Vis Comput (2010) 26: 277–286
DOI 10.1007/s00371-009-0410-9

ORIGINAL ARTICLE

Polynomial splines over general T-meshes

Xin Li · Jiansong Deng · Falai Chen

Published online: 8 December 2009

Abstract The present authors have introduced polynomial splines over T-meshes (PHT-splines) and provided theories and applications for PHT-splines over hierarchical T-meshes. This paper generalizes PHT-splines to arbitrary topology over general T-meshes with any structures (GPT-splines). GPT-spline surfaces can be constructed through a unified scheme to interpolate the local geometric information at the basis vertices of the T-mesh. We also discuss general edge insertion and removal algorithms for GPT-splines. As applications, we present algorithms to construct a GPT-spline surface from a quadrilateral mesh and to simplify a tensor-product B-spline surface into a GPT-spline surface with superfluous edges removal.

Keywords Splines · PHT-splines · T-meshes · Basis function · Simplification

1 Introduction

Computer graphics and computer-aided design prefer parametric surfaces, especially tensor-product B-spline surfaces to represent free-form surfaces [5]. However, tensor-product B-spline surfaces suffer from the major weakness that the control points must lie topologically in a rectangular grid which will induce many superfluous control points in B-spline representations. Furthermore, local refinement of B-spline surfaces by knot insertion will influence entire rows or columns of the control grid. To overcome the problem, Forsey et al. introduced hierarchical B-splines [6–8]. Hierarchical B-splines were also studied by Kraft [9], who constructed multilevel spline spaces which are a linear span of tensor-product B-splines on different, hierarchically ordered grid levels. In order to eliminate most superfluous control points and permit real local refinement, Sederberg et al. [11, 12] invented the notions of T-splines. T-splines support many valuable operations within a consistent framework. The present authors introduced polynomial spline spaces over T-meshes (PHT-splines) in [2], where the function on each cell is a bi-degree polynomial, and achieves specified smoothness across the common edges. And we further constructed basis functions for spline spaces over hierarchical T-meshes and discussed their applications in surface fitting and stitching in [3, 10].

The present paper generalizes the PHT-splines to arbitrary topology over general T-meshes with any structures which have more flexibility in surface modeling and finite element analysis. The main contributions of the present paper include the following issues.

- Construct the basis functions for GPT-splines which must be nest structures in former papers;
- Provide a general edge insertion and edge deletion algorithm. In the former papers, we could only insert and remove edges of the highest level directly;
- Define a unified interpolation scheme for arbitrary topology T-mesh;
- Provide an algorithm to simplify a tensor-product B-spline surface into a GPT-spline surface with superfluous edges removal.

X. Li is currently a research fellow in School of Computer Engineering, NTU, Singapore.

X. Li (✉) · J. Deng · F. Chen
Department of Mathematics, University of Science and Technology of China, Hefei, Anhui 230026, China
e-mail: lixustc@ustc.edu.cn

J. Deng
e-mail: dengjs@ustc.edu.cn

F. Chen
e-mail: chenfl@ustc.edu.cn

Although the bi-cubic spline surfaces have only C^1 global continuity, they have excellent locality which makes some operators, such as interpolation, approximation and fairing, not involve a big linear system of constraints. Furthermore, GPT-spline is always polynomial instead of rational. Thus geometric computation with GPT-splines is simpler and less costly. Furthermore, the local refinement does not need a whole chain of additional refinement steps.

The remainder of the paper is organized as follows. Section 2 recalls some preliminary knowledge about PHT-spline surfaces. We construct the basis functions for GPT-splines and describe the edge insertion and edge deletion operators in Sect. 3. Section 4 discusses a unified interpolation scheme for GPT-splines. Section 5 provides a tensor-product B-spline surface simplification algorithm, which approximates a given tensor-product B-spline surface with a GPT-spline. In the end, Sect. 6 concludes the paper with some summaries and future work.

2 Polynomial splines over T-meshes

In [2, 3], we provided the notions for spline over T-meshes and the theory and application for the spline over special T-meshes with nested structures. Given a T-mesh $\mathcal{T} \in \mathbb{R}^2$, let $\mathcal{F}$ denote all the cells in $\mathcal{T}$ and Ω the region occupied by all the cells in $\mathcal{T}$. Define

$$\mathcal{S}(m,n,\alpha,\beta,\mathcal{T}) := \left\{ s \in C^{\alpha,\beta}(\Omega) \mid s|_\phi \in \mathrm{P}_{mn}, \forall \phi \in \mathcal{F} \right\},$$

where P_{mn} is the space of all the polynomials with bi-degree (m,n), and $C^{\alpha,\beta}(\Omega)$ is the space consisting of all the bivariate functions which are continuous in Ω with order α along s direction and with order β along t direction. It is obvious that $\mathcal{S}(m,n,\alpha,\beta,\mathcal{T})$ is a linear space, which is called the spline space over the given T-mesh $\mathcal{T}$. According to Theorem 4.2 in [2], it follows that

$$\dim \mathcal{S}(2\alpha+1, 2\beta+1, \alpha, \beta, \mathcal{T}) = (\alpha+1)(\beta+1)\left(V^b + V^+\right),$$

where V^b and V^+ represent the number of the boundary vertices and the interior crossing vertices, respectively. Hereafter, boundary vertices and interior crossing vertices are called *regular vertices*.

For any function $b(u,v)$ defined on the T-mesh, *geometric information* at some parametric position (u_0, v_0) is the function value, the first order partial derivatives and the mixed partial derivative of $b(u,v)$ at (u_0, v_0). If all these values are zero, we say that the function $b(u,v)$ has geometric information vanished at (u_0, v_0).

A T-vertex is called a *share T-vertex* of two adjacent faces if the T-vertex belongs to both faces and lies in the interior of an edge of one face. For example, in Fig. 1, v_3 is a share T-vertex of faces F_1 and F_3. It is also a share T-vertex of faces F_2 and F_3 but is not a share T-vertex of faces F_1 and F_2. Share-vertex is very important for construction of basis functions.

