Hindawi
Complexity
Volume 2020, Article ID 7103963, 12 pages
https://doi.org/10.1155/2020/7103963

Research Article

A Polynomial Splines Identification Method Based on Control Nets

Zhihua Wang[1,2] **and Hongmei Kang**[3]

[1]*School of Mathematical Sciences, University of Science and Technology of China, Hefei 230026, China*
[2]*School of Mathematics and Computer Science, Anqing Normal University, Anqing 246011, China*
[3]*School of Mathematical Sciences, Soochow University, No. 1 Road Shizi, Suzhou, Jiangsu, China*

Correspondence should be addressed to Zhihua Wang; 1208044765@qq.com

Received 22 April 2020; Revised 29 June 2020; Accepted 13 July 2020; Published 19 August 2020

Academic Editor: Oh-Min Kwon

In this study, based on Polynomial Splines with control nets, an identification method is investigated. We introduce polynomial splines with control nets defined over T-mesh. The basic idea is to extend T-vertices such that those T-vertices become interior cross vertices or boundary vertices. To this end, we introduce the design-suitable T-mesh for constructing polynomial splines with control net. In design-suitable T-meshes, there are no extra basis vertices produced by an appropriate extension of T-vertices. The basis functions are defined over each vertex in a design-suitable T-mesh by the means of constructing PHT-splines basis functions.

1. Introduction

T-splines [1, 2] are introduced to overcome the weakness of NURBS by allowing T-junctions. T-splines are considered as a generalization of NURBS surfaces and capable of local refinement, which makes T-splines a powerful modeling tool for advanced geometric modeling and adaptive isogeometric analysis (IGA) [3, 4]. Stimulated by the advent of T-splines and isogeometric analysis, locally refinable splines are born and now becoming flourishing. Currently, there are Hierarchical B-splines [5–7], truncated hierarchical B-splines (THB-splines) [8], truncated T-splines [9], truncated hierarchical tricubic C^0 spline [10], blended B-spline based on unstructured quadrilateral and hexahedral meshes [11], analysis-suitable T-splines (AST-splines) [12], LR B-splines [13], modified T-splines [14], and polynomial splines over hierarchical T-meshes (PHT-splines) [15].

PHT-splines are introduced in [15] as bicubic C^1 continuous polynomial splines spaces defined over hierarchical T-meshes. PHT-splines possess a set of nonnegative and linearly independent basis functions and allow for very efficient local refinement. PHT-splines have been widely applied in geometric processing and analysis. The finite element discretization of elliptic equations based on PHT-splines was discussed in [16], where numerical solutions are refined adaptively and have the optimal convergence rate. PHT-splines are also favored in isogeometric analysis for solving elastic problems [3, 17, 18] and adaptive isogeometrican analysis [19]. PHT-splines are also applied in reconstructing surface models efficiently. PHT-splines were used in stitching several surface patches to construct complex models in paper [20] and PHT-splines were also applied in surface reconstruction from a very large set of point clouds in implicit form [21]. IGA collocation approaches are introduced in paper [22], and PHT-splines have been used as basis functions for the adaptive collocation method [23].

In order to make PHT-splines better suited for analysis and geometric processing, some improvements and extensions have been made to PHT splines in recent years. In [24], the authors discussed the decay phenomenon of PHT-splines basis functions. They found some of the basis functions will tend to zero as the refinement level increases under certain types of refinement. Such a decay makes the stiffness matrix in isogeometric analysis be ill-conditioned, which is not expected by analysis. Thus a new basis

construction method is proposed for PHT-splines to avoid the decay. The extension of PHT-splines to general T-meshes is considered in [25], where the bicubic C^1 continuous polynomial splines are defined over general T-meshes. The basis functions are constructed by computing the geometric information at basis vertices such that the resulting basis functions are nonnegative and linearly independent and form a partition of unity. PHT splines were defined over hierarchical T-meshes without irregular vertex in [15], and then the bicubic C^1 continuous splines construction based on irregular quad layout were discussed in [26]. Current implementation of PHT-splines stores the basis functions in Bézier forms, which saves some computational costs but consumes a lot of memories. In [27], an algorithm to evaluate PHT-splines is provided where only the information about the control coefficients and the hierarchical mesh structure is given. The evaluation algorithm takes about the same computational costs while requiring much less amount of memory compared with the Bézier representations.

For PHT-splines, there is no one-to-one correspondence between the vertices in the underlying T-mesh and PHT-splines basis functions. The boundary vertices and interior cross vertices are called basis vertices. Each basis vertex is associated with four basis functions. The control net concept is essential in computer aided geometric design (CAGD), because the control net makes editing models easily and intuitively. In this paper, we introduce polynomial splines with control polygon to complement PHT-splines. The proposed splines defined over all the vertices of the underlying T-meshes, such that there is a one-to-one correspondence between control points and vertices in T-meshes. The basis idea is to extend T-vertices to be basis vertices. In order to avoid generating extra basis vertices when extending T-vertices, we introduce a subset of T-meshes, called design-suitable T-meshes. Over design-suitable T-meshes, the T-vertices can be extended appropriately without generating extra basis vertices. For a given T-mesh, it needs to connect some T-vertices to be a design-suitable T-mesh. We define polynomial splines basis functions over the design-suitable T-mesh. The resulting splines not only inherit the nice properties of PHT-splines, but also have control nets.

The paper is organized as follows. In Section 2, we review some preliminary knowledge about polynomial splines over T-meshes. In Section 3, we proposed a subset of T-mesh called design-suitable T-mesh and prove that there are no extra basis vertices produced by an appropriate extension. In Section 4, we present an algorithm of local refinement of the proposed splines. Finally, we give a conclusion of this paper in Section 5.

2. Preliminary Knowledge

2.1. T-Meshes. A T-mesh is a rectangular grid with T-junctions. The cells in a T-mesh must be rectangles. A grid point in a T-mesh is called a *vertex*. And the vertex on a boundary grid line is called a *boundary vertex*; otherwise, it is called an *interior vertex*. Interior vertices consist of cross vertices and T-vertices. We adopt the notations $\top$, $\perp$, $\vdash$, and $\dashv$ to indicate the four possible orientations of T-vertices. The T-vertices of type $\vdash$ and $\dashv$ are called horizontal T-vertices and the T-vertices of type $\top$ and $\perp$ are called vertical T-vertices.

2.2. Bicubic C^1 Continuous B-Splines. Let

$$t_0, t_0 = t_1, t_1 < t_2, t_2 < \cdots < t_j, t_j < \cdots < t_{n-1}, t_{n-1} = t_n, t_n, \quad (1)$$

be a knot vector with interior knots of multiplicity two. Then each knot t_j is associated with two cubic B-splines $N_j^1(t) = N^3[t_{j-1}, t_{j-1}, t_j, t_j, t_{j+1}](t)$ and $N_j^2(t) = N^3[t_{j-1}, t_j, t_j, t_{j+1}, t_{j+1}](t)$. Both $N_j^1(t)$ and $N_j^2(t)$ together with their derivatives have the same support $[t_{j-1}, t_{j+1}]$. Except for $N_j^1(t)$ and $N_j^2(t)$, all the other cubic B-splines and their derivatives vanish at t_j.

