Restricted Trivariate Polycube Splines for Volumetric Data Modeling

Kexiang Wang, Xin Li, *Member*, *IEEE Computer Society*, Bo Li, Huanhuan Xu, and Hong Qin, *Senior Member*, *IEEE*

Abstract—This paper presents a volumetric modeling framework to construct a novel spline scheme called restricted trivariate polycube splines (RTP-splines). The RTP-spline aims to generalize both trivariate T-splines and tensor-product B-splines; it uses solid polycube structure as underlying parametric domains and strictly bounds blending functions within such domains. We construct volumetric RTP-splines in a top-down fashion in four steps: 1) Extending the polycube domain to its bounding volume via space filling; 2) building the B-spline volume over the extended domain with restricted boundaries; 3) inserting duplicate knots by adding anchor points and performing local refinement; and 4) removing exterior cells and anchors. Besides local refinement inherited from general T-splines, the RTP-splines have a few attractive properties as follows: 1) They naturally model solid objects with complicated topologies/bifurcations using a one-piece continuous representation without domain trimming/patching/merging. 2) They have guaranteed semistandardness so that the functions and derivatives evaluation is very efficient. 3) Their restricted support regions of blending functions prevent control points from influencing other nearby domain regions that stay opposite to the immediate boundaries. These features are highly desirable for certain applications such as isogeometric analysis. We conduct extensive experiments on converting complicated solid models into RTP-splines, and demonstrate the proposed spline to be a powerful and promising tool for volumetric modeling and other scientific/engineering applications where data sets with multiattributes are prevalent.

Index Terms—Trivariate splines, polycube splines, polycube mapping.

1 Introduction

Volumetric data of massive size are now available in a wide variety of scientific and research fields, because of the rapid advancement of modern data acquisition technologies. A recurring problem is how to convert acquired 3D raw data of discrete samples into a continuous representation upon which simulation and analysis processes can be efficiently developed and accurately computed. The majority of traditional solid modeling techniques in the past four decades have been established upon the following theoretic foundations: constructive solid geometry (CSG), boundary representation (B-reps), and cell/space decomposition. Most of these representations lack the ability of smoothly modeling solid geometry. Smooth modeling of solids is critical for modern engineering design that requires direct and efficient applying physical simulation on volumetric regions, without the expensive procedures of remeshing finite-element structure, and converting from discrete representations to continuous ones and from linear finite elements to higher degree piecewise splines.

In practice, many real-world objects have complex geometries and nontrivial topologies. Constructing efficient representations for general solid objects in favor of physical simulation and engineering design is therefore highly challenging. Trivariate simplex splines [2] have been developed to model multidimensional, material attributes of volumetric objects. However, computing blending functions and their derivatives on simplex splines is not straightforward and much less efficient compared with nonuniform B-splines (NURBS) and tensor-product B-splines. Also, how to place boundary knots to avoid numerical degeneracies remains open. Trivariate simplex splines are defined over an unstructured tetrahedral mesh, which can be generated from boundary triangular meshes (e.g., using [3]). Although solid object of complex topologies and geometries can be modeled by trivariate simplex splines upon such unstructured structure, the majorities of simulation solvers prefer regular grids. Low-quality tetrahedral meshes usually cause large simulation errors and numerical instability. Motivated by current industrial practice in engineering design and analytic systems, we focus on designing a volumetric spline modeling framework based on structured grid domains. Tensor-product splines have the potentials to become an ideal representation scheme for this purpose.

In the framework of *isogeometric analysis* [4], [5], trivariate tensor-product B-splines/NURBS are directly used to model smooth geometry and material attributes of solid objects for physical simulation. Martin et al. [6] convert solid models to cylindrical trivariate B-splines by parameterizing models on solid cylinders. Due to the topological limitation of the cylindrical domain, the constructed trivariate tensor-product splines cannot model solid objects with bifurcations and arbitrary topologies, without enormous patch gluing/trimming efforts and imposing smoothness constraints along patch boundaries. Furthermore, tensor-product splines do not support local refinement

- *K. Wang, B. Li, and H. Qin are with the Department of Computer Science, Stony Brook University, Stony Brook, NY 11790-4400. E-mail: {kwang, bli, qin}@cs.sunysb.edu.*
- *X. Li and H. Xu are with the Department of Electrical and Computer Engineering and the Center of Computation and Technology, Louisiana State University, Baton Rouge, LA 70803. E-mail: {xinli, hxu4}@lsu.edu.*

Manuscript received 10 June 2010; revised 11 May 2011; accepted 20 May 2011; published online 13 June 2011.
Recommended for acceptance by W. Wang.
For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org, and reference IEEECS Log Number TVCG-2010-06-0118.
Digital Object Identifier no. 10.1109/TVCG.2011.102.

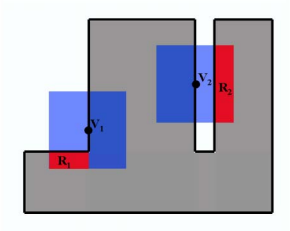


Fig. 1. Extra support regions. On a concave domain, if the supporting box of a blending function intersects with the domain boundary (e.g., boxes of v_1 and v_2), extra control points (e.g., in red regions) could contribute to the function blending unnecessarily.

and level-of-detail modeling because refining their basis functions will introduce many superfluous control points across the entire domain. As an extension of NURBS, T-splines [1], [7] solve this problem on semiregular grid domains. To the best of our knowledge, T-splines have not been generalized for 3D, multiattribute volumetric geometry, and data modeling.

Generalization of T-splines from surface to volumetric data is nontrivial. A general T-spline function defined over a bivariate domain can be formulated as

$$\mathbf{F}(u,v) = \frac{\sum_{i=1}^{n} w_i \mathbf{p}_i B_i(u,v)}{\sum_{i=1}^{n} w_i B_i(u,v)} \quad (u,v) \in \mathbb{R}^2, \qquad (1)$$

where $\mathbf{p}_i$ are control points associated with weights w_i, and $B_i(u,v)$ denotes the basis function. Two pieces of T-spline patches can be stitched together by blending boundary basis functions, and form a new T-spline that preserves smoothness across the boundary. Trivariate T-splines inherit such nice features, and T-splines defined on polycube volumetric domains can be similarly constructed by gluing a group of T-spline cubes. However, the calculation of this T-spline function and its derivatives requires dividing blending functions by the sum of all the contributing ones. This makes the evaluation computationally inefficient. On the other hand, *Semistandard* T-splines introduced recently in [1] guarantee $\sum_{i=1}^{n} w_i B_i(u,v) \equiv 1$ in (1) across the entire domain. In this setting, the computation of $\mathbf{F}(u,v)$ and its derivatives can be much more efficient.

However, constructing a semistandard T-spline is challenging over nontrivial parametric domains. Also, conventional T-splines are defined with floating boundaries, i.e., the support regions of blending functions may go beyond the domain boundaries. Such a floating-boundary scheme upon a polycube domain causes control points to unnecessarily contribute to extra domain regions. Two examples are illustrated as red regions in Fig. 1. This might cause geometric inconsistencies in modeling the underlying solid objects and in physical simulations. Therefore, it would be ideal to have a trivariate spline inherit from T-splines, that 1) is defined within the largest visible region inside the domain, and 2) has the property of semistandardness. Such splines will greatly facilitate direct modeling and physical simulation of arbitrary solid objects with complex geometries and sophisticated topologies. The spline constructed in this paper has these properties, and we call it the Restricted Trivariate Polycube Spline (RTP-spline).