3 Construction of basis functions

In [3], the authors constructed the basis functions for the coarse tensor-product mesh first and then modified the basis functions level by level according to the mesh's nested structures. In this section, we construct a set of basis functions for GPT-splines. The algorithm has several advantages compared with that in [3].

- The method suits for any type of T-mesh which must have nested structure in [3];
- The basis functions are represented in B-spline form, which in [3] are represented in Bézier form. Only we have almost 1/4 of the terms in this representation for each basis function.

Suppose we are given a T-mesh such as that in Fig. 1a, we can get a new T-mesh by extending all the T-vertices to

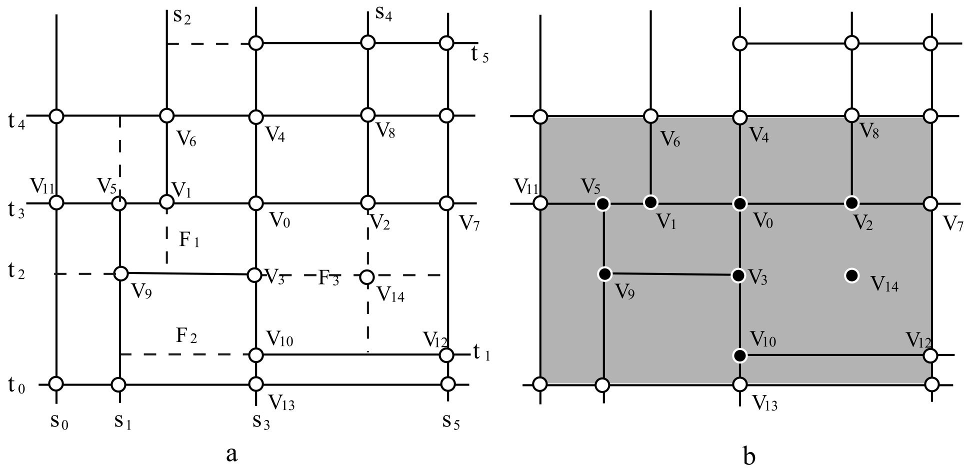

Fig. 1 A possible T-mesh and the notations for basis functions construction of v_0

the opposite edge (the red dashed lines in Fig. 1a). In the new T-mesh, we assume all the knots with multiplicity two and, at the vertex V_i, we can define four B-spline functions $N_i(u,v) = (N_{i1}(u,v), N_{i2}(u,v), N_{i3}(u,v), N_{i4}(u,v))$ by the knots for the four nearest regular vertices in four directions. For example, the four B-spline functions associated with v_2 are the four B-spline basis functions defined by the knot sequences $[s_3, s_3, s_4, s_4, s_5, s_5] \times [t_2, t_2, t_3, t_3, t_4, t_4]$ and those for v_{10} are the B-spline basis functions defined by the knot sequences $[s_1, s_1, s_3, s_3, s_5, s_5] \times [t_0, t_0, t_1, t_1, t_2, t_2]$. We also define the four basis functions for the new cross-vertices by the T-vertices extension.

One observation is that N_i have geometric information vanished at other vertices in the original T-mesh. The other observation is that every basis function in $\mathcal{S}^{1,1}(\mathcal{T})$ can be represented as the linear combination of N_i. So the four basis functions $b_j(u,v)$ associated with the jth regular vertex v_j can be represented in the form

$$b_j(u,v) = \sum_{i \in L(j)} N_i(u,v) c_j^i. \tag{1}$$

Here $b_j(u,v)$, $N_i(u,v)$ both are 1×4 vectors and c_j^i is a 4×4 matrix. $L(j)$ is a set of indices of the T-vertices which have non-zero c_j^i.

We compute $b_j(u,v)$ in the following four steps.

1. Forming set $L(j)$ for the jth vertex;
2. Computing the geometric information at the jth vertex;
3. Computing the geometric information at other vertices;
4. Computing c_j^i.

Forming $L(j)$ (Step 1) We illustrate the algorithm for computing $L(j)$ with an example in Fig. 1. The algorithm will check the vertex itself and its neighbor vertices recursively. Figure 1 shows a T-mesh from which we wish to compute the $L(0)$ for basis function associated with basis vertex v_0. First, we add v_0 into the set and check the four neighboring vertices v_1, v_2, v_3 and v_4. As v_i, $i = 1, 2, 3$, are share T-vertices of two adjacent faces of edge $v_0 v_i$, so we add v_1, v_2, v_3 into the set. In the iterative of checking v_1, v_2, v_3, we will add v_5 and v_{10} into the set according to the same fashion. In the next iterative, we will add v_9 into the set when we check vertex v_5. The algorithm will terminate when no new vertices will be added into the set when we check the vertices added in the last iterative. All vertices in $L(0)$ are marked with black in Fig. 1b.

Computing the geometric information at the jth vertex (Step 2) In this step, we first find a rectangle containing inside it all the vertices in $L(j)$. Suppose the position of the left-down and right-up points of the rectangle are (u_1, v_1) and (u_2, v_2), the position for jth basis vertex is (u_0, v_0); then the geometric information for $b_j(u,v)$ at (u_0, v_0) is chosen to be that of the four B-spline functions defined by knot sequences $[u_1, u_1, u_0, u_0, u_2, u_2] \times [v_1, v_1, v_1, v_1, v_2, v_2]$ at (u_0, v_0).

