ be the scattered points whose parameters are in the triangles $T(\mathbf{t}_s, 2)$.

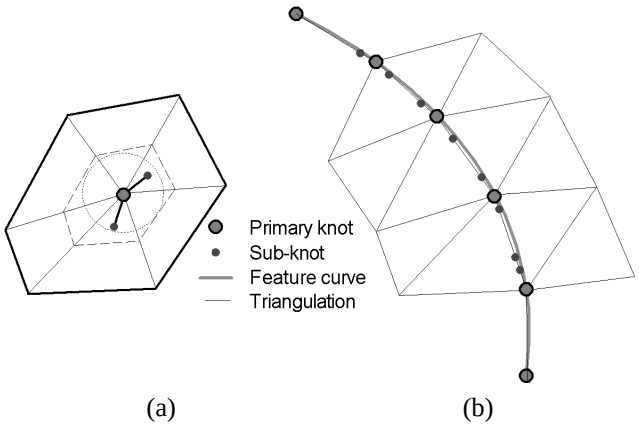



**Figure 1. Constraints on knots**

The goal of *OptimizeKnot*($\mathbf{t}_s$) is to minimize the following objective function (compare this to Equation (6))

$$\min \sum_{\mathbf{p}_i \in P(\mathbf{t}_s)} \{\|\mathbf{p}_i - \mathbf{F}(\mathbf{u}_i)\|^2 + \lambda((\mathbf{n}_i \cdot \mathbf{F}_u(\mathbf{u}_i))^2 + (\mathbf{n}_i \cdot \mathbf{F}_v(\mathbf{u}_i))^2)\} \quad (8)$$

with respect to $\{\mathbf{t}_{s,0}, \ldots, \mathbf{t}_{s,n-1}\}$ which are subject to
(1) $\mathbf{t}_{s,0} \in T(\mathbf{t}_s, 1)$.
(2) $\|\mathbf{t}_{s,i} - \mathbf{t}_{s,0}\| \leq r$ for $i = 1, \ldots, n-1$ where $r$ (the radius of circle in Figure 1(a)) is half of the minimal height of triangles in $T(\mathbf{t}_s, 1)$.
(3) If the user wants to enforce $C^0$ continuity on feature lines, then $\{\mathbf{t}_{s,i}\}_{i=1}^{n-1}$ must lie on the edge between adjacent primary knots that are also on the sharp feature curve.

Equation (8) is a local fitting problem that considers only scattered data points which are in $\mathbf{t}_s$'s 1-ring neighboring triangles. Since *OptimizeKnot*($\mathbf{t}_s$) decreases the objective functional around $\mathbf{t}_s$, the algorithm can reach local minimum for each vertex. Another advantage of this algorithm is parallelism. If vertices $\mathbf{t}_p$ and $\mathbf{t}_q$'s topological distance is more than 4, *OptimizeKnot*($\mathbf{t}_p$) and *OptimizeKnot*($\mathbf{t}_q$) can be done in parallel since any change of $\{\mathbf{t}_{p,i}\}_{i=0}^{n-1}$ does not affect the local shape around $\mathbf{t}_q$.

### 4.4. Adaptive refinement

The surface fitting algorithm described in Section 4.3 attempts to minimize the total squared distance of the scattered data points $\mathbf{p}_i$ to the triangular $B$-spline surface $\mathbf{F}$. It is often desirable to specify an error tolerance $\epsilon$, such that the surface satisfies $E_{dist} \leq \epsilon$. Similar to Eck and Hoppe's method [2], we use adaptive refinement to introduce new degrees of freedom into the surface representation in order to improve the fitting quality. The goal of this refinement is to subdivide any domain triangle whose fitting error is greater than a threshold. This step is performed as follows:

1. *Repeat*
2. Subdivide the domain triangles with large fitting error.
3. Flip edges to avoid poor quality triangles.
4. Solve the control points sub-problem for affected triangles.
5. Call *OptimizeKnot* for the new vertices.
6. *until* $E_{dist} \leq \epsilon$

Essentially, Step 2 is the knot insertion for triangular $B$-splines. Seidel et al. [14] prove that the new control points in the refined triangulation can be computed directly by the polar form. To simplify our implementation, in this paper we simply solve the control points sub-problem to calculate the control points.

## 5. Experimental Results

Figures 2(a-j) illustrate our surface reconstruction method applied to the skidoo model. We also present other examples in Figure 3. In order to compare the fitting error across different models, we uniformly scale the data points $P$ to fit within a unit cube.