This paper presents a framework of RTP-spline construction and the data conversion of volumetric models to this spline representation. It has the following major contributions:

- We formulate a new spline (RTP-spline) scheme over a polycube domain, restricting blending functions inside the domain boundaries. The RTP-spline has the following advantages:
  - It is capable of local refinement.
  - Its function and derivative evaluations are much more efficient than that of traditional T-spline surfaces.
  - Its polycube domain enables natural modeling of arbitrary solid objects, and the domain shape mimics the geometry and topology of the model and introduces low parameterization distortion and few singularity points
  - Its restricted boundaries ensure that the physical modeling and simulation adhere to geometry of underlying objects.
- We develop a novel framework to construct RTP-splines in an effective top-down fashion.
- We construct RTP-splines on several volumetric models with both geometry and synthesized textures (to mimic material properties), which demonstrates that our RTP-splines can model not only geometry but also multiattribute fields within a unified paradigm.

The remainder of this paper is organized as follows: we review the related literature in Section 2, then introduce preliminaries and define necessary notations in Section 3. The methodology of RTP-spline construction is illustrated in Section 4. The entire process of converting discrete volumetric data into the spline representation is then explained in Section 5. We demonstrate experimental results in Section 6 and conclude the paper in Section 7.

2 Related Works

Research on spline-based volumetric modeling has gained much attention recently. Four-dimensional uniform rational cubic B-spline volume is used to constructively model FRep solids defined by real-valued functions [8]. Martin and Cohen [9] model physical attributes across a trivariate NURBS volume. It is more desirable in engineering design to have an integrated modeling framework that can represent geometry and material and conduct simulations simultaneously. Trivariate NURBS are used to model skeletal muscle with anisotropic attributes [4], on which NURBS-FEM analysis is directly conducted. Martin et al. [6] parameterize a volumetric solid into a solid cylinder upon which a trivariate B-spline is constructed. Hua et al. develop a framework based on triangular simplex splines [2] to model and render multidimensional material attributes of solid objects with complicated geometries and topologies.

The splines proposed in this paper are founded upon the T-spline technique [7]. The T-spline is a generalization of NURBS, but permits T-junctions on its control mesh and enables local insertion of additional knots without introducing superfluous control points. A local refinement method

is proposed in [1] and [10] to simplify NURBS surfaces into T-spline representations by removing superfluous control points. Ipson [11] thoroughly discusses how to merge B-spline patches defined over different local domains to get a single T-spline representation on manifold domain.

Bazilevs et al. [12] propose an isogeometric analysis framework based on T-splines. Its main focus is on planar T-splines for surfaces, and volumetric T-splines are only briefly mentioned without offering any technical details. Generalized trivariate T-splines (whose control points are associated with weights) are employed by Song and Yang [13] to model free-form deformation fields. For the purpose of shape metamorphosis, 3D level sets represented by T-splines are adopted in [14], [15], [16], and [17] for its efficiency. This is because, the distribution of T-spline control points can be made adaptive to the geometries of the morphing objects.

Our work relies on the construction and parameterization of a polycube domain. The parameterization on polycubes originated for seamless texture mapping with low distortion [18]. Polycubes serve as nice parametric domains because they approximate well the geometry of the model and possess great regularity. A polycube mapping can be constructed either manually [18], [19], [20] or automatically [21], [22]. Based upon specially designed surface parameterization, Wang et al. [19] build manifold bivariate T-spline over a polycube that can handle models with arbitrary topology. A few recent works [23], [24], [25], [26], [27], [28] study the parameterization of a solid object to canonical domains such as spheres, polycubes, star-shaped volumes, etc. Volumetric parameterization typically starts from a given surface mapping, and parameterizing volumetric data onto a solid polycube domain serves as an important preprocessing step for the conversion of any solid model to RTP-splines.

3 Preliminaries and Notations

In this section, we introduce a general algorithm to construct trivariate T-spline with duplicate knots on a regular box domain, review the theory of basis function refinement, and define necessary notations for the rest of the paper.

3.1 Trivariate T-Splines with Duplicate Knots

Defined on a grid structure that allows T-junctions (or T-mesh), the T-spline proposed in [7] is a generalization of nonuniform B-splines. When considering a simple cube domain, the definition of T-spline surfaces can be straightforwardly extended to three dimensions and generate trivariate T-splines on T-lattice grids, where "T-junctions" are referred to the intersections between faces and/or lines.

Let $T(\mathcal{V}, \mathcal{C}, \mathcal{F})$ denote a rectilinear grid structure that permits T-junctions, where $\mathcal{V}$, $\mathcal{C}$, and $\mathcal{F}$ are sets of vertices, cells, and faces, respectively. $\mathcal{K} \subseteq \mathcal{V} \times \{-1, 0, +1\}^3$ denote a set of *anchors* attached to vertices. At most 27 anchors are allowed at each vertex, and they can be visually imagined to be organized on a $3 \times 3 \times 3$ grid of infinitesimal size, as shown in Fig. 2. We require that each vertex has a *master anchor* at the center of the local grid, while the others are optional and called *subanchors*. In the rest of the paper, we denote an anchor at $\mathbf{v}_i$ as $\mathbf{k}_{i(\alpha,\beta,\gamma)}$, in which the triplet (α, β, γ) indicates a unique nodal position on the local grid. Given $\mathbf{v}_i = (v_i^0, v_i^1, v_i^2)$, all the corresponding anchors $\mathbf{k}_{i(\alpha,\beta,\gamma)}$ share the same coordinator (v_i^0, v_i^1, v_i^2) in parametric space. To distinguish these anchors in T-spline construction, we define $\bar{\mathbf{k}}_{i(\alpha,\beta,\gamma)} = \mathbf{v}_i + (\alpha, \beta, \gamma)\epsilon$ as the coordinator of $\mathbf{v}_{i(\alpha,\beta,\gamma)}$ in *construction space*, where ϵ is an infinitesimal with respect to the minimal cell size. In the rest of this paper, we sometimes represent an anchor by a simpler notation $\mathbf{k}_j$, where j indicates the index of $\mathbf{k}_{j(\alpha,\beta,\gamma)}$ in $\mathcal{K}$.

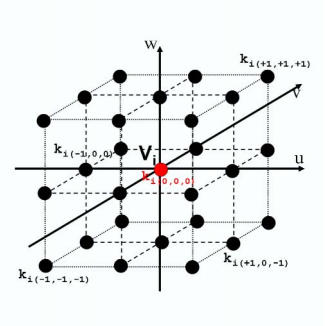


Fig. 2. A vertex $\mathbf{v}_i$ can have at most 27 anchors organized on a $3 \times 3 \times 3$ virtual grid. The central one (red) is the master anchor and the rest (black) are subanchors.