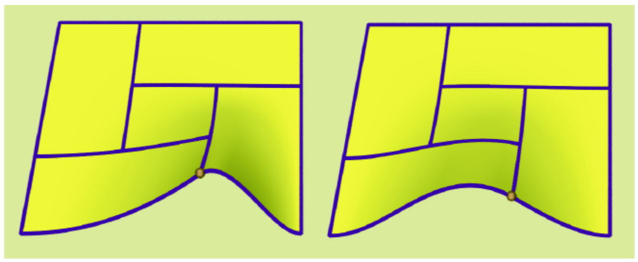
Fig. 2 Two images of basis functions

Computing the geometric information at other vertices (Step 3) The other c_j^i are computed by solving a linear system which comes from the observation that two rows of Bézier coefficients should be C^3 along the edge if the middle point is a T-vertex. For example in Fig. 1, the geometric information for v_0, v_3 and v_{10} should have four constraints. This will give four linear functions for any T-vertex in L_j. And the geometric information can be solved from this linear system by assigning the geometric information to be vanished at other basis vertices.

Computing c_j^i (Step 4) c_j^i can be computed by interpolating the geometric information at each vertex according to the given geometric information.

The above algorithm is very efficient since in general this matrix of the linear system is almost diagonal. Figure 2 illustrates an example for two basis functions over a T-mesh without nested structure. So we cannot apply the method given in [3] to construct the basis functions. The color lines are the images of mapping the T-mesh to the surface with moving a little in the normal direction.

The basis functions constructed have the following properties:

– *Nonnegativity*: This follows from the similar idea from the Powell-Sabin splines [4].
– *Partition of Unity*: Notice that the sum of the four basis functions $b_j(u,v)$ has the same geometric information as identical function. From this follows that the basis functions are a partition of unity.
– *Local Support*: The support of the basis functions is the union of all the supports of the B-spline functions $N_i(u,v)$.

Given a T-mesh $\mathcal{T}$, suppose the basis functions are $\{b_j(u,v)\}$, $j = 1, \ldots, N$, here N is the number of the basis vertices. Then a spline surface over $\mathcal{T}$ can be defined as

$$\mathbf{S}(u, v) = \sum_{j=1}^{N} \mathbf{C}_j b_j(u, v), \tag{2}$$

where $\mathbf{C}_j$ are control points associated with the jth basis vertex.

3.1 Edge insertion and edge deletion

Local refinement is the most important operator for splines, which is corresponding to the edge insertion operator for splines over T-meshes. Edge deletion is the inverse operator of the edge insertion. In this section, we will discuss the general edge insertion and deletion algorithms for spline surface $\mathbf{S}(u, v)$ in the form of (2).

Suppose we insert an edge into T-mesh $\mathcal{T}$ and get a new T-mesh $\mathcal{T}^1$. Edge insertion means how to define the surface $\mathbf{S}(u, v)$ over the T-mesh $\mathcal{T}^1$.

The first step of the operation is to construct the basis functions for the new T-mesh. Without loss of generality, suppose the edge insertion leads to one new basis vertex v_{new}. First we can construct the basis functions for the new basis vertex according to Sect. 3. Suppose the four new basis functions are $n_i(u, v), i = 0, \ldots, 3$. Then for any existing basis function $b_j^k(u, v)$, it will be replaced by $\hat{b}_j^k(u, v) = b_j^k(u, v) - \sum_{i=0}^{3} \lambda_i n_i(u, v)$. Here λ_i's satisfy the condition that $b_j^k(u, v)$ and $\sum_{i=0}^{3} \lambda_i n_i(u, v)$ have the same geometric information at v_{new}. So $\hat{b}_j^k(u, v)$ will have geometric information vanished at v_{new}. We show a modification of one basis function with some edge insertion in Fig. 3.

The second step is to modify the control points. Here we just compute the new control points and keep the existing control points unchanged. First, we compute the geometric information of $\mathbf{S}(u, v)$ at the new basis vertices and then interpolate this geometric information to get the new control points.

Edge deletion is the inverse operator of the edge insertion. Suppose the T-mesh after an edge removal is $\mathcal{T}^2$ and the corresponding spline space is $\mathcal{S}^2$. Since edge deletion is not an exact operator, there are many opinions how to define the surface after edge being removed. The opinion we use here is to keep the geometric information at all the existing basis vertices unchanged.

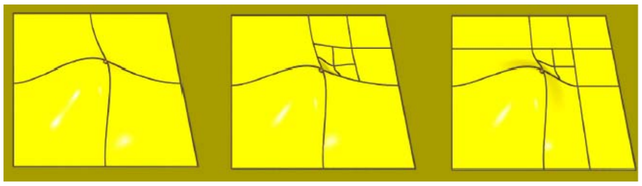

Fig. 3 The modification of one basis function with general edge insertion operator

As we have mentioned, some basis vertices will disappear with an edge being removed. When an edge is removed, three cases may appear according to the number of removed basis vertices. In the first case, two T-junctions disappear and in this case the spline surface is kept unchanged with the edge removal. In the second or third case, respectively one and two basis vertices disappear after the edge removed.

For any basis function $B(u, v)$, suppose the modification of $B(u, v)$ in the new spline space is $R(u, v)$.

If the deleted edge belongs to the first case, we apply an inverse Bézier subdivision in [5] directly to compute the Bézier ordinates for $R(u, v)$ in $\mathcal{T}^2$ from $B(u, v)$.

If the deleted edge belongs to the second case, a basis vertex $\mathbf{v}$ disappears when we delete the edge. Suppose the four basis functions associated with $\mathbf{v}$ are $b_k(u, v)$, $k = 0, \ldots, 3$. For any other basis function $B(u, v) \in \mathcal{S}^1$, since $R(u, v) \in \mathcal{S}^1$, $R(u, v)$ is a linear combination of all the basis functions of $\mathcal{S}^1$. Here we let all the coefficients be zero except those of $B(u, v)$ and $b_k(u, v)$. We also let the coefficient of $B(u, v)$ be one. So there exist constants λ^k, $k = 0, \ldots, 3$, such that

$$R(u, v) = B(u, v) + \sum_{k=0}^{3} \lambda^k b_k(u, v). \tag{3}$$

As $R(u, v)$ is C^3 along the removed edge, so (3) is equivalent to a system of linear equations with four unknowns λ^k, $k = 0, \ldots, 3$. It is evident that the linear system has a solution. Now we will prove that the linear system has a unique solution. In fact, suppose α^k and β^k are two solutions for the linear system, $R_1(u, v)$ and $R_2(u, v)$ are corresponding basis functions; then according to the interpolation theorem, we have $R_1(u, v) = R_2(u, v)$. That means $\sum_{k=0}^{3}(\alpha^k - \beta^k) b_k(u, v) = 0$. So $\alpha^k = \beta^k$.