For given a tensor product mesh $\mathcal{T}$, the two associated global knot vectors in s-direction and t-direction are

$$U = \{s_0, s_0 = s_1, s_1 < s_2, s_2 < \cdots < s_j, s_j < \cdots < s_{m-1}, s_{m-1} = s_m, s_m\},$$
$$V = \{t_0, t_0 = t_1, t_1 < t_2, t_2 < \cdots < t_j, t_j < \cdots < t_{n-1}, t_{n-1} = t_n, t_n\}, \quad (2)$$

respectively. The bicubic C^1 continuous B-splines surface defined over $\mathcal{T}$ is spanned by the $\{N_{i,j}^k\}_{i=0,j=0,k=0}^{i=m,j=n,k=3}$, where $N_{i,j}^k(s,t) = N_i^{k\%2}(s) N_j^{[k/2]}(t)$, $k = 0, 1, 2, 3$ are the four B-splines associated with the vertex (s_i, t_j).

For a function $f(x, y)$, the function value, the first partial derivatives, and the mixed partial derivative are called the geometric information of $f(x, y)$, denoted by

$$\mathcal{G}f(x,y) = \left(f(x,y), f_x(x,y), f_y(x,y), f_{xy}(x,y)\right). \quad (3)$$

For the basis function $N_{i,j}^k$ defined above, it has $\mathcal{G}N_{i,j}^k(s_p, t_q) = \delta_{ip}\delta_{jq}$ and $\delta_{ip} = 1$, if $i = p$; otherwise, $\delta_{ip} = 0$. That is the geometric information of $N_{i,j}^k$ vanishes at other vertices in $\mathcal{T}$ except its associated vertex (s_i, t_j).

3. Polynomial Splines with Control Nets Defined over T-Meshes

PHT-splines span the polynomial splines space $S(3, 3, 1, 1)$ over a T-mesh and are defined over cross interior vertices and boundary vertices (called basis vertices), resulting in PHT-splines lacking control net. Our aim is to equip PHT-splines with control net. For any given control mesh, we parameterize the control mesh into a T-mesh in 2D domain and then extend all T-vertices in the T-mesh such that they become basis vertices and no extra basis vertices is introduced, and finally construct basis functions of $S(3, 3, 1, 1)$ over the extended T-mesh. Thus, there is a one-to-one correspondence between the control mesh and parameter T-mesh. For arbitrary T-mesh, it is inevitable that extra basis vertices are produced by directly extending T-vertices. Therefore, in order to avoid extra basis vertices, we introduce design-suitable T-meshes for defining basis functions.

3.1. Design-Suitable T-Meshes. For a given T-mesh $\mathcal{T}$, we shall adopt some definitions and notations (Figure 1).

(i) *Extension of T-vertex*. The closed line segment created by extending a T-vertex in the missing direction until it intersects with an edge is called the extension of this T-vertex. Particularly, if the edge is an edge in $\mathcal{T}$, then the extension is called a full-extension. The extension of a horizontal (vertical) T-vertex is called the horizontal (vertical) extension, respectively. In Figure 1(b), the dotted line segments are full-extensions of T-vertices.

(ii) *Full-Extension Mesh*. A T-mesh together with the full-extensions of all T-vertices in this T-mesh forms the full-extension mesh. Figure 1(b) shows the full-extension mesh of the T-mesh in Figure 1(a).

(iii) *Edge-Type Intersections*. The intersection between two same type extensions and the intersection between an extension and a boundary edge is called an edge-type intersection (E-intersection for short). In Figure 1(b), $v_0, v_1, \ldots, v_7$ are all E-intersections.

(iv) *Face-Type Intersections*. The intersection between a horizontal extension and a vertical extension is called a face-type intersection (F-intersection for short). In Figure 1(b), $v_8, v_9, \ldots, v_{13}$ are all F-intersections.

(v) *Connectable Vertex*. Two vertices can be connected if the line segment formed by these two vertices splits the lying face into two subrectangles. If two T-vertices can be connected, then they are called connectable T-vertices (CT-vertex for short). The vertices are marked by blue solid circles in Figure 1(a).

(vi) *E-Connectable T-Vertex*. For a T-vertex, if its extension intersects with the extension of another same type T-vertex or with a boundary edge, then the T-vertex is called an e-connectable T-vertex (ECT-vertex). The vertices marked by yellow solid circles in Figure 1(a) are ECT-vertices.

(vii) *Normal T-Vertex*. For a T-vertex, if it is either a CT-vertex or an ECT-vertex, then it is called a normal T-vertex.

(viii) *T-Element*. If there are T-vertices lying on the edges of an element, then the element is called a T-element.

(ix) *C-Element*. For a T-element in $\mathcal{T}$, if there are ECT-vertices on the edges of this element, then the element is called a C-element. Those elements $F_1, F_2, \ldots, F_6$ in Figure 1(a) are connectable elements.

(x) *S-Element*. For a C-element, if there exists a normal vertical (horizontal) T-vertex in the element such that its extension subdivides the element into two subelements and one subelement contains all the horizontal (vertical) ECT-vertices, then the connectable element is called a S-element and the normal vertical (horizontal) T-vertex is called a ST-vertex. The element F_6 is a S-element, while $F_1, \ldots, F_5$ are not S-elements.

Definition 1. For a given T-mesh $\mathcal{T}$, the corresponding full-extension T-mesh is denoted by $\mathcal{T}_{ext}$, if

(i) there are no connectable T-vertices in $\mathcal{T}$

(ii) for each edge-type intersection in $\mathcal{T}_{ext}$, the underlying edge is attached to a S-element;

then $\mathcal{T}$ is called a design-suitable T-mesh.

The diagonal T-mesh (the diagonal elements are refined in the diagonal direction) shown in Figure 2(a) is a design-suitable T-mesh. The T-mesh shown in Figure 1(b) is not a design-suitable T-mesh. But if we connect all the CT-vertices and ECT-vertices, then the resulting mesh is a design-suitable T-mesh which is shown in Figure 3(a).

For a design-suitable T-mesh, there are no extra basis vertices produced by extending T-vertices appropriately. The intuitive way is to extend a T-vertex in the missing direction, until it intersects with an edge in the underlying mesh instead of the original T-mesh. We call such a way of extending T-vertices as p-extending to distinguish from the extension mentioned above. For the design-suitable T-mesh shown in Figure 2(a), the new T-mesh produced by p-extending T-vertices is shown in Figure 2(b). We also plot the full-extension T-mesh in Figure 2(c) for comparison. This p-extending way is not enough for the T-meshes containing ECT-vertices. It still requires an order of extending T-vertices. Thus, we introduce the following algorithm to extend T-vertices in a given T-mesh, and the resulting T-mesh is denoted by $\mathcal{T}_e$.