Given T and $\mathcal{K}$, a trivariate T-spline can be defined as

$$\mathbf{F}(u, v, w) = \frac{\sum_{i=1}^{|\mathcal{B}|} \mathbf{p}_i B_i(u, v, w)}{\sum_{i=1}^{|\mathcal{B}|} B_i(u, v, w)} \quad (u, v, w) \in \mathbb{R}^3, \tag{2}$$

where (u, v, w) denotes 3D parametric coordinates, $\mathbf{p}_i$ are control points, and $\mathcal{B} = \{B_i(u, v, w)\}$ is the collection of blending functions. Each $B_i(u, v, w)$ is a tensor product of three B-spline basis functions written as

$$B_i(u, v, w) = N_{i0}^3(u) N_{i1}^3(v) N_{i2}^3(w), \tag{3}$$

where $N_{i0}^3(u)$, $N_{i1}^3(v)$, and $N_{i2}^3(w)$ are defined along u, v, and w directions, respectively. In the case of cubic T-spline, the univariate function N_{ij}^3 is constructed upon the knot vector Ξ_i^j, which is deduced from T and a collection of anchors $\mathcal{K}$.

We refer the knot vector in construction space by notation $\bar{\Xi}_i^j = [\bar{\xi}_{i0}^j, \bar{\xi}_{i1}^j, \bar{\xi}_{i2}^j, \bar{\xi}_{i3}^j, \bar{\xi}_{i4}^j]$ for the rest of the paper, unless mentioned otherwise. In the case of cubic T-splines, each blending function must be associated with an anchor, which coincides with the middle knot of its three knot vectors in the construction space.

To infer knot vectors from a T-lattice is essentially the same as that for T-mesh, except that the searching is conducted in construction space. Starting from an anchor $\bar{\mathbf{k}} = (\bar{\xi}_{i2}^0, \bar{\xi}_{i2}^1, \bar{\xi}_{i2}^2)$, $\bar{\xi}_{i3}^0$ and $\bar{\xi}_{i4}^0$ are found by shooting a ray $L(t) = (\bar{\xi}_{i2}^0 + t, \bar{\xi}_{i2}^1, \bar{\xi}_{i2}^2)$ in the construction space. ξ_{i3}^0 and ξ_{i4}^0 are the corresponding coordinate values of the first two intersections where $L(t)$ comes across either an anchor or a face in $\mathcal{F}$. If $L(t)$ does not make two intersections before traveling outside T, the last coordinate value is repeated, e.g., $\xi_{i3}^0 = \xi_{i4}^0$ or $\xi_{i2}^0 = \xi_{i3}^0 = \xi_{i4}^0$ (see Fig. 3). The knots along other directions are determined in a similar fashion.

3.2 Refinement of B-Spline Functions

To refine blending functions on trivariate T-splines, we need to review the knot insertion algorithm for univariate

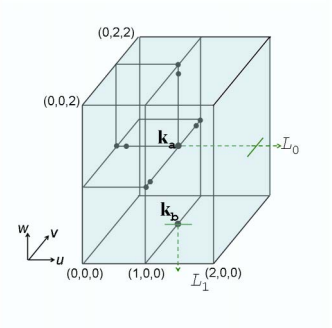


Fig. 3. Knot vectors are derived from a T-lattice associated with a set of anchors (dots). The knot vector from $\mathbf{k}_a$ is $[0,0,1,2,2]$ in $+u$ direction, where 2 repeats twice because L_0 intersects once with the rightmost boundary. The knot vector from $\mathbf{k}_b$ toward $-w$ direction has 0 repeated three times because L_1 intersects nothing from T or $\mathcal{K}$ except $\mathbf{k}_b$.

B-spline functions. Let $\Xi = [\xi_0, \xi_1, \xi_2, \xi_3, \xi_4]$ be a knot vector and $N(\xi)$ denote the cubic B-spline basis function defined on it. If there is an additional knot $k \in [\xi_0, \xi_4]$ inserted into Ξ, $N(\xi)$ can be written as a linear combination of two scaled B-spline functions

$$N(\xi) = c_1 N_1(\xi) + c_2 N_2(\xi), \tag{4}$$

where c_1, c_2, and knot vectors for $N_1(\xi)$ and $N_2(\xi)$ are all determined by the rules in Table 1.

TABLE 1
Refinement of $N(\xi)$ by Inserting k into Knot Vector $[\xi_0, \xi_1, \xi_2, \xi_3, \xi_4]$, Which Generates Two Basis Functions $N_1(\xi)$ and $N_2(\xi)$, Scaled by c_1 and c_2, Respectively

k	c_1	c_2	knot vector of $N_1(\xi)$	knot vector of $N_2(\xi)$
$\xi_0 \le k < \xi_1$	$\frac{k-\xi_0}{\xi_3-\xi_0}$	1	$[\xi_0, k, \xi_1, \xi_2, \xi_3]$	$[k, \xi_1, \xi_2, \xi_3, \xi_4]$
$\xi_1 \le k < \xi_2$	$\frac{k-\xi_0}{\xi_3-\xi_0}$	$\frac{\xi_4-k}{\xi_4-\xi_1}$	$[\xi_0, \xi_1, k, \xi_2, \xi_3]$	$[\xi_1, k, \xi_2, \xi_3, \xi_4]$
$\xi_2 \le k < \xi_3$	$\frac{k-\xi_0}{\xi_3-\xi_0}$	$\frac{\xi_4-k}{\xi_4-\xi_1}$	$[\xi_0, \xi_1, \xi_2, k, \xi_3]$	$[\xi_1, \xi_2, k, \xi_3, \xi_4]$
$\xi_3 \le k \le \xi_4$	1	$\frac{\xi_4-k}{\xi_4-\xi_1}$	$[\xi_0, \xi_1, \xi_2, \xi_3, k]$	$[\xi_1, \xi_2, \xi_3, k, \xi_4]$

4 Constructing RTP-splines

The construction of RTP-splines includes four major steps (see Fig. 4):

1. Extending given polycube P domain to a box domain.
2. Building trivariate B-splines with restricted boundaries.
3. Introducing duplicate knots by inserting additional anchors, and performing local refinement to separate interior and exterior blending functions.
4. Producing RTP-splines by removing structures/anchors outside P.

4.1 Extending Polycubes to Bounding Boxes

Following the notations introduced in Section 3.1 on the trivariate T-spline domain, let $P = (\mathcal{V}^P, \mathcal{C}^P, \mathcal{F}^P)$ be a given polycube structure, where $\mathcal{V}^P$, $\mathcal{C}^P$, and $\mathcal{F}^P$ denote vertices, cubes, and cell faces, respectively. In order to extend P to a box volume with rectilinear grids, P should not have T-junctions or intersections between its cell faces. Our parametric polycube domains (see Section 5.1) do not contain T-junctions. If other polycube mapping methods are used to construct the parametric domain and the generated domain has T-junctions, we can always eliminate them simply by splitting the cells across the domain, through the extended planes of these intersecting cell faces. Now, P can be extended to its bounding box domain $T(\mathcal{V}, \mathcal{C}, \mathcal{F})$ by filling in some solid cuboid structures $G = (\mathcal{V}^G, \mathcal{C}^G, \mathcal{F}^G)$, where $\mathcal{V}^G = \mathcal{V} - \mathcal{V}^P$, $\mathcal{C}^G = \mathcal{C} - \mathcal{C}^P$, and $\mathcal{F}^G = \mathcal{F} - \mathcal{F}^P$. G represents the exterior structure of P and we call its domain the *ghost region*. Note that there is a rectilinear grid embedded in the space of T, and the grid coordinates in k-axis direction are represented by

$$\mathbf{S}_k = \left[s_1^k, s_2^k, \ldots, s_{n_k}^k\right] \quad k = 0, 1, 2,$$

where n_k is the resolution of rectilinear grid along k-axis.