Table 1 indicates the configurations of these data sets and the surface complexities. In this 3-step pipeline, Step 1 can be implemented by various existing methods. Therefore, only execution times of Step $2-3$ are shown in Table 1. The execution times are obtained on a Pentium IV machine at $2.4$ GHz.

As seen in Figure 2(b), the parametric domain of the skidoo has a very irregular boundary and two holes. The horse model also has an oval-like domain. Due to the distortion of parameterization, the distribution of parameter points is not even, e.g., the dense area of the horse parameterization corresponds to the ears and nose. With the triangular $B$-spline, we can construct the parametric domain easily according to the point distribution and sharp features. Furthermore, since our surface reconstruction algorithm is based on global parameterizations, there is no cutting and patching work, which is usually necessary when using tensor-product $B$-splines.

## 6. Conclusions

In this paper we developed a new algorithm for surface reconstruction based on triangular $B$-splines which has several advantages: it can handle a parametric domain with arbitrary topology; it generates a single triangular $B$-spline patch that has user-specified continuity over both smooth regions and sharp features; and the positions of knots and control points can be determined automatically. Our experimental examples demonstrate that we can achieve high continuity and good fitting results when using triangular $B$-splines for surface reconstruction.

| object | #points | degree | #domain triangles | #control points | max. error | root-mean-square error | time(m:s) |
|---|---|---|---|---|---|---|---|
| horse | 24,236 | 3 | 364 | 1,663 | 1.04e-2 | 1.09e-3 | 5:26 |
| skidoo | 37,974 | 4 | 464 | 3,863 | 3.24e-3 | 1.31e-4 | 13:16 |
| venus | 50,002 | 3 | 1,055 | 4,831 | 6.36e-3 | 9.74e-4 | 22:51 |

**Table 1. Surface complexities and execution times**

## 7. Acknowledgments

We wish to thank Hugues Hoppe for the parameterization data and Michael Franssen for the code of evaluation of triangular $B$-spline. Thanks are also due to Kevin T. McDonnell for proofreading this manuscript. This research was supported in part by the NSF grants IIS-0082035 and IIS-0097646, and an Alfred P. Sloan Fellowship.

## References

[1] W. Dahmen, C. A. Micchelli, and H.-P. Seidel. Blossoming begets $B$-spline bases built better by $B$-patches. *Mathematics of Computation*, 59(199):97–115, 1992.

[2] M. Eck and H. Hoppe. Automatic reconstruction of b-spline surfaces of arbitrary topological type. In *Proceedings of SIGGRAPH96*, pages 325–334. ACM Press, 1996.

[3] P. Fong and H.-P. Seidel. An implementation of multivariate $B$-spline surfaces over arbitrary triangulations. In *Proceedings of Graphics Interface '92*, pages 1–10, 1992.

[4] P. Fong and H.-P. P. Seidel. Control points for multivariate $B$-spline surfaces over arbitrary triangulations. *Computer Graphics Forum*, 10(4):309–317, 1991.

[5] M. Franssen, R. C. Veltkamp, and W. Wesselink. Efficient evaluation of triangular $B$-spline surfaces. *Computer Aided Geometric Design*, 17:863–877, 2000.

[6] M. Garland and P. S. Heckbert. Surface simplification using quadric error metrics. In *Proceedings of SIGGRAPH97*, pages 209–216. ACM Press, 1997.

[7] G. Greiner and H.-P. Seidel. Modeling with triangular $B$-splines. *IEEE Computer Graphics and Applications*, 14(2):56–60, Mar. 1994.

[8] S. Han and G. Medioni. Triangular NURBS surface modeling of scattered data. In *Proceedings of the Conference on Visualization*, pages 295–302, 1996.

[9] Y. He, H. Qin, and X. Gu. Triangular $B$-splines with free knots. submitted, 2003.

[10] R. Pfeifle and H.-P. Seidel. Faster evaluation of quadratic bivariate DMS spline surfaces. In *Proceedings of Graphics Interface '94*, pages 182–189, 1994.

[11] R. Pfeifle and H.-P. Seidel. Fitting triangular $B$-splines to functional scattered data. In *Graphics Interface '95*, pages 26–33, 1995.

[12] H. Prautzsch, W. Boehm, and M. Paluszny. *Bézier and $B$-Spline Techniques*. Springer Verlag, October 2002.

[13] H. Qin and D. Terzopoulos. Triangular NURBS and their dynamic generalizations. *Computer Aided Geometric Design*, 14(4):325–347, 1997.

[14] H.-P. Seidel and A. H. Vermeulen. Simplex splines support surprisingly strong symmetric structures and subdivision. In *Curves and Surfaces II*, pages 443–455. AK Peters, 1994.