(i) Step one: p-extend T-vertices in S-elements and the adjacent elements of S-elements

  (a) S-element: First p-extend the ST-vertices of an S-element; then p-extend the rest of T-vertices in the element

  (b) Adjacent elements: First p-extend ECT-vertices; then p-extend the normal T-vertices

(ii) Step two: p-extend T-vertices in the rest of T-elements

  (a) First p-extend horizontal T-vertices; then p-extend vertical T-vertices

We take Figure 3 as an example to explain the above algorithm. In the T-mesh shown in Figure 3(a), $F_1, F_2, \ldots, F_6$ are T-elements. We need to extend the T-vertices in these elements. The element F_3 is an S-element. We start from extending T-vertices in this element. The vertex v_5 is a ST-vertex, and it is extended until it intersects with an existing edge. Then we extend the vertex v_4, and the first edge it intersects with is the extension of v_5. The element F_6 is the adjacent element of F_3. The vertex v_8 is p-extended. For the rest T-elements, if there are horizontal T-vertices, then extend them first. Thus, in F_3, v_3 is extended first and

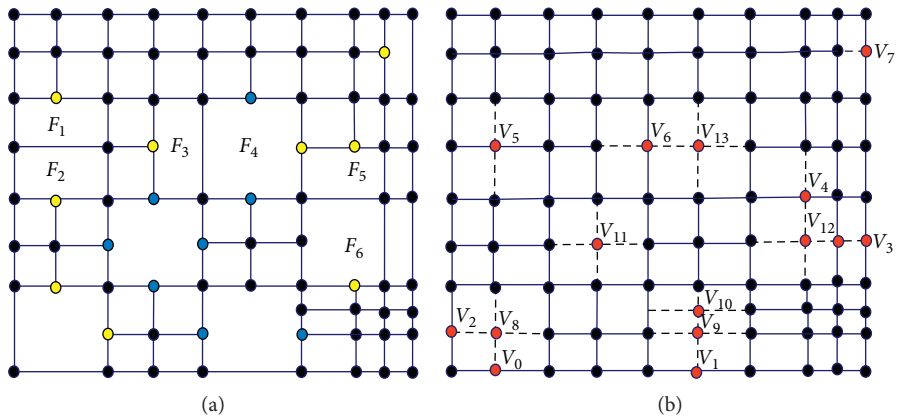

Figure 1: Full-extension T-mesh. (a) T-mesh; (b) full-extension T-mesh $\mathcal{T}_{ext}$.

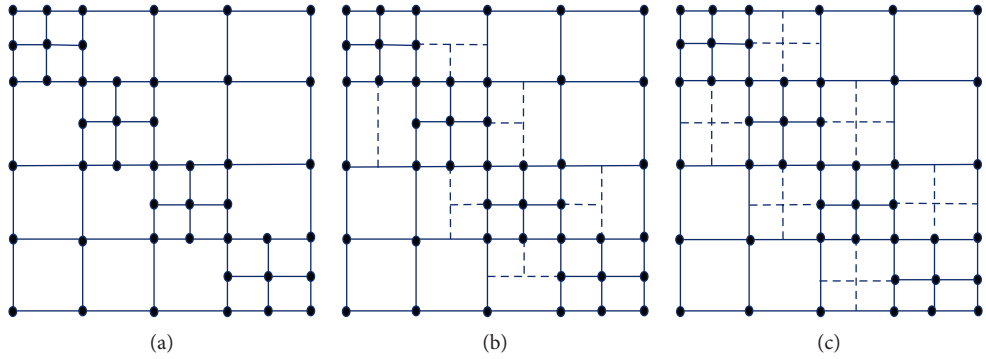

Figure 2: A design-suitable T-mesh and the p-extending. (a) Diagonal T-mesh, (b) p-extending, and (c) full-extension T-mesh.

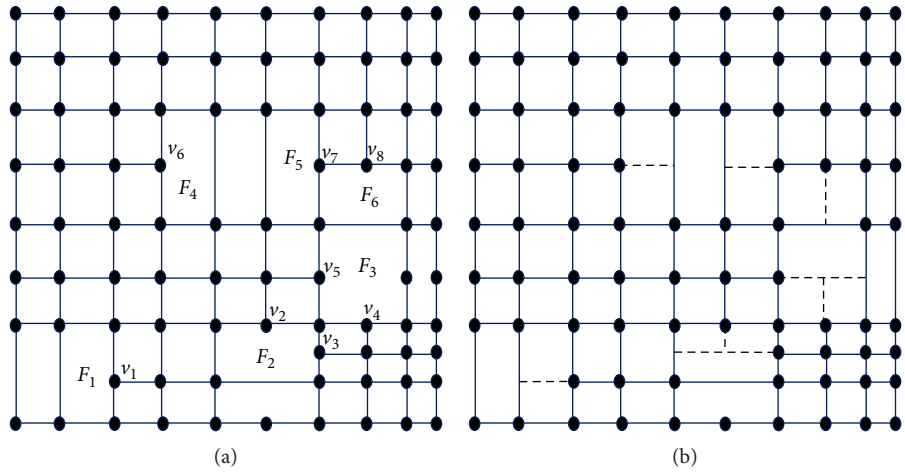

Figure 3: The proposed algorithm for extending T-vertices. (a) Design-suitable T-mesh; (b) extended T-mesh $\mathcal{T}_e$.

then v_2 is extended. The resulting extended T-mesh is shown Figure 3(b).

Theorem 1. *For a design-suitable T-mesh $\mathcal{T}$, extend the T-vertices in $\mathcal{T}$ as above. Then there is a one-to-one correspondence between basis vertices in $\mathcal{T}_e$ and vertices in $\mathcal{T}$, which means there are no extra basis vertices produced in $\mathcal{T}_e$ by extending T-vertices in $\mathcal{T}$ appropriately.*

Proof. We only need to deal with T-elements. We extend T-vertices element by element. We need to prove there are no edge-type intersections and face-type intersections in $\mathcal{T}_e$. In a

design-suitable T-mesh, only extending ECT-vertices may produce edge-type intersections. Furthermore, there are no face-type intersections produced by p-extending. S-elements are extended first according to the above algorithm. Since ST-vertices are normal T-vertices, then their extensions intersect with existing edges and the intersections are T-vertices. According to the definition of ST-vertices, the extensions of ST-vertices prevent the intersection between ECT-vertices. Then extending T-vertices in the adjacent T-elements of S-elements will not produce new basis vertices. For the rest T-elements, there are no basis vertices produced by p-extending. □

3.2. Construct Basis Functions over $\mathcal{T}_e$. Let $\mathbf{B}_i(s,t) = (B_i^1(s,t), B_i^2(s,t), B_i^3(s,t), B_i^4(s,t))$ be the basis functions associated with V_i needed to be constructed. There are mainly three steps for constructing $\mathbf{B}_i(s,t)$ in [25]:

(i) Find a rectangle containing all the vertices at which $\mathbf{B}_i(s,t)$ which do not vanish in $\mathcal{T}_e$

(ii) Set the geometric information of bicubic C^1 continuous B-splines associated with V_i over the rectangle as the geometric information of $\mathbf{B}_i(s,t)$

(iii) Set the geometric information at other basis vertices in the rectangle as zeros and represent $\mathbf{B}_i(s,t)$ in the Bézier form or B-splines form