4.2 Building the B-Spline Volume with Restricted Boundary

With the bounding box domain T constructed, it is not difficult to construct a trivariate tensor-product B-spline on it. We may use $\mathbf{K}_s$ as the knot vectors to define the trivariate B-spline. However, $\mathbf{K}_s$ must be augmented to ensure that the definition is valid and covers the entire domain T. One method is to add extra knots outside the domain region, generating a floating-boundary scheme. In this paper, we duplicate the knots at both ends of $\mathbf{S}_k$ in order to restrict the B-spline blending function within the domain T, i.e., $\mathbf{S}_k$ turns into

$$\mathbf{S}_k = \left[s_1^k, s_1^k, s_1^k, s_1^k, s_2^k, \ldots, s_{n_k-1}^k, s_{n_k}^k, s_{n_k}^k, s_{n_k}^k, s_{n_k}^k\right]$$

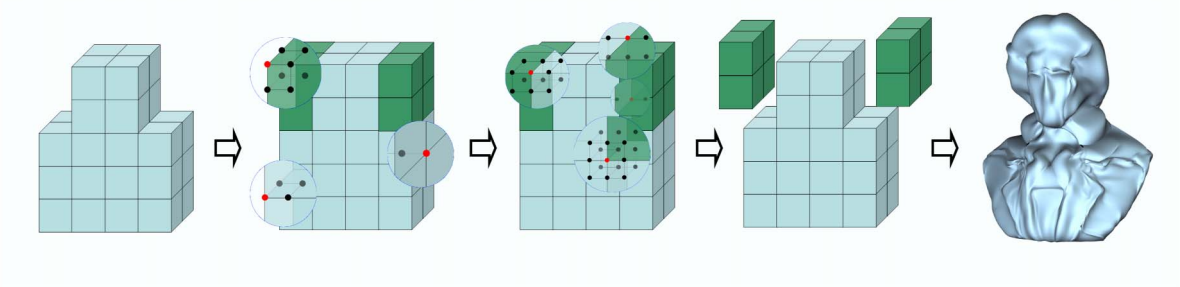


Fig. 4. RTP-spline construction pipeline. (1) Extend polycube domain to its bounding box. (2) Build B-spline volume with bounded boundaries. (3) Insert anchors and refine blending functions. (4) Remove exterior regions.

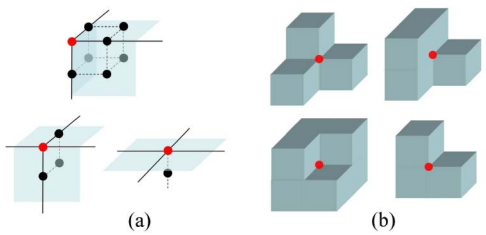


Fig. 5. (a) Knot configurations at corner, edge, and face vertices for restricted boundaries. (b) Examples of extraordinary corners on a polycube.

in which three extra knots are added to each end. Therefore, the trivariate tensor-product B-spline defined on T is formulated as

$$\mathbf{F}(u,v,w)=\sum_{i=1}^{n}\mathbf{p}_i B_i(u,v,w)\quad (u,v,w)\in \mathbb{R}^3, \tag{5}$$

where $n=(n_0+2)\times(n_1+2)\times(n_2+2)$ is the number of control points, and $B_i(u,v,w)$ are blending functions defined in (3).

Alternatively, we can obtain $\mathbf{F}$ by constructing blending functions similar to T-splines (Section 3.1), instead of computing them from three global knot vectors. We let

$$\mathcal{S}=\left\{s_1^0,s_1^0+\epsilon,\ldots,s_{n_0}^0-\epsilon,s_{n_0}^0\right\}\times\left\{s_1^1,s_1^1+\epsilon,\ldots,s_{n_1}^1-\epsilon,s_{n_1}^1\right\}\times\left\{s_1^2,s_1^2+\epsilon,\ldots,s_{n_2}^2-\epsilon,s_{n_2}^2\right\}$$

and choose the anchor set $\mathcal{K}=\{\mathbf{k}_{i(\alpha,\beta,\gamma)}|\bar{\mathbf{k}}_{i(\alpha,\beta,\gamma)}\in\mathcal{S}\}$, then build blending functions associated with each anchor. $\mathcal{K}$ contains subanchors that only exist at corner, edge, and face vertices (see their configurations in Fig. 5a). These subanchors guarantee partition of unity of $\mathbf{F}$ and limit the influential regions of blending functions within the domain T.

4.3 Local Refinement and Anchor Insertion

Let internal and ghost blending functions refer to the blending functions associated with anchors in P and G, respectively. In this section, we seek to refine existing blending functions with knot insertion and local refinement, so that the resulting internal and ghost blending functions are isolated and restricted boundary forms along the surface of P. More precisely, our goal is to enforce the following rules to the blending function set:

1. No ghost blending function influences any part of the polycube domain.
2. Semistandardness is preserved on the internal blending function set if G and all the ghost anchors are removed.
3. No internal blending function influences any region outside the polycube domain if G and all the ghost anchors are removed.

To achieve this goal, we systematically add new anchors in two steps. First, add subanchors at the polycube boundary vertices (Section 4.3.3). Second, keep inserting subanchors to refine those blending functions that violate the above rules, until no violation exists. Adding new subanchors ultimately introduces duplicate knots into knot vectors, which serves for two purposes: 1) reducing the influential region of a blending function and 2) degenerating the continuity of a blending function to C^0 at desired places (Section 4.3.2). Moreover, as new anchors may lead to disagreements between existing blending functions and underlying knot vectors implied by T and new $\mathcal{K}$, an algorithm (Section 4.3.1) is necessary to resolve these inconsistencies after new anchors have been inserted.

4.3.1 Local Refinement of Blending Functions

We need to introduce an algorithm to update blending functions $\mathcal{B}$ accordingly, once there occurs any change in the anchor set $\mathcal{K}$ and/or the domain structure T. The refinement algorithm proposed in [1] and [10] is designed for surface editing, the primary goal of which is to preserve the shape of a T-spline surface whenever new control points are inserted. In this paper, we extend this algorithm to 3D and enhance it to support trivariate T-spline with duplicate knots. By interpreting the B-spline volume previously obtained as a general trivariate T-spline, we can rewrite its representation from (5) to

$$\mathbf{F}(u,v,w)=\frac{\sum_{i=1}^{|\mathcal{B}|}w_i\mathbf{p}_i B_i(u,v,w)}{\sum_{i=1}^{|\mathcal{B}|}w_i B_i(u,v,w)}\quad (u,v,w)\in \mathbb{R}^3, \tag{6}$$

where w_i is the weight associated with each blending function B_i. Note that the T-spline so far is essentially a B-spline volume: $\sum_{i=1}^{|\mathcal{B}|}w_i B_i(u,v,w)\equiv 1$ for any (u,v,w) and $w_i=1$ for any i.