If the deleted edge belongs to the third case, two basis vertices disappear when we delete the edge. Suppose the eight basis functions associated with the two basis vertices are $b_k(u, v)$, $k = 0, \ldots, 3$ and $c_k(u, v)$, $k = 0, \ldots, 3$, respectively. Then there exist constants λ^k, $k = 0, \ldots, 3$, and μ^k, $k = 0, \ldots, 3$, such that

$$R(u, v) = B(u, v) + \sum_{k=0}^{3} \lambda^k b_k(u, v) + \sum_{k=0}^{3} \mu^k c_k(u, v). \tag{4}$$

Similarly, (4) is equivalent to linear systems with λ^k and μ^k being respectively unknowns. According to the dimension formula, the solutions of these linear systems also exist and are unique.

In both cases, we can compute the Bézier ordinates for $R(u, v)$ in $\mathcal{T}^1$. It is easy to compute the Bézier ordinates for $R(u, v)$ in $\mathcal{T}^2$ according to Bézier subdivision algorithm in [5].

We show a modification of one basis function with some edge removal in Fig. 4.

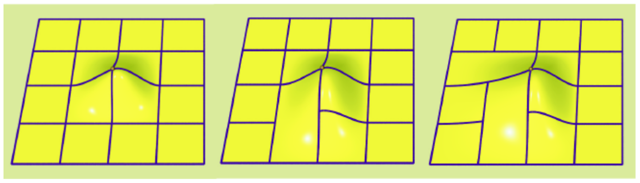

Fig. 4 The modification of one basis functions with general edge removal operator

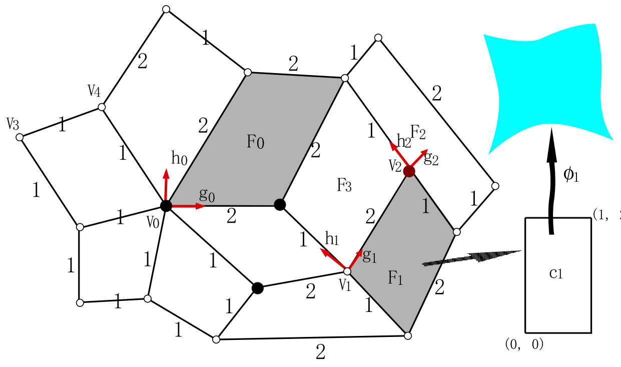

Fig. 5 A possible T-mesh in R^3

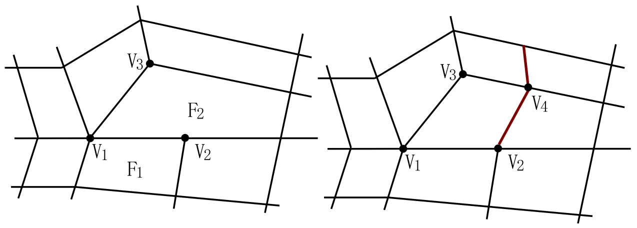

Fig. 6 Local refinement (red edge in the second picture) to make any two extraordinary faces have no share T-junctions

4 Arbitrary topology PHT-spline

However, the GPT-splines defined in the previous section can only model objects with simple topology. In this section, we will define an arbitrary topology G^1 GPT-spline by generalizing the interpolation algorithm to extraordinary vertices.

A T-mesh $\mathcal{T} \in \mathbb{R}^3$ is a mesh with all faces to be quadrilateral, which allows for T-junctions (T-vertex); see Fig. 5 for an example. The basic elements for a T-mesh are vertices, edges and faces. The grid point in a T-mesh is also called *vertex* of the mesh. The number of *edges* incident to the vertex is also referred to as the *valence* of the vertex. Each *face* F_i corresponds to a rectangle domain c_i in $\mathbb{R}^2$ whose width and length are assigned to the corresponding edges. We also call them the *knot interval* for the edge. The knot intervals on the opposite edges of a face should be the same. *The extraordinary vertex* (the black vertices in Fig. 5) is ineluctable for arbitrary topology spline surface. A face F_i is called *extraordinary face* if the face has some extraordinary vertices: F_0 in Fig. 5 is an extraordinary face. In the present paper, regular vertex and extraordinary vertex are called *basis vertices*.

A G^1 GPT-spline surface over the T-mesh $\mathcal{T} \in \mathbb{R}^3$ is a collection of maps $\Phi|_{c_i} = \phi_i : c_i \mapsto \mathbb{R}^3$, such that ϕ_i and ϕ_j are in tangent plane continuity if F_i and F_j are adjacent.

Let g_k and h_k be two independent unit orthogonal vectors lying in the tangent plane of Φ at v_k, $k = 0, 1, \ldots, N$; then the following interpolation problem can be considered for GPT-splines. Given a set of values or vectors f^j, f_u^j, f_v^j for extraordinary vertices V_j and f^i, f_u^i, f_v^i, f_{uv}^i for other basis vertices V_i, find $\Phi|_{c_i} = \phi_i$ such that

1. $\Phi(V_i) = f^i, D_{g_i}\Phi(V_i) = f_u^i, D_{h_i}\Phi(V_i) = f_v^i$, $D_{g_i h_i}\Phi(V_i) = f_{uv}^i$ for regular vertices V_i,
2. $\Phi(V_j) = f^j, D_{g_j}\Phi(V_j) = f_u^j, D_{h_j}\Phi(V_j) = f_v^j$ for extraordinary vertices V_j.