The method for finding a rectangle for each basis vertex in $\mathcal{T}_e$ is stated as follows. For a basis vertex $v_i = (s_{i_0}, t_{i_0})$ in $\mathcal{T}_e$, first put v_i into K_i, then check the neighboring vertices of the vertices in K_i and put the T-vertices into K_i recursively until there is no vertex added into K_i. The minimal rectangle containing all the vertices in K_i is denoted by $R_V = [s_0, s_1] \times [t_0, t_1]$, then R_V together with the knot lines $s = s_{i_0}$ and $t = t_{i_0}$ forms a 2×2 tensor product mesh $M_V = \{s_0, s_{i_0}, s_1\} \times \{t_0, t_{i_0}, t_1\}$. For the sake of clearness, we call M_v as the support mesh of basis vertex v_i. Figure 4 shows the support meshes of four basis vertices v_{13}, v_{17}, and v_4 in the extended T-mesh, where the support meshes are shaded. In Figure 4(a), v_{13} is a basis vertex and among the neighbors of v_{13}, v_{48} is a T-vertex; thus, it is put into K_i. Then we check the neighbors of v_{48}, and v_{49} is a T-vertex which is put into K_i. For v_{49}, there are no more vertices needed to be added into K_i. Thus, the set $K_i = \{v_{13}, v_{48}, v_{49}\}$. Similarly for v_{17} in Figure 4(b), now $k_i = \{v_{17}, v_{50}, v_{51}\}$. For the basis vertex v_4 shown in Figure 4(c), $K_i = \{v_4, v_{53}\}$.

The four B-spline basis functions defined over M_v are denoted by $N_k(s,t)$, $k = 0, 1, 2, 3$, and

$$N_k(s,t) = N^3\left[s_0, s_0, s_{i_0}, s_{i_0}, s_1, s_1\right](s) \times N^3\left[t_0, t_0, t_{i_0}, t_{i_0}, t_1, t_1\right](t). \tag{4}$$

Then the geometric information of four basis functions $\mathbf{B}_i(s,t)$ at V_i is defined as the geometric information of $N_k(s,t), k = 0, 1, 2, 3$. The geometric information of $\mathbf{B}_i(s,t)$ at the other basis vertices in M_v is set as zero. And the geometric information of $\mathbf{B}_i(s,t)$ at T-vertices in M_v is computed by C^1 constraints. According to the known geometric information of $\mathbf{B}_i(s,t)$ at the vertices in M_v, then $\mathbf{B}_i(s,t)$ can be easily represented in Bézier form or B-splines form.

3.3. Polynomial Splines with Control Net Defined on T-Meshes. With the help of index T-mesh, there is a one-to-one correspondence between vertices and basis functions. The index T-mesh is adopted in [3] for ease of constructing T-splines. It is constructed by plotting the knots at equally spaced intervals regardless of their actual spacing and labeling each knot line with integer values [3]. Figure 5(b) shows the index T-mesh of the mesh shown in Figure 5(a). Figure 5(c) shows a control net in $\mathbb{R}^3$ corresponding to the T-mesh shown in Figure 5(a). Figure 5(d) shows the polynomial splines surface with control net Figure 5(c) defined on the design-suitable T-mesh Figure 5(a). Notice the knots in this paper are of multiplicity two; thus, if we view the T-mesh in index space, each vertex in T-meshes corresponds to four vertices in index T-mesh. In a design-suitable T-mesh, it happens that each vertex is associated with four basis functions. Thus, there is a one-to-one correspondence between vertices and basis functions exactly in a design-suitable T-mesh. For convenience, in this paper, we still use T-meshes for constructing T-meshes and local refinement. The index T-mesh is used only to explain the control net.

For a design-suitable T-mesh $\mathcal{T}$, we construct basis functions on the extended T-mesh $\mathcal{T}_e$ as stated in section 3.2. The polynomial splines surface defined over $\mathcal{T}$ is defined as follows:

$$S(u,v) = \sum_{i=1}^{n} \sum_{k=1}^{4} \mathbf{P}_i^k B_i^k(u,v),\ (u,v) \in [a,b] \times [c,d], \tag{5}$$

where $B_i^k(u,v)$ are basis functions constructed in section 3.2 and $\mathbf{P}_i^k \in \mathbb{R}^3$ are the corresponding control points. Figure 6 presents the polynomial splines model with control nets. The construction method is based on the above polynomial splines with control nets defined over T-meshes; we design a goblet model and a dolphin model; see Figures 7 and 8.

For a design-suitable T-mesh, we define four basis functions at each vertex. If we need to adjust the surface part corresponding to a T-vertex, we can adjust the control point corresponding to the basis functions defined at the T-vertex to modify the surface directly. But for PHT-spline surface, it can only be adjusted indirectly by adjusting the control points corresponding to the basis functions defined at other basis vertices. Figure 9 shows the modification of the surface by adjusting the position of the control points corresponding to the T-vertices and the normal basis vertices.

4. Local Refinement

Given a design-suitable T-mesh $\mathcal{T}^1$, denote the extended T-mesh as $\mathcal{T}_e^1$. The edge insertion is to insert edges into $\mathcal{T}^1$, the resulting T-mesh is denoted by $\mathcal{T}^2$. The local refinement is attributed to how to construct the extended T-mesh $\mathcal{T}_e^2$ such that $\mathcal{S}(3,3,1,1,\mathcal{T}_e^1) \subseteq \mathcal{S}(3,3,1,1,\mathcal{T}_e^2)$, which is equivalent to $\mathcal{T}_e^1 \subseteq \mathcal{T}_e^2$.

Suppose a vertical edge $e = \overline{v_0 v_1}$ is inserted into a element F, and the 2-neighboring elements of F in vertical direction are denoted by F_i, $i = 1, 2, \ldots, l$ (if existing). The local refinement algorithm is described as follows:

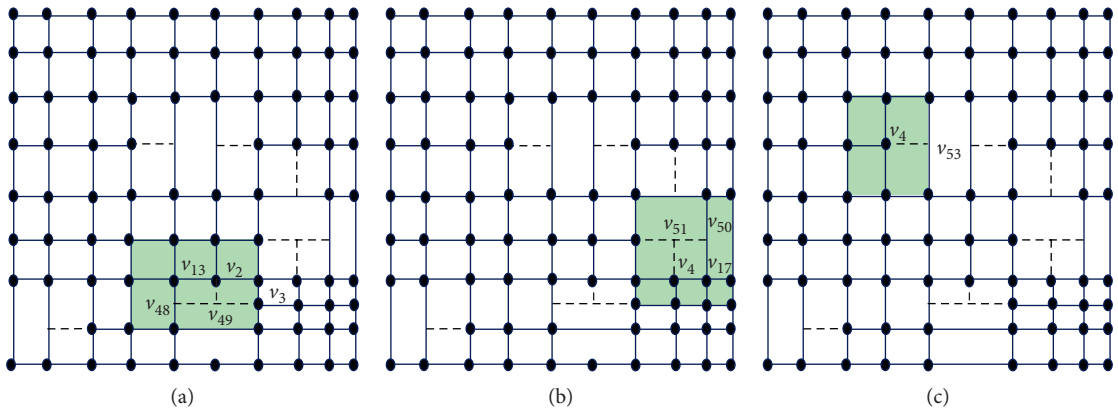

FIGURE 4: The support meshes of three basis vertices v_{13}, v_{17} and v_4. (a) T-mesh, (b) extended T-mesh, and (c) extended T-mesh.