Let $\mathcal{K}^*$ denote the updated anchor set and $T^*(\mathcal{V}^*,\mathcal{C}^*,\mathcal{F}^*)$ be the new grid structure after vertex insertion or cell splitting. Given $\mathcal{K}^*$, T^*, $\mathcal{W}$, and $\mathcal{B}$, Algorithm 1 generates new blending function set $\mathcal{B}^*$ with new corresponding weights $\mathcal{W}^*$, as well as the updated versions of $\mathcal{K}^*$ and $\mathcal{T}^*$.

In Algorithm 1, the superscript indicates the index of the original blending function from $\mathcal{B}$, with which a variable is associated, and the subscript indicates the index of the associated central anchor. For example, B_i^t is a blending function associated with anchor $\mathbf{k}_i$ and originates from the tth blending function from $\mathcal{B}$. The star superscript indicates that the variables are obtained from the modified domain T^*, e.g., Ξ_i^* denotes the knot vectors deduced from T^* and *centered* at $\mathbf{k}_i^*$ (i.e., the three middle knots of Ξ_i^* coincide with $\mathbf{k}_i^*$).

The basic idea of Algorithm 1 is as follows: first, we decouple blending functions from T and $\mathcal{K}$. Then, by either inserting new anchors or refining basis functions (Section 3.2), we keep resolving the inconsistencies between $\mathcal{B}$ and the local knot vectors implied by $\mathcal{K}^*$ and $\mathcal{T}^*$. A cell splits into half if there are vertices on its edges forming an axis-aligned plane. Finally, any blending functions arising from different refinements but having equivalent knot vectors in parametric space are combined into a single one with their weights being added together.

Note that any blending function introduced by Algorithm 1 must center at a certain anchor, but not vice versa, i.e., there could be anchors not associated with any blending functions. Moreover, the new T-spline after refinement is still semistandard, because the denominators in (6) remain unchanged in Algorithm 1, because of the fact

$$w_i^t B_i^t\equiv w_i^t\cdot c_1 B_j^t+w_i^t\cdot c_2 B_i^t\equiv \tilde{w_j^t}\tilde{B_j^t}+\tilde{w_i^t}\tilde{B_i^t}.$$

Algorithm 1: Refinement of trivariate T-spine blending functions in support of duplicate knots

Input: $T^*(\mathcal{V}^*, \mathcal{C}^*, \mathcal{F}^*)$, $\mathcal{K}^*$, $\mathcal{B}$ and $\mathcal{W}$.
Output: T^*, $\mathcal{K}^*$, $\mathcal{B}^*$ and $\mathcal{W}^*$
s.t. $\sum_{i=1}^{|\mathcal{B}^*|} w_i^* B_i^* \equiv \sum_{i=1}^{|\mathcal{B}|} w_i B_i$

$Q \Leftarrow \{(w_i^i, B_i^i) : w_i \in \mathcal{W}, B_i \in \mathcal{B}\}$
while $\exists (w_i^t, B_i^t) \in Q : \Xi_i^t \neq \Xi_i^*$ *in parametric space* **do**
  forall the $(w_i^t, B_i^t) \in Q$ **do**
    infer the knot vectors Ξ_i^* that is *centered* at $\mathbf{k}_i^*$ from T^*
    if $\Xi_i^t = \Xi_i^*$ *in parametric space* **then**
      $\Xi_i^t \Leftarrow \Xi_i^*$
    else if Ξ_i^* *is more refined than* Ξ_i^t **then**
      insert a knot of Ξ_i^* into Ξ_i^t which is not there, and do the refinement:
      $B_i^t = c_1 \tilde{B}_j^t + c_2 \tilde{B}_i^t$ (Section 3.2)
      $\tilde{w_j^t} \Leftarrow w_i^t \cdot c_1$; $\tilde{w_i^t} \Leftarrow w_i^t \cdot c_2$
      $Q \Leftarrow Q - \{(w_i^t, B_i^t)\} \cup \{(\tilde{w_j^t}, \tilde{B_j^t}), (\tilde{w_i^t}, \tilde{B_i^t})\}$
    else if Ξ_i^t *indicates an anchor* $\mathbf{k}_{j(\alpha,\beta,\gamma)} \notin \mathcal{K}^*$ **then**
      $\mathcal{K}^* \Leftarrow \mathcal{K}^* \cup \mathbf{k}_{j(\alpha,\beta,\gamma)}$
      if $\mathbf{k}_{j(0,0,0)} \notin \mathcal{K}^*$ **then**
        $\mathcal{K}^* \Leftarrow \mathcal{K}^* \cup \{\mathbf{k}_{j(0,0,0)}\}$
        $\mathcal{V}^* \Leftarrow \mathcal{V}^* \cup \{\mathbf{v}_j\}$ `// Insert a new vertex`
      end if
    end if
  end forall
  forall the $c \in \mathcal{C}^*$ **do**
    if *vertices on the edges of* c *form an axis-aligned plane that splits* c *into* c_1 *and* c_2 **then**
      $\mathcal{C}^* \Leftarrow \mathcal{C}^* - \{c\} \cup \{c_1, c_2\}$ `// divide` c `into` c_1 `and` c_2
    end if
  end forall
end while
$\mathcal{B}^* \Leftarrow \{B_i : (w_i^t, B_i^t) \in Q\}$
$\mathcal{W}^* \Leftarrow \{w_j = \sum_{\forall (w_j^t, B_j^t) \in Q} w_j^t\}$

4.3.2 Modifying Blending Functions with Anchor Insertions

The anchor operation is our fundamental tool to modify existing blending functions of trivariate T-splines in order to get rid of all violations against rules 1, 2, and 3. As blending functions of trivariate T-splines are tensor products of three univariate cubic B-spline bases, let us illustrate this method in 1D by using two examples given in Fig. 6. In Fig. 6a, $N_0 = N[-2,-1,0,1,2]$ represents an internal basis which apparently violates rule 2. If two extra knots 0s are inserted, N_0 is refined into two internal bases $N_1 = N[-2,-1,0,0,0]$ and $N_2 = N[-1,0,0,0,1]$, one ghost basis $N_3 = N[0,0,0,1,2]$, such that $N_0 = \frac{2}{3}N_1 + \frac{2}{3}N_2 + \frac{2}{3}N_3$ according to the refinement algorithm in Section 3.2. Once