In the next section, we will express the maps Φ in Bézier form according to the geometric information at all the vertices.

4.1 Bézier forms

In this section, we will express the map Φ in Bézier form according to the geometric information at all the vertices. We do not need to separate the extraordinary vertices and all the maps can be computed by a unified interpolation scheme. In the scheme, ϕ_i are bi-cubic polynomial maps except those for extraordinary faces, which are bi-quintic maps.

Because the maps have different degrees, not all T-meshes are acceptable T-meshes for the scheme. A T-mesh is *acceptable T-mesh* if any two extraordinary faces have no share T-vertices. Any T-meshes can turn to be acceptable T-meshes by local refinements. For example in Fig. 6, T-vertex V_2 is a share T-vertex of extraordinary faces F_1 and F_2, so it is not an acceptable T-mesh. But we can make some refinements such as those in the right picture, and turn it into an acceptable T-mesh. In the following, we assume that all the T-meshes are acceptable T-meshes.

The construction mainly contains two steps.

Step 1: Define a bi-cubic map for each face

The bi-cubic Bézier form for ϕ_i which corresponds to the face without extraordinary vertices is very easy to compute since we have known the local geometric information for each corner and the basis functions for each regular vertex. However, we should have different rules for the maps corresponding to the extraordinary faces because we do not have mixed partial derivatives. We will illustrate the equations for the initial bi-cubic maps according to Fig. 7. In the figure, V is a valence n vertex with n neighboring vertices V_i with the valence n_i. The knot interval for edge VV_i is respectively a_i.

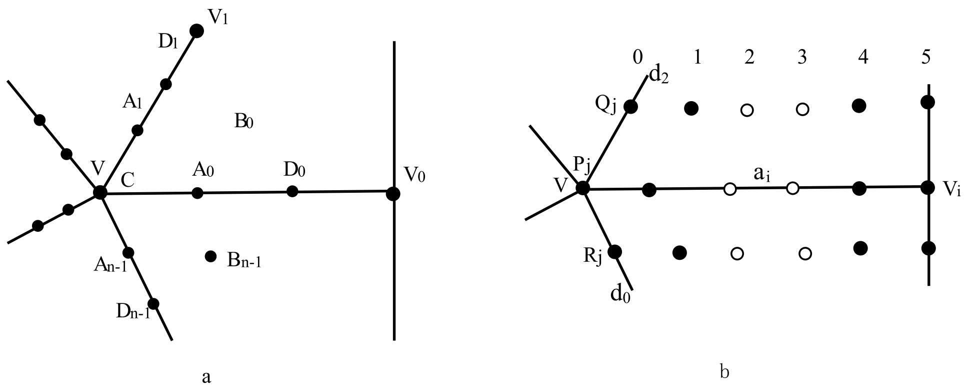

Fig. 7 The notations for G^1 smoothness of a map corresponding to the extraordinary face

According to the geometric information at the vertices, we can determine the Bézier control points C, A_i and D_i. The only undermined Bézier control points for the initial bi-cubic maps are those associated with the twist vector at the extraordinary vertices such as B_i in Fig. 7.

$$\text{Denote } b_i(t) = \begin{cases} 2\cos\frac{2\pi}{n}(1-t) - 2\cos\frac{2\pi}{n_i}t, & \text{if } n_i \neq 4; \\ 2\cos\frac{2\pi}{n}(1-t)^2, & \text{else;} \end{cases}$$

and

$$U_i = b_i^{'}(0)\frac{A_i - C}{a_i} + b_i(0)\frac{D_i - 2A_i + C}{a_i^2}.$$

Here $b_i^{'}(t)$ is the first derivative of $b_i(t)$. Then we can compute B_i by the following equation:

$$B_i = A_i + A_{i+1} - C + a_i a_{i+1}\left(\frac{U_i}{4} + \frac{U_{i+1}}{4}\right). \tag{5}$$

We have computed all the bi-cubic Bézier control points for the maps which interpolate all the geometric information at the basis vertices. However, the maps corresponding to the extraordinary faces are not in tangent continuity. So the next step is to smooth the maps to be in tangent continuity by perturbating the next two rows of Bézier control points.

Step 2: G^1 smoothness for the extraordinary vertex
Suppose the next two rows of Bézier control points for the maps adjacent to edge VV_i are labeled as in Fig. 7b. The black control points are determined by the local geometric information which will be kept unchanged. The main process of G^1 smoothness is listed below.

1. Compute P_2 and P_3 according to the following equations:

$$P_2 = 2P_1 - C + \frac{a_i^2}{b_i(0)}\left(\frac{U_{i-1} + 2U_i + U_{i+1}}{4} - \frac{b_i'(0)(P_1 - P_0)}{a_i}\right);$$

if $n_i = 4$,

$$P_3 = P_2 + \frac{P_4 - P_1}{2} - \frac{P_5 - P_0}{10}.$$

Otherwise, P_3 is computed by the similar equations, as P_2.