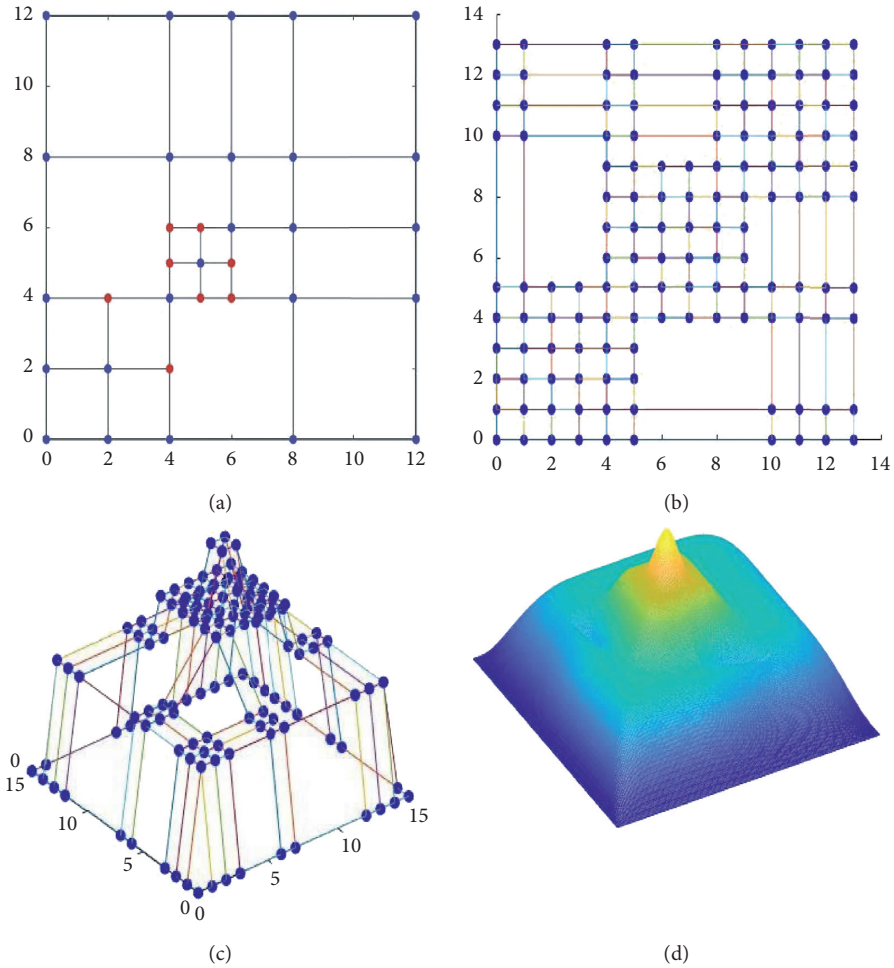

FIGURE 5: The index space and the control mesh. (a) A Design-suitable T-mesh. (b) The index mesh. (c) The control net. (d) The surface generated by the control net shown in (c).

(1) If $v_0\,(v_1)$ is a CT-vertex or a ECT-vertex, then extend them until it is either a CT-vertex or ECT-vertex.

(2) Among the T-vertices in $\{F_i\}_{i=1}^{l}$ and F, if the extension of a T-vertex in $\mathcal{T}_e^1$ intersects with e, then p-extend the T-vertex. The resulting T-mesh is denoted by $\mathcal{T}^2$.

(3) Construct the extended T-mesh $\mathcal{T}_e^2$.

(i) Update the new T-vertices which are in $\mathcal{T}^2$ but not in $\mathcal{T}^1$ by p-extending.

(ii) Keep the extensions in $\mathcal{T}_e^1$ of the original T-vertices.

We use Figure 10 to demonstrate the local refinement algorithm as proposed above. A design-suitable T-mesh $\mathcal{T}^1$ is shown in Figure 10(a) and the corresponding extended

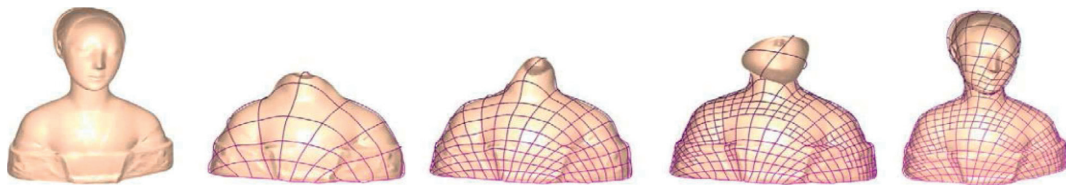

Figure 6: A model of the polynomial splines with control nets.

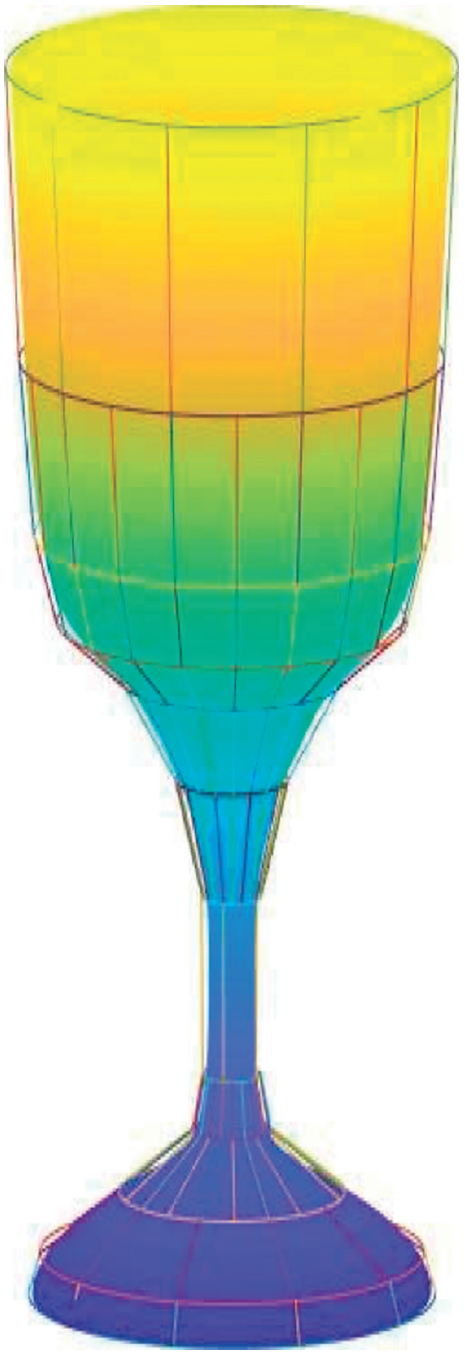

Figure 7: A goblet model designed by the polynomial splines with control nets.