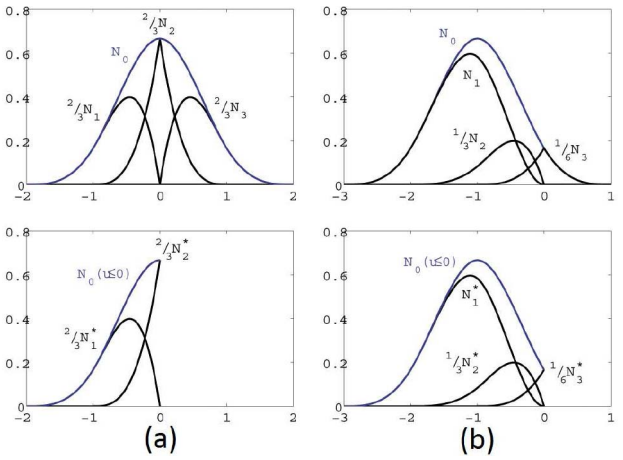


Fig. 6. Examples of eliminating violations against rules 1 and 2 in the case of cubic B-spline basis functions. Suppose $x = 0$ is the boundary and the ghost region covering $(0, \infty)$ in (a), N_0 represents an internal basis against rule 2. After two extra knots 0 are inserted, N_0 is refined to N_1 and N_2 which comply with rule 2. In (b), the ghost region covers $(-\infty, 0)$ and ghost basis N_0 violates rule 1. After inserting duplicate knots, it is replaced by N_1 and N_2 in ghost blending functions. Thus, no violation against rule 1 exists.

the ghost region is gone, N_1 and N_2 change to $N_1^* = N[-2,-1,0,0,0]$ and $N_2^* = N[-1,0,0,0,0]$, respectively, and we still have $N_0(u) = \frac{2}{3}N_1^*(u) + \frac{2}{3}N_2^*(u)$ on $u \in [-2,0]$, as shown in the bottom of Fig. 6a. Therefore, the violation against rule 2 is successfully eliminated. Fig. 6b depicts a scenario where $N_0 = N[-3,-2,-1,0,1]$ violates rule 1 overlaps with the domain region at $[0,1]$. By inserting two duplicate knots at 0, we may replace N_0 with two resulting ghost bases N_1 and N_2, both of which abide with rule 1. For trivariate T-splines, knot insertions are replaced by anchor insertions conducted on T-lattice, and a much more complex refine algorithm (see Section 4.3.1) is employed instead.

4.3.3 Anchor Insertions on Polycube Boundary

It is easy to see that the blending functions associated with those master anchors either on, or adjacent to the interfaces between P and G are in violation of rule 2. Therefore, we need to insert subanchors to boundary vertices. The basic idea is analogous to that in Section 4.2 where subanchors are added on the surface of a box domain to ensure its restricted boundary. However, a variety of corner types may be found on polycube surfaces (see Fig. 5b); thus, we have to handle all of them for proper anchor insertions. To exhaust all possible corner types, then choose subanchors to insert is tedious and inefficient. Instead, we develop a general algorithm to determine which subanchors to be inserted at arbitrary boundary vertex. Given a boundary vertex $\mathbf{v}_i$, we first add the master anchor to it, along with all the subanchors that lie within the domain of T in construction space. Then, the subanchors lying within the domain of P in construction space are colored red, and the others are blue. If there exists $\mathbf{k}_{i(-\alpha,\beta,\gamma)} \in \mathcal{K}$ for all $\mathbf{k}_{i(\alpha,\beta,\gamma)} \in \mathcal{K}$ and $color(\mathbf{k}_{i(-\alpha,\beta,\gamma)}) = color(\mathbf{k}_{i(\alpha,\beta,\gamma)})$ for $\alpha \in \{-1,1\}$ $\beta,\gamma \in \{-1,0,1\}$, we delete $\{\mathbf{k}_{i(\pm1,\beta,\gamma)}\}$ from $\mathcal{K}$, that is, the subanchors on the first and the third layers in 0-axis direction of the $3 \times 3 \times 3$ grid at $\mathbf{v}_i$ that match in color pattern are deleted. Then, this operation is

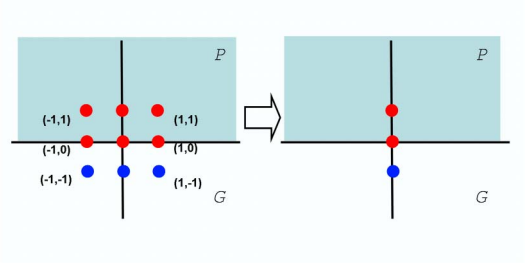


Fig. 7. Inserting subanchors to a boundary vertex. Red dots denote anchors inside P and blue ones are those in G. As the color patterns on the leftmost and rightmost grid layer match, all subanchors on both layers are removed.

performed similarly in the other directions. The intuition of this method is to generate C^0 continuities at the boundaries with as few subanchors as possible, in order to keep the smoothness along the other directions. An example is given in Fig. 7 in which subanchors are inserted at a boundary vertex on 2D mesh. After all the required subanchors are added at the interface between P and G, Algorithm 1 is then applied to generate a new set of blending functions and a new set of weights.

4.3.4 Other Anchor Insertions

Section 4.3.3 has resolved most violations against rules 1 and 2 arising from the blending functions that are associated with the master anchors close to the polycube boundary. Nevertheless, there are still other violations left. They can be categorized into four types as follows:

1. (See Fig. 8a) Ghost blending functions associated with subanchors violate rule 1. For example, the support region of the blending function associated with $\mathbf{k}_{i(1,0)}$ (the other index is omitted for conciseness reason) overlaps with P. A pair of anchors $\mathbf{k}_{a(1,1)}$ and $\mathbf{k}_{a(1,-1)}$ can be added to reduce the support region to the boundary while no further violations being introduced. The violation arising from $\mathbf{k}_{j(1,-1)}$ is treated in the same fashion. In the case of $\mathbf{k}_{k(1,1)}$, only one subanchor $\mathbf{k}_{k(1,-1)}$ is required to eliminate the violation.
2. (See Fig. 8b) Internal blending functions associated with subanchors violate rule 2. For example, removal of the ghost region and ghost anchors will change the shape of the blending function associated with $\mathbf{k}_{i(1,0)}$ because its knot vector goes into the ghost region. Similar to case 1, $\mathbf{k}_{a(1,1)}$ and $\mathbf{k}_{a(1,-1)}$ can be added to cut off the blending function from outside. Only one anchor insertion is necessary to resolve the violation arising from $\mathbf{k}_{j(1,-1)}$. Even though the new blending functions after refinement still covers nearby ghost region, they do not violate rule 2 anymore. This has been explained in Section 4.3.2 (Fig. 6a).
3. (See Fig. 8c. We illustrate four different cases together in Fig. 8c. Thus, we consider the existence of only one blue anchor each time.) Ghost blending functions near a convex corner of P violate rule 1. For example, in spite of its knot vectors being apart from P and any internal anchors, the blending function associated with $\mathbf{k}_{a(0,0)}$ still influences the internal corner region. To remedy this violation, two subanchors $\mathbf{k}_{c(1,0)}$ and $\mathbf{k}_{b(0,1)}$ are added to the extended surfaces of the convex corner, reducing the support region of the blending function centered at $\mathbf{k}_{a(0,0)}$ to separate it from P. Similarly, $\mathbf{k}_{c(1,0)}$ and $\mathbf{k}_{b(-1,1)}$ are added for $\mathbf{k}_{a(-1,0)}$, $\mathbf{k}_{c(1,-1)}$, and $\mathbf{k}_{b(0,1)}$ for $\mathbf{k}_{a(0,-1)}$ and $\mathbf{k}_{c(1,-1)}$ and $\mathbf{k}_{b(-1,1)}$ for $\mathbf{k}_{a(-1,-1)}$.
4. (See Fig. 8d. There are also four independent cases presented in Fig. 8d.) Internal blending functions near a concave corner of P violate rule 3. This type of violation is similar to case 3 except that the domain region and the ghost regions are interchanged and the purpose of eliminating this kind of violations is to ensure restricted boundary of P.