2. Denote $P(t)$, $Q(t)$ and $R(t)$ three quintic Bézier curves with control points P_i, Q_i and R_i, respectively. Let $V(t)$ be a cubic Bézier curve with control points $Q_0 - R_0$, $\frac{5(Q_1-R_1)}{3} - \frac{2(Q_0-R_0)}{3}$, $\frac{5(Q_4-R_4)}{3} - \frac{2(Q_5-R_5)}{3}$ and $Q_5 - R_5$. Then $Q(t)$ and $R(t)$ be linear combination of $P(t)$ and $V(t)$:

$$Q(t) = P(t) + \frac{a_{i+1}a_{i-1}b_i(t)P^{'}(t)}{5(a_{i+1} + a_{i-1})a_i} + \frac{a_{i+1}}{a_{i+1} + a_{i-1}}V(t),$$

$$R(t) = P(t) + \frac{a_{i+1}a_{i-1}b_i(t)P^{'}(t)}{5(a_{i+1} + a_{i-1})a_i} - \frac{a_{i-1}}{a_{i+1} + a_{i-1}}V(t).$$

The two maps are G^1 because

$$\frac{Q(t) - P(t)}{a_{i+1}} + \frac{R(t) - P(t)}{a_{i-1}} = \frac{b_i(t)P^{'}(t)}{5a_i}. \tag{6}$$

Why we use bi-quintic maps instead of bi-cubic maps [10] is for the following two reasons. The first is that we cannot control the geometric information at extraordinary vertices if we use bi-cubic. And the second is that the bi-cubic maps will give very bad result for higher-order saddle points of an even valence (for example, the valence 6 extraordinary vertex in Fig. 10b). However, it looks pretty good for higher-order saddle points of an even valence if to use the bi-quintic maps.

4.2 Spline surface from quadrilateral mesh

A direct application of GPT-spline is to construct a smooth spline surface from a given 2-manifold mesh in $\mathbb{R}^3$ with quadrilateral faces of arbitrary topological genus. This can be achieved by interpolating the local geometric information at the basis vertices, which can be estimated from the mesh.

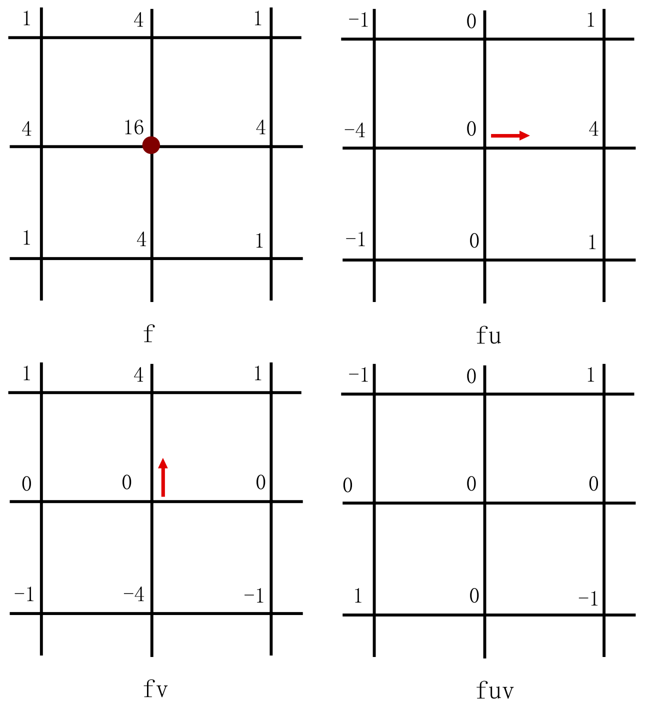

Fig. 8 The mask for local geometry estimator at regular vertex

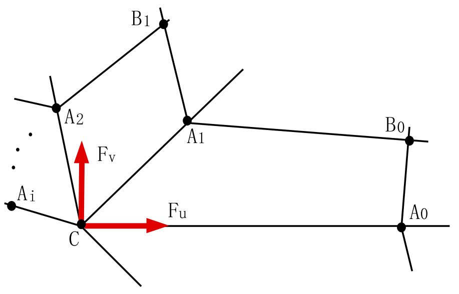

Fig. 9 An extraordinary vertex

For a regular vertex, we can estimate the local geometric information by traditional B-spline theory. The masks are illustrated in Fig. 8. For a valence n, extraordinary vertex C which is surrounded by A_i and B_i is illustrated in Fig. 9. The position F of the surface at C is defined as the limit position of Catmull–Clark subdivision

$$F = \frac{n}{n+5}C + \frac{4}{n(n+5)}\sum_{i=0}^{n-1} A_i + \frac{1}{n(n+5)}\sum_{i=0}^{n-1} B_i. \qquad (7)$$

F_u and F_v are defined to make the normal of the surface to be the same as that of Catmull–Clark subdivision:

$$F_u = \frac{2}{n(2+\omega_n)}\sum_{i=0}^{n-1}\left(\omega_n \cos\left(\frac{2i\pi}{n}\right)A_i + \left(\cos\left(\frac{2i\pi}{n}\right) + \cos\left(\frac{2(i+1)\pi}{n}\right)\right)B_i\right),$$

$$F_v = \frac{2}{n(2+\omega_n)}\sum_{i=0}^{n-1}\left(\omega_n \sin\left(\frac{2i\pi}{n}\right)A_i + \left(\sin\left(\frac{2(i+1)\pi}{n}\right) + \sin\left(\frac{2i\pi}{n}\right)\right)B_i\right).$$

Here $\omega_n = 1 + \cos(\frac{2\pi}{n}) + \cos(\frac{\pi}{n}\sqrt{2(9+\cos(\frac{2\pi}{n}))})$.

Figure 10 illustrates several examples for constructing spline over T-mesh form the given quadrilateral meshes.

5 B-spline surface simplification

Surface fitting with tensor-product B-spline surfaces might have many superfluous control points or surface patches. This section presents an algorithm for converting a tensor-product B-spline surface into a PHT-spline surface within a given tolerance according to the edges removal operator.

Cardon [1], Deng et al. [3] and Sederberg et al. [12] discussed the B-spline surface simplification with T-splines and PHT-spline using iterative refinement. However, the iterative refinement method has the following disadvantages. First, it is dependent on the mesh topology. So it is not a trivial task to extend the method for arbitrary topology surface. Second, it is not general to simplify a rational B-spline surface. Third, it is not easy to simplify the given region of the given surface. In this section we will discuss the surface simplification algorithm based on iterative edges removal algorithm.