T-mesh $\mathcal{T}_e^1$ is shown in Figure 10(b). A vertical edge $\overline{v_0v_1}$ (marked by red line) is inserted in $\mathcal{T}^1$, which is shown in Figure 10(c). Now v_0 is a CT-vertex and v_1 is a ECT-vertex; then they are extended as shown in Figure 10(d). For the horizontal T-vertices in F_1 and F, their extensions with respect to $\mathcal{T}_e^1$ intersect with e, extend them as shown in

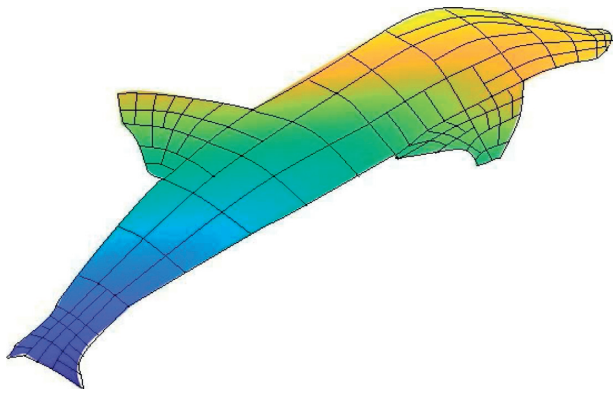

Figure 8: A dolphin model designed by the polynomial splines with control nets.

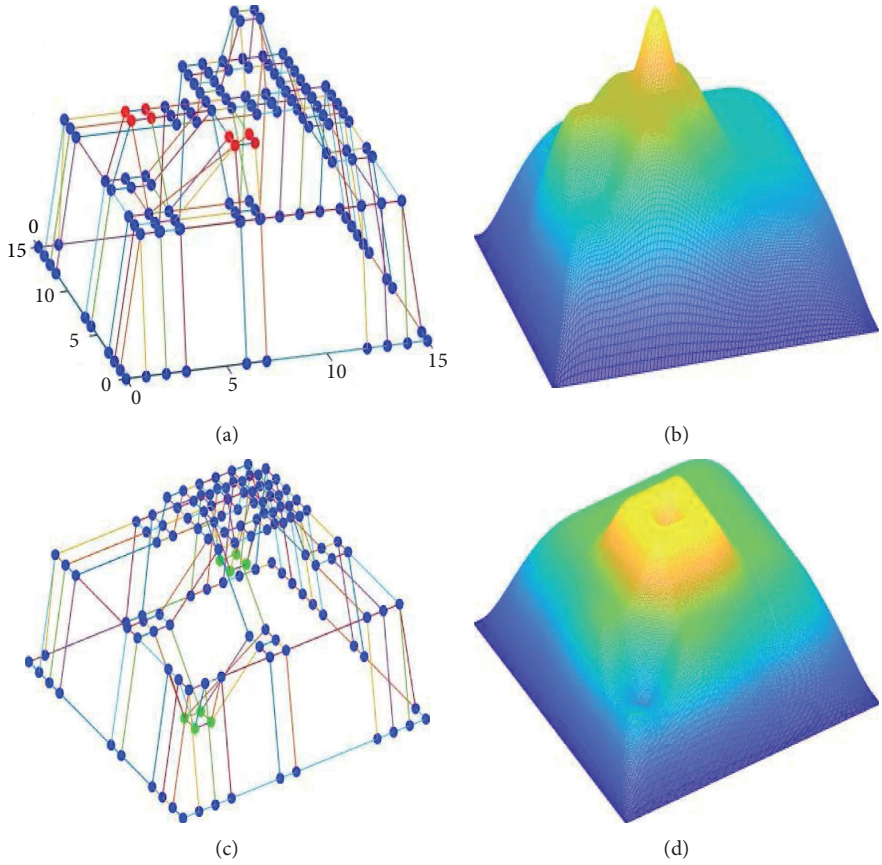

Figure 9: The index space and the control mesh. (a) Adjusting the position of the control points corresponding to the T-vertices in the Figure 5(c). (b) The surface figure obtained by adjusting the position of the control point corresponding to the T-vertices. (c) Adjusting the position of the control points corresponding to the basis vertices in the Figure 5(c). (d) The surface figure obtained by adjusting the position of the control point corresponding to the basis vertices.

Figure 10(e). The resulting T-mesh is a design-suitable T-mesh, denoted by $\mathcal{T}^2$. Finally, we construct the extended T-mesh corresponding to $\mathcal{T}^2$. We only need to update the extension of the new T-vertices. Figure 10(f) shows the extended T-mesh $\mathcal{T}_e^2$.

Figure 11 shows the local refinement of a diagonal T-mesh. The inserted edges are marked by red line segments and the additional inserted edges are marked by blue line segments in Figures 11(c) and 11(e). Figures 10(d) and 10(f) show the extended T-meshes of Figures 11(c) and 11(e), respectively. Obviously, the extended T-mesh of the diagonal T-mesh shown in Figure 11(a) is contained in both the extended T-meshes shown in Figures 10(d) and 10(f). Notice there are at most 1 neighboring elements of the element F are need to be inserted into edges; thus, the refinement are local.