Once new subanchors are inserted, we apply the refinement algorithm given in Section 4.3.1 and obtain new sets of blending functions, weights, and anchors along with the updated T-lattice structures. Since extra anchors may be introduced by the refinement, we have to search for new violations and resolve them again. These two steps are repeated until no violation is found. We notice in our experiment that it only takes one or two iterations in practice to eliminate all violation cases. On the other hand, the proposed anchor insertion method is guaranteed to terminate due to the fact that no vertex is added during refinement, and there are only a finite number of subanchors that can be added to T. In the worst case, each cube of T turns into a small Bézier volume.

4.4 Generating RTP-Spline Function

By removing G and all ghost anchors from $\mathcal{K}$, we obtain an RTP-spline, which is a single-piece smooth function,

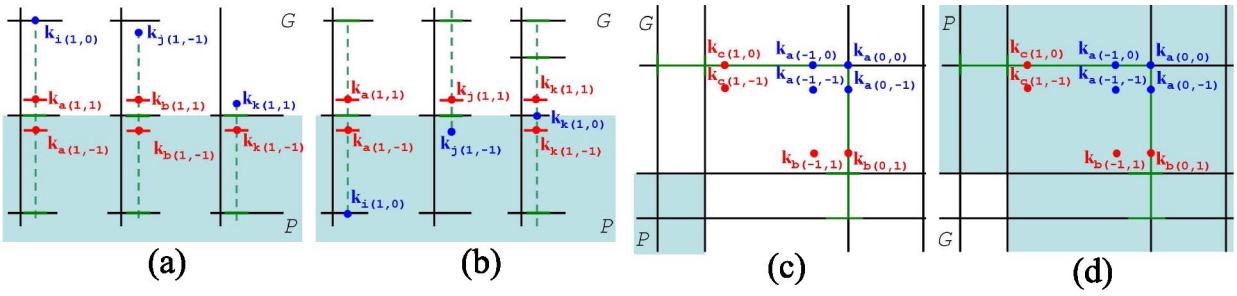


Fig. 8. Violation cases. The blue dots represent the associated anchors of violating blending functions. The anchors added to remedy the corresponding violations are colored red. (a) and (c) show the violation cases against rule 1; (b) shows the cases against rule 2; and (d) shows the cases against rule 3. Note that in (c), we illustrate four independent violation cases in one figure. In (d), we also show four independent cases.

defined over a polycube domain P. Our anchor insertion method guarantees that the resulting RTP-splines have a restricted boundary. Furthermore, the refinement algorithm proposed in Section 4.3.1 ensures semistandardness of the obtained RTP-splines from the original B-spline volume. Since the denominator remains 1 over the entire domain P, we can rewrite (6) in a simpler formulation:

$$\mathbf{F}(u,v,w) = \sum_{i=1}^{|\mathcal{B}|} w_i \mathbf{p}_i B_i(u,v,w) \quad (u,v,w) \in \mathbb{R}^3. \qquad (7)$$

5 Modeling Solid Objects

It is a challenging task to build single-piece and smooth spline representations for arbitrary solid objects, especially for those with bifurcations and high genus. This section addresses how to construct an RTP-spline for a given solid model. In this work, an input solid model is represented as a dense tetrahedral mesh $M = \{\mathcal{V}, \mathcal{T}\}$. Its geometry and other material attributes are discretely represented on vertices $\mathcal{V}$, and are interpolated linearly within each tetrahedron of $\mathcal{T}$. Note that our volumetric mapping algorithm is a meshless method with a closed-form mapping representation, and it works for other volumetric data representations such as point clouds and voxel grids. Therefore, the entire RTP spline construction pipeline can be easily generalized to handle other volumetric data formats.

We first construct a polycube P following the geometry and topology of M and compute a volumetric mapping $\mathbf{f}: P \to M$ (see Section 5.1), then construct an RTP-spline function $\mathbf{F}(u,v,w)$ over the polycube domain P (using the algorithm proposed in Section 4), and finally fit it to a group of data point chosen from M.

5.1 Parameterization on Polycube Domains

Computing volumetric parameterization is an important issue for the RTP-spline construction. Tensor-product trivariate splines usually need to be defined over a parametric (box) domain, and the quality of the parameterization can affect the fitting efficacy of splines. Therefore, we choose to use the polycube parametric domain which possesses great regularity while well approximating the geometry of the original object.

A volumetric parameterization of a solid model M embedded in $\mathbb{R}^3$ on a polycube P is a bijective mapping $\mathbf{f}: P \to M, P, M \subset \mathbb{R}^3$. The polycube P can be constructed either manually [19], [20], [29] or automatically [21], [22]. These techniques also provide the boundary mapping $\mathbf{g}$ from the polycube boundary surface (denoted as ∂P) to the boundary of M (∂M). We use such surface mapping $\mathbf{g}: \partial P \to \partial M$ as the boundary condition of $\mathbf{f}$. The volumetric parameterization is then defined as the seeking of a harmonic energy minimizer:

$$\begin{cases} \Delta \mathbf{f}(\mathbf{x}) = 0, & \mathbf{x} \in P, \\ \mathbf{f}(\mathbf{x}) = \mathbf{g}(\mathbf{x}), & \mathbf{x} \in \partial P, \end{cases}$$

where Δ is the 3D Laplace operator, defined for each real function f in $\mathbb{R}^3$ as

$$\Delta f = \nabla \cdot \nabla f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} + \frac{\partial^2 f}{\partial z^2}.$$

$\Delta \mathbf{f} = 0$ for $\mathbf{f} = (f^1, f^2, f^3)$ is equivalent to $\Delta f^i = 0$ in all the $i = 1, 2, 3$ coordinate directions. We compute the volumetric polycube mapping using the method of fundamental solutions (MFS) [24], [30]. We recap the basic algorithm here and refer more details to [24].

Based on the maximum principal of harmonic functions, critical points of harmonic functions exist only on the boundary. Furthermore, function values in the interior region of P are fully determined by the boundary values $f(\mathbf{x}), \mathbf{x} \in \partial P$ and can be computed by Green's functions. Specifically, the real harmonic function value $f(\mathbf{x})$ can be computed as the integration of its boundary values and the kernel function (i.e., fundamental solutions associated with the 3D Laplacian operator Δ). The kernel function of Δ has the following formula:

$$K(\mathbf{x}, \mathbf{x}') = \frac{1}{4\pi} \frac{1}{|\mathbf{x} - \mathbf{x}'|},$$

which matches the electrostatistics. In other words, solving a harmonic function can be converted to designing a specific electric field determined by an electronic particle system, whose electric potential mimics f and shall satisfy the boundary condition $\mathbf{g}$ on ∂P.