Simplification algorithm involves two non-trivial tasks: how to estimate the error and how to select the candidate edge removal. The solution presented here does not produce the result with fewest surface patches that fall bellow the given tolerance. Since in general the models may have several thousands of surface patches, it is intractable to make an exhaustive search. However, the surface will exactly fall under the given tolerance and it works on rational surface and arbitrary topology mesh.

The error estimator for edge removal is based on the difference between the control points of the surface before and after edge removal. Suppose a spline surface $\mathbf{S}(u,v)$ will become $\hat{\mathbf{S}}(u,v)$ with the Nth vertex being deleted, and the basis functions for the jth basis vertex are $\hat{b}_j^k(u,v)$. As $\hat{\mathbf{S}}(u,v) = \sum_{j=1}^{N-1}\sum_{k=0}^{3}\mathbf{C}_j^k\hat{b}_j^k(u,v)$, $\hat{\mathbf{S}}(u,v)$ can be represented by basis functions $\{b_j^k(u,v)\}$, $j = 0,\dots,N$, as well. Suppose it has the form

$$\hat{\mathbf{S}}(u,v) = \sum_{j=1}^{N-1}\sum_{k=0}^{3}\mathbf{C}_j^k b_j^k(u,v) + \mathbf{D}_N^k b_N^k(u,v).$$

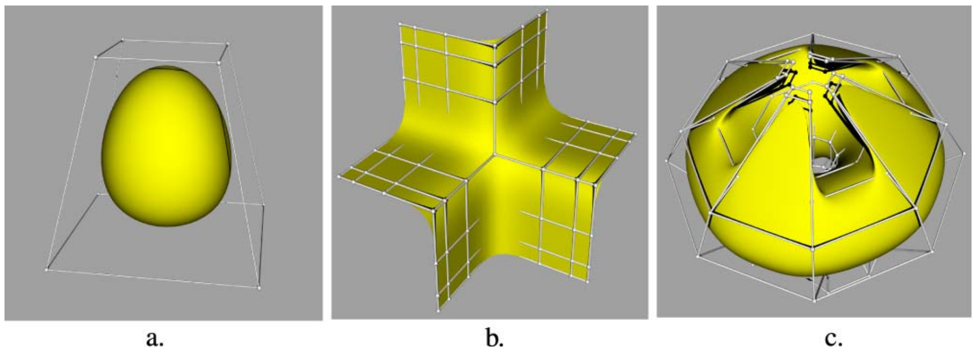

Fig. 10 Several GPT-spline surfaces from quadrilateral meshes

So the difference between the surfaces is

$$\begin{aligned}\left|\mathbf{S}(u,v)-\hat{\mathbf{S}}(u,v)\right| &= \sum_{k=0}^{3}\left\|\mathbf{C}_N^k-\mathbf{D}_N^k\right\| b_N^k(u,v)\\ &\le \max_{0\le k\le 3}\left\|\mathbf{C}_N^k-\mathbf{D}_N^k\right\|.\end{aligned}$$

This equation provides the approximation error estimation between the original surface and the approximation surface after the edge removal. In fact, the above equation demonstrates the smoothness between the polynomial patches on the two adjacent cells of the removed edge. Since the surface is polynomial in each cell, an edge can be removed if and only if the surface is C^3 or almost C^3 along the edge.

In considering the task for selecting the candidate edges removal, we focus on two desirable properties for the simplification T-meshes. First, the resultant mesh should have a uniform structure that is more intuitive to human users. Second, it will produce the simplest structure T-mesh if the given tolerance is big enough. To meet these goals, we perform the following steps.

In the simplification algorithm, we have two basis operators, *poly-edge removal* and *vertex removal*. A poly-edge is a set of edges which satisfy the following properties. Suppose an edge belongs to a poly-edge with two end points A and B; if A or B is a valence 4 vertex, then the opposite edge for vertex A or B also belongs to the poly-edge. For example, in the mesh of Fig. 11, the yellow, red and the dashed lines are all poly-edges. A poly-edge removal operator is to check the whole poly-edge and remove all the possible removable edges from the poly-edge.

Since poly-edges in a mesh are very complicated, we provide a local simplification operator, vertex removal operator. Vertex removal is a local operator to check the four edges which contain the vertex and remove the removable edges. This operator will modify the maps associated with the four neighboring faces.

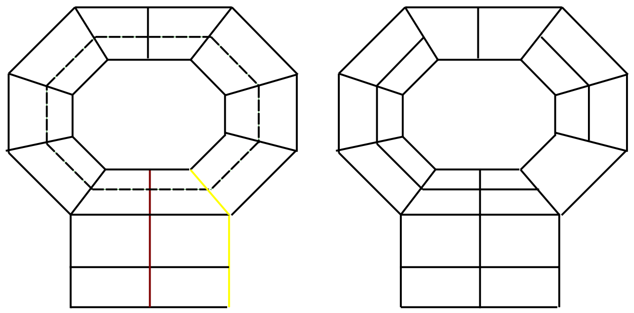

Fig. 11 The poly-edge for a mesh. The *yellow*, *red* and the *dashed lines* are all poly-edges for the mesh, and the *right picture* is a possible removal result for the dashed poly-edge

Table 1 Simplification data

Fig.	Original No. (Patch)	Error (1.0%) No. (Patch)	Error (1.4%) No. (Patch)
12.1	8280	713	387
12.3	648	49	35

B-spline surface simplification algorithm

1. Represent the B-spline surface with the polynomial spline surface over T-mesh according to the interpolation theorem. Set all the edges unchecked.
2. Remove poly-edges under the current tolerance one by one.
3. Remove unchecked cross-vertices.

We present some examples to illustrate the effect of the algorithm. Each of these examples can be computed within twenty seconds on a personal computer with Pentium 4 CPU 3.20 GHz and 1.0 GB RAM. Table 1 give the numbers of the bi-cubic patches for different errors. We also provide the T-mesh structure of simplification result of the first example in Fig. 13.