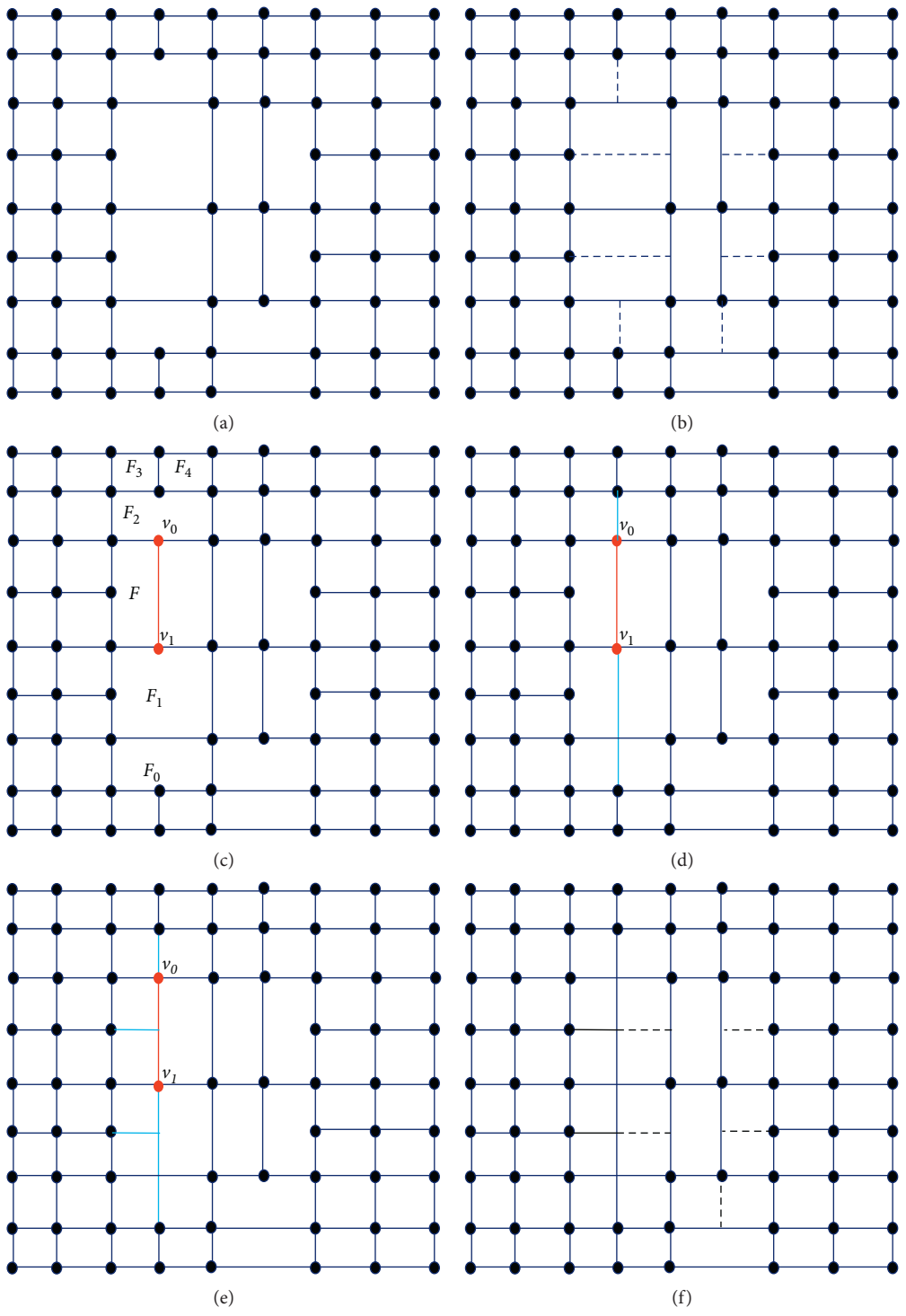

FIGURE 10: Steps of local refinement algorithm. (a) Design-suitable T-mesh $\mathcal{T}^1$, (b) extended T-mesh $\mathcal{T}_e^1$, (c) insert an edge (red line), (d) first step: extend v_0 and v_1, (e) design-suitable T-mesh $\mathcal{T}^2$, (f) extended T-mesh $\mathcal{T}_e^2$.

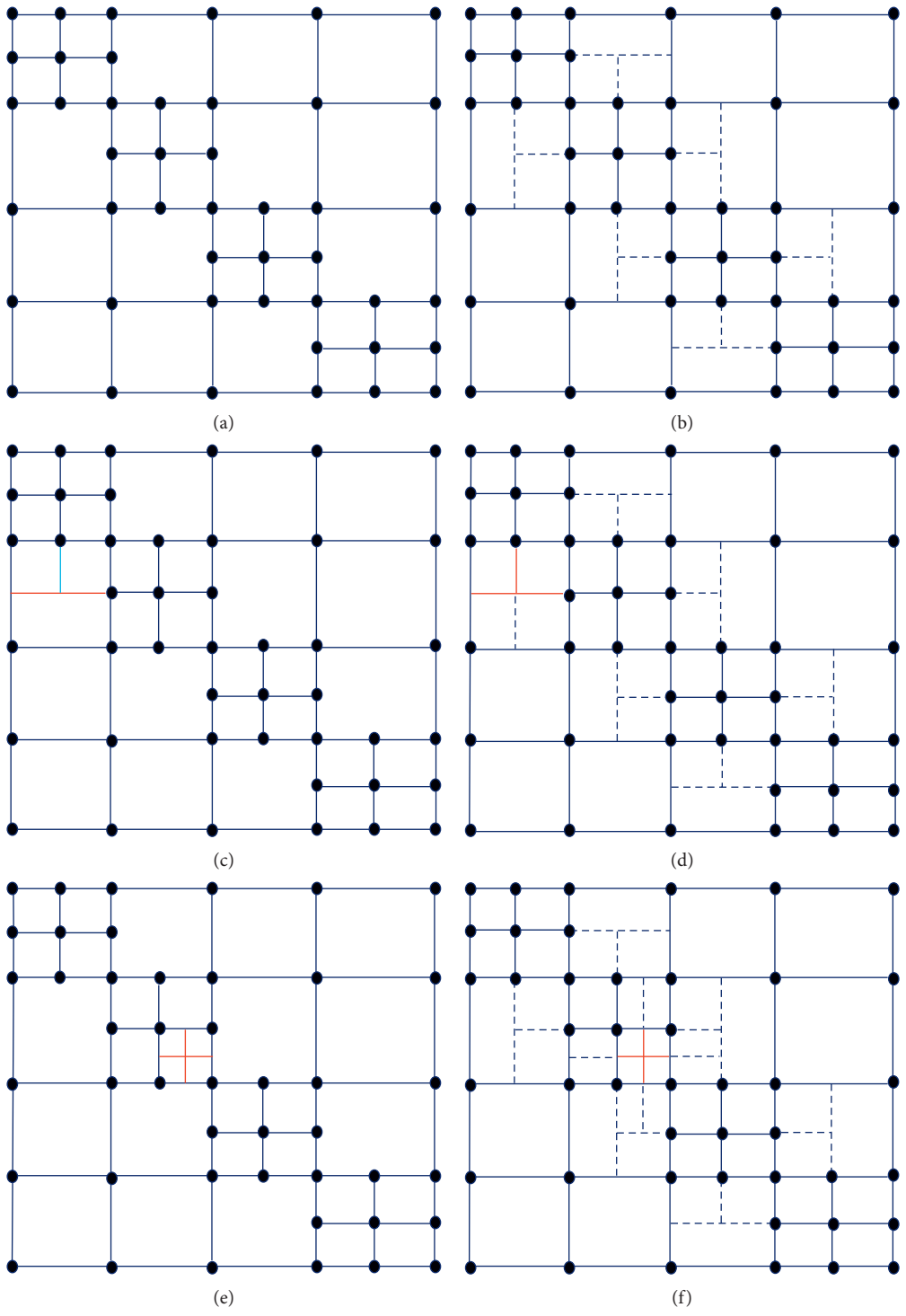

Figure 11: Local refinement of a diagonal T-mesh. (a) Design-suitable T-mesh $\mathcal{T}^1$. (b) Extended T-mesh $\mathcal{T}_e^1$. (c) Insert an edge (red line) $\mathcal{T}^2$. (d) Extended T-mesh $\mathcal{T}_e^2$. (e) Insert an edge (red line) $\mathcal{T}^2$. (f) Extended T-mesh $\mathcal{T}_e^2$.