[15] C. R. Traas. Practice of bivariate quadratic simplicial splines. In *Computation of curves and surfaces (Puerto de la Cruz, 1989)*, volume 307 of *NATO Adv. Sci. Inst. Ser. C Math. Phys. Sci.*, pages 383–422. Kluwer Acad. Publ., Dordrecht, 1990.

[16] M. Wagner, K. Hormann, and G. Greiner. $C^2$-continuous surface reconstruction with piecewise polynomial patches. Preprint, September 2003.

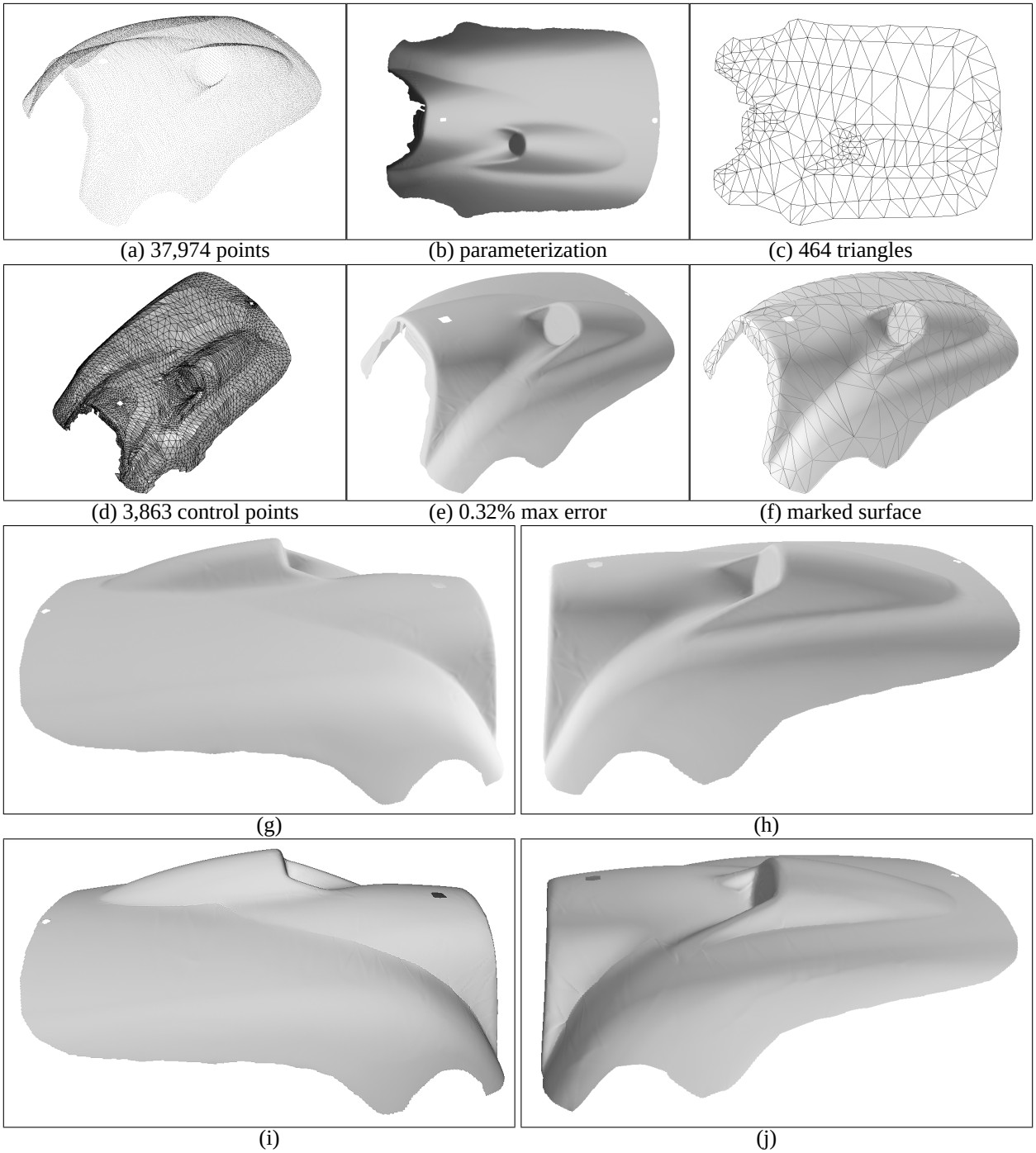
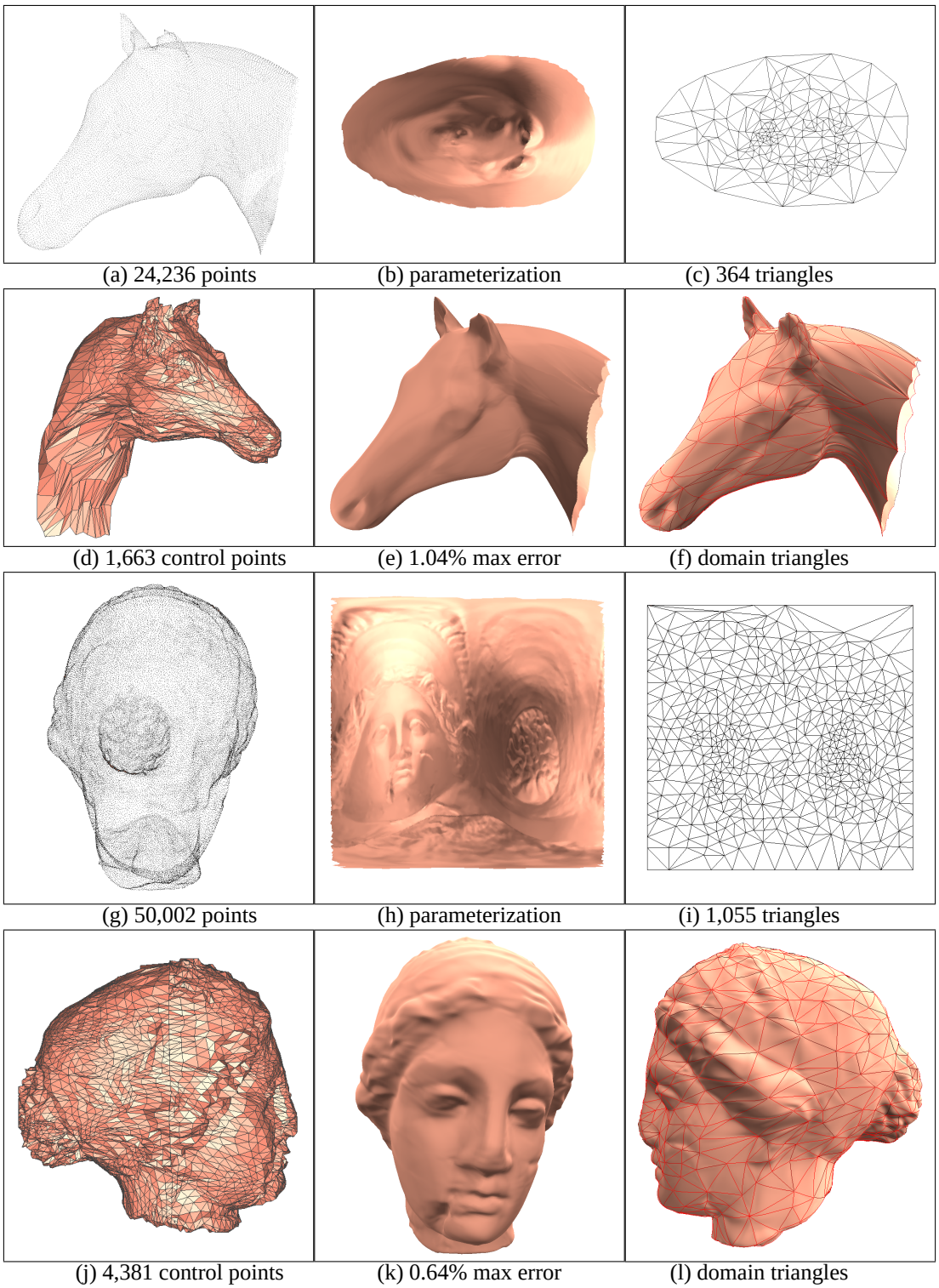



**Figure 2. Illustration of the triangular $B$-spline surface reconstruction procedure. (a) 3D points. (b) parameterization. (c) parametric domain. (d) control net. (e) A $C^3$ triangular $B$-spline surface with maximal fitting error of 0.32%. (f) surface marked with domain triangles. (g)-(h) closed view of reconstructed surface without feature recovery. (i)-(j) closed view of reconstructed surface with feature recovery.**



(a) 24,236 points (b) parameterization (c) 364 triangles

(d) 1,663 control points (e) 1.04% max error (f) domain triangles

(g) 50,002 points (h) parameterization (i) 1,055 triangles

(j) 4,381 control points (k) 0.64% max error (l) domain triangles

**Figure 3. $C^2$ triangular $B$-spline surfaces. (Parameterization data courtesy of Hugues Hoppe)**