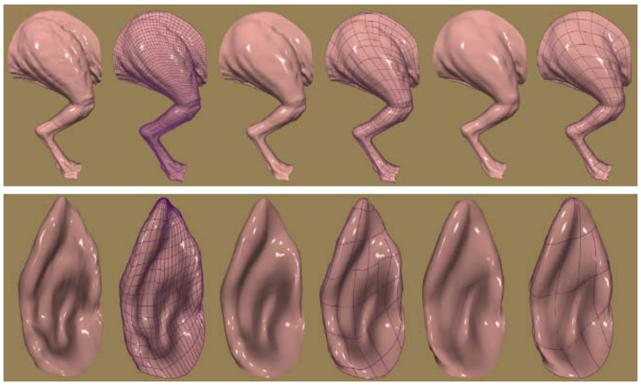
Fig. 12 Several examples of B-spline surface simplification

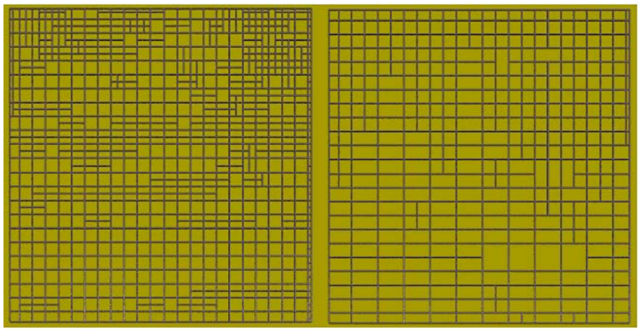
Fig. 13 The mesh structure of the simplification of Fig. 12 example 1

6 Conclusions and future work

The paper generalizes the PHT-splines to arbitrary topology over general T-meshes with any structure. GPT-spline surfaces can be constructed through a unified scheme to interpolate the local geometric information at the basis vertices. We also discuss the general edge insertion and removal algorithms for GPT-splines. As applications, we present algorithms to construct a GPT-spline surface from a quadrilateral mesh and simplify a tensor-product B-spline surface into a GPT-spline surface by edges removal algorithm. The method is simple and straightforward for implementation. It is evident that all the processes can be generalized to GPT-spline of higher degrees.

There exist a number of problems for future research. For example, how to compute the dimension of the spline space $\mathcal{S}(3,3,2,2,\mathcal{T})$ over general T-meshes or hierarchical T-meshes $\mathcal{T}$, and how to construct the basis functions for the spline space? And what is the relationship between T-splines and GPT-splines?

Acknowledgements The authors are supported by the National Key Basic Research Project of China (No. 2004CB318000), the NSF of China (Nos. 60533060, 10671192, 10701069, 60903148), One Hundred Talent Project supported by CAS, Program for New Century Excellent Talents in University, Chinese Academy of Science (Startup Scientific Research Foundation), the Specialized Research Fund for the Doctoral Program of Higher Education (No. 20060358055), and the 111 Project (No. b07033).

References

1. Cardon, D.L.: T-spline simplification. Master thesis, Brigham Young University, Provo, UT, USA (2007)
2. Deng, J., Chen, F., Feng, Y.: Dimensions of spline spaces over T-meshes. J. Comput. Appl. Math. **194**, 267–283 (2006)
3. Deng, J., Chen, F., Li, X., Hu, C., Tong, W., Yang, Z., Feng, Y.: Polynomial splines over hierarchical T-meshes. Graph. Models **74**, 76–86 (2008)
4. Dierckx, P.: On calculating normalized Powell–Sabin B-splines. Comput. Aided Geom. Des. **15**, 61–78 (1998)
5. Farin, G.: Curves and Surfaces for CAGD—A Practical Guide. Morgan Kaufmann, San Mateo (2002)
6. Forsey, D., Bartels, R.H.: Hierarchical B-spline refinement. Comput. Graph. **22**, 205–212 (1988)
7. Forsey, D., Wong, D.: Multiresolution surface reconstruction for hierarchical B-splines. Graph. Interf. **2**, 57–64 (1998)
8. Gonzalez-Ochoa, C., Peter, J.: Localized hierarchy surface splines. In: Proceedings of the 1999 Symposium on Interactive 3D graphics, vol. 2, pp. 7–15 (1999)
9. Kraft, R.: Adaptive and linearly independent multilevel b-splines. Surf. Fitt. Multiresolut. Methods **2**, 209–218 (1997)
10. Li, X., Deng, J., Chen, F.: Surface modeling with polynomial splines over hierarchical T-meshes. Vis. Comput. **23**, 1027–1033 (2007)
11. Sederberg, T.W., Zheng, J., Bakenov, A., Nasri, A.: T-splines and T-NURCCs. ACM Trans. Graph. **22**, 161–172 (2003)
12. Sederberg, T.W., Cardon, D.L., Finnigan, G.T., North, N.S., Zheng, J., Lyche, T.: T-spline simplification and local refinement. ACM Trans. Graph. **23**, 276–283 (2004)

Xin Li was born in Hubei, China, in 1980. He received his Ph.D. degree from the University of Science and Technology of China. Currently he is a associate professor in the Department of Mathematics at the University of Science and Technology of China. His research interests include Computer Aided Geometric Design and Computer Graphics.

Jiansong Deng was born in Shangdong, China, in 1971. He received his Ph.D. degree from the University of Science and Technology of China in 1998. Currently he is a professor in the Department of Mathematics at the University of Science and Technology of China. His research interests include Computer Aided Geometric Design and Computer Graphics.

Falai Chen was born in Anhui, China, in 1966. He received his Ph.D. degree from the University of Science and Technology of China in 1994. He is a professor in the Department of Mathematics at the University of Science and Technology of China. His research interests include Computer Aided Geometric Design and Computer Graphics.