5. Conclusions

In this paper, we present an algorithm for extending bicubic C^1 polynomial spline spaces defined over any given T-meshes such that there is a one-to-one correspondence between basis functions and vertices in the T-meshes. The key idea is to extend the T-junctions such that the T-junctions become basis vertices. Thus how to extend these T-junctions to avoid extra basis vertices produced is the main challenge. In this paper, we proposed a subset of T-meshes called design-suitable T-meshes. For design-suitable T-meshes, we first extend T-vertices through an appropriate order and then construct basis functions on the extended T-meshes. There is a one-to-one correspondence between basis functions and vertices in the T-mesh. Furthermore, it is easy to modify a given T-mesh such that it becomes a design-suitable T-mesh.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

The work was supported by the Natural Science Foundation of China (Nos. 11871447 and 11801393), the Natural Science Foundation of Jiangsu Province (No. BK20180831), and the Natural Science Foundation of the Education Department of Anhui Province (Grant nos. KJ2018A0363 and KJ2019A0580).

References

[1] T. W. Sederberg, J. Zheng, A. Bakenov, and A. Nasri, "T-splines and T-NURCCs," *ACM Transactions on Graphics*, vol. 22, no. 3, pp. 477–484, 2003.

[2] T. W. Sederberg, D. L. Cardon, G. T. Finnigan, N. S. North, J. Zheng, and T. Lyche, "T-spline simplification and local refinement," *ACM Transactions on Graphics*, vol. 23, no. 3, pp. 276–283, 2004.

[3] T. J. R. Hughes, J. A. Cottrell, and Y. Bazilevs, "Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement," *Computer Methods in Applied Mechanics and Engineering*, vol. 194, no. 39–41, pp. 4135–4195, 2005.

[4] J. A. Cottrell, T. J. R. Hughes, and Y. Bazilevs, *Isogeometric Analysis: Toward Integration of CAD and FEA*, Wiley, Hoboken, NJ, USA, 2009.

[5] D. R. Forsey and R. H. Bartels, "Hierarchical B-spline refinement," *ACM SIGGRAPH Computer Graphics*, vol. 22, no. 4, pp. 205–212, 1988.

[6] D. R. Forsey and R. H. Bartels, "Surface fitting with hierarchical splines," *ACM Transactions on Graphics (TOG)*, vol. 14, no. 2, pp. 134–161, 1995.

[7] A. V. Vuong, C. Giannelli, B. Jüttler, and B. Simeon, "A hierarchical approach to adaptive local refinement in isogeometric analysis," *Computer Methods in Applied Mechanics and Engineering*, vol. 200, pp. 3554–3567, 2011.

[8] C. Giannelli, B. Jüttler, and H. Speleers, "THB-splines: the truncated basis for hierarchical splines," *Computer Aided Geometric Design*, vol. 29, no. 7, pp. 485–498, 2012.

[9] X. Wei, Y. Zhang, L. Liu, and T. J. R. Hughes, "Truncated T-splines: fundamentals and methods," *Computer Methods in Applied Mechanics and Engineering*, vol. 316, pp. 349–372, 2017.

[10] X. Wei, Y. J. Zhang, and T. J. R. Hughes, "Truncated hierarchical tricubic C^0 spline construction on unstructured hexahedral meshes for isogeometric analysis applications," *Computers & Mathematics with Applications*, vol. 74, no. 9, pp. 2203–2220, 2017.

[11] X. Wei, Y. J. Zhang, D. Toshniwal et al., "Blended B-spline construction on unstructured quadrilateral and hexahedral meshes with optimal convergence rates in isogeometric analysis," *Computer Methods in Applied Mechanics and Engineering*, vol. 341, pp. 609–639, 2018.

[12] M. A. Scott, X. Li, T. W. Sederberg, and T. J. R. Hughes, "Local refinement of analysis-suitable T-splines," *Computer Methods in Applied Mechanics and Engineering*, vol. 213–216, pp. 206–222, 2012.

[13] T. Dokken, T. Lyche, and K. F. Pettersen, "Polynomial splines over locally refined box-partitions," *Computer Aided Geometric Design*, vol. 30, no. 3, pp. 331–356, 2013.

[14] H. Kang, F. Chen, and J. Deng, "Modified T-splines," *Computer Aided Geometric Design*, vol. 30, no. 9, pp. 827–843, 2013.

[15] J. Deng, F. Chen, X. Li et al., "Polynomial splines over hierarchical T-meshes," *Graphical Models*, vol. 70, no. 4, pp. 76–86, 2008.

[16] L. Tian, F. Chen, and Q. Du, "Adaptive finite element methods for elliptic equations over hierarchical T-meshes," *Journal of Computational and Applied Mathematics*, vol. 236, no. 5, pp. 878–891, 2011.

[17] N. Nguyen-Thanh, H. Nguyen-Xuan, S. P. A. Bordas, and T. Rabczuk, "Isogeometric analysis using polynomial splines over hierarchical T-meshes for two-dimensional elastic solids," *Computer Methods in Applied Mechanics and Engineering*, vol. 200, no. 21-22, pp. 1892–1908, 2011.

[18] N. Nguyen-Thanh, J. Kiendl, H. Nguyen-Xuan et al., "Rotation free isogeometric thin shell analysis using PHT-splines," *Computer Methods in Applied Mechanics and Engineering*, vol. 200, no. 47-48, pp. 3410–3424, 2011.

[19] P. Wang, J. Xu, J. Deng, and F. Chen, "Adaptive isogeometric analysis using rational PHT-splines," *Computer-Aided Design*, vol. 43, no. 11, pp. 1438–1448, 2011.

[20] X. Li, J. Deng, and F. Chen, "Surface modeling with polynomial splines over hierarchical T-meshes," *The Visual Computer*, vol. 23, no. 12, pp. 1027–1033, 2007.

[21] J. Wang, Z. Yang, L. Jin, J. Deng, and F. Chen, "Parallel and adaptive surface reconstruction based on implicit PHT-splines," *Computer Aided Geometric Design*, vol. 28, no. 8, pp. 463–474, 2011.

[22] Y. Jia, C. Anitescu, Y. J. Zhang, and T. Rabczuk, "An adaptive isogeometric analysis collocation method with a recovery-based error estimator," *Computer Methods in Applied Mechanics and Engineering*, vol. 345, pp. 52–74, 2019.

[23] Y. Jia, A. Cosmin, J. Z. Yongjie, G. Xu, C. Li, and R. Timon, "PHT-spline-based enhanced isogeometric collocation method," *Journal of Computer-Aided Design & Computer Graphics*, vol. 30, no. 4, pp. 702–706, 2018.

[24] H. Kang, J. Xu, F. Chen, and J. Deng, "A new basis for PHT-splines," *Graphical Models*, vol. 82, pp. 149–159, 2015.

[25] X. Li, J. Deng, and F. Chen, "Polynomial splines over general T-meshes," *The Visual Computer*, vol. 26, no. 4, pp. 277–286, 2010.

[26] T. Nguyen and J. Peters, "Refinable C^1 spline elements for irregular quad layout," *Computer Aided Geometric Design*, vol. 43, no. 8, pp. 123–130, 2016.

[27] Z. Wang, F. Chen, and J. Deng, "Evaluation algorithm of PHT-spline surfaces," *Numerical Mathematics: Theory, Methods and Applications*, vol. 10, no. 4, pp. 760–774, 2017.