The computation pipeline is to first place a set of charge points $\{q_s\}$ outside the domain $q_s \in \partial \widetilde{P}, P \subset \widetilde{P} \subset \mathbb{R}^3$. Then, we conduct a boundary fitting which solves the charge distribution $\{w_s\}$ on these points $\{q_s\}$. The harmonic function $f(x)$ is represented using the MFS equation:

$$f(\mathbf{x}, W, Q) = \sum_{s=1}^{N_s} w_s \cdot K(\mathbf{x}, q_s), \mathbf{x} \in P, q_s \in \partial \widetilde{P},$$

where f is guaranteed harmonic, and we need to enforce the boundary condition on ∂P. For boundary fitting, we sample N_c collocation points on the domain boundary ∂P to set up the constraint equations. If we have N_s charge points and N_c collocation points, for a real harmonic function f (e.g., along an individual axis direction), we only need to solve an $Ax = b$ linear system where A is an $N_c \times N_s$ matrix. The system can be efficiently solved by a truncated singular value decomposition [24], [31].

The parameterization of a general solid model on its adaptive polycube domain can get lower distortion than that on a single box domain, since the polycube can be constructed to have the same topology and similar geometry as the model. In fact, in RTP-spline construction, the parameterization without fully satisfying the conformality and equivolume property does not have side effects to the volume fitting, as long as the overall parameterization mapping is continuous and smooth. Therefore, the current parameterization is efficient and sufficient, i.e., the shape (angle) distortion and volume distortion of our volumetric mapping are satisfactory.

Along the following two directions, we would like to also explore volumetric mapping techniques for parameterization with higher quality: 1) We can use more complicated/general parametric domains such as manifold domains (directly represented by tetrahedral meshes),

TABLE 2
Statistics of Volumetric Fitting

Models	Data Points	# Control Points	RMS Error $\times 10^{-3}$	Timing (seconds)
Bimba	35511	4543	1.20	31.21
Kitten	60144	3820	1.27	44.53
2-Torus	26384	2888	3.69	20.65
Hand	1502700	9035	0.554	1150
Head	472122	12880	0.291	422.4
Beethoven				
(1st level)	103361	1001	1.80	67.79
(2nd level)	103361	3283	1.34	80.78
(3rd level)	103361	14699	0.718	123.28

polytubes [29], and so forth, which may more flexibly approximate the shape and yield lower distortion. However, on such domains, it becomes more challenging to construct regular splines providing the same favorable features of RTP-splines. 2) The current volumetric mapping is fully determined by the boundary constraint, i.e., the\bigpolycube surface mapping [19]. We can reduce the distortion by conducting relaxation of boundary surface mapping [32], now driven by the volumetric mapping distortion. However, this makes the mapping computation a nonlinear optimization which is inefficient.

5.2 RTP-Spline Volume Fitting

Given $\mathbf{f} : P \rightarrow M$, we evenly select a group of points $U = \{\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_m\}$ from the polycube parametric domain p; hence, their counterparts in the real-world domain are $X = \{\mathbf{x}_i = \mathbf{f}(\mathbf{u}_i), i = 1, \ldots, m\}$. The problem of fitting the RTP-splines $\mathbf{F}(u, v, w)$ resorts to minimizing the following equation using U and X, with respect to control points $\mathbf{p}_i$

$$\sum_{i=1}^{m} (\mathbf{F}(\mathbf{u}_i) - \mathbf{x}_i)^2. \tag{8}$$

Alternatively, it can be represented in format of

$$\frac{1}{2}\mathbf{P}^T\mathbf{B}^T\mathbf{B}\mathbf{P} - \mathbf{X}^T\mathbf{B}\mathbf{P}, \tag{9}$$

in which $\mathbf{P}_j = \mathbf{p}_j^T$, $\mathbf{X}_i = \mathbf{x}_i^T$, and $\mathbf{B}_{ij} = \mathbf{I}_{3\times 3}B_i(\mathbf{u}_j)$. This is a typical least-squares problem, and we solve it for $\mathbf{P}$ using the optimization package *MOSEK* [33].

If the fitting results do not meet the requirement, we can improve them by refitting after adaptively subdividing cells where large fitting errors occur. Each cell from P is split into two, four, or eight smaller ones, depending on its aspect ratio. Once vertices, faces, and cells are added, Algorithm 1 is employed to refine existing RTP-spline and introduce additional degrees of freedom (DOFs) for better fitting. Note that Algorithm 1 is originally devised to work on a box domain, it can be however straightforwardly applied to RTP-splines defined on polycube domains, with a slight modification. That is, whenever a new boundary vertex is added, we have to insert a few subanchors in addition to the master anchor by following the way described in Section 4.3.3, in order to preserve the restricted boundary on the resulting RTP-splines.

Compared with the number of degrees of freedom in the optimization problem (8), U normally contains a much greater number of parametric points evenly distributed inside the polycube domain. So, the optimization problem is well posed and the resulting linear equations form a nondegenerate system. If there are too many subdivisions, the increased number of DOFs may lead to degenerate systems. In this case, we will first enlarge U by adding more points on the parametric domain near the regions where subdivisions take place and then recalculate X.

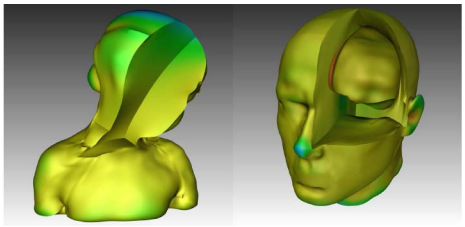


Fig. 9. RTP-spline function considered as a deformation function describing transformation from polycube to solid object. The Jacobians of the deformation gradients are colorized to illustrate smoothness of derivatives of the splines.

6 Results and Discussion

We implement the entire volumetric parameterization, RTP-spline construction, and data fitting framework in C++ and perform experiments on a 3 GHz Pentium-IV PC with 4 G RAM. Our experimental data include solid models of Bimba, Beethoven, eight (genus 2), kitten (genus 1), hand (five bifurcations), and head (with brain being excavated), which are represented as tetrahedral meshes. We successfully convert them into representations of single-pieced smooth RTP-splines using our proposed algorithms. The experimental results are given in Fig. 11.

The RTP-splines construction is efficient and usually takes only a few seconds, which consists of deducing knot vectors, building blending functions, calculating weights, and initializing necessary data structures. In all our experiments, this step takes at most 6 seconds (for the Beethoven model at level 3). In contrast, fitting RTP-splines to volumetric data sets is compositionally more expensive. The statistics of volumetric fitting are documented in Table 2, where the data points are parameterized on polycube domains, the fitting quality is measured by RMS errors, and the fitting errors are normalized by the overall sizes of solid models. From Table 2, we find that the volumetric fitting of the RTP-splines can be achieved efficiently and yield reasonable results. In addition, RTP-splines enable local subdivision of cells over desired regions to improve fitting qualities. As shown in the Beethoven model, the initial error is 1.80×10^{-3} without subdivisions and is reduced to 7.18×10^{-4} after two levels of subdivisions. The geometric details of Beethoven are also gradually revealed with the increasing level of subdivision (see Fig. 10).

RTP-spline is semistandard and hence computing blending functions and their derivatives on the domain is much more efficient than on traditional T-splines. To empirically prove this, we compared computational cost on the models Bimba, Kitten, and two-hole torus with both kinds of spline representations. To ensure the fairness in comparison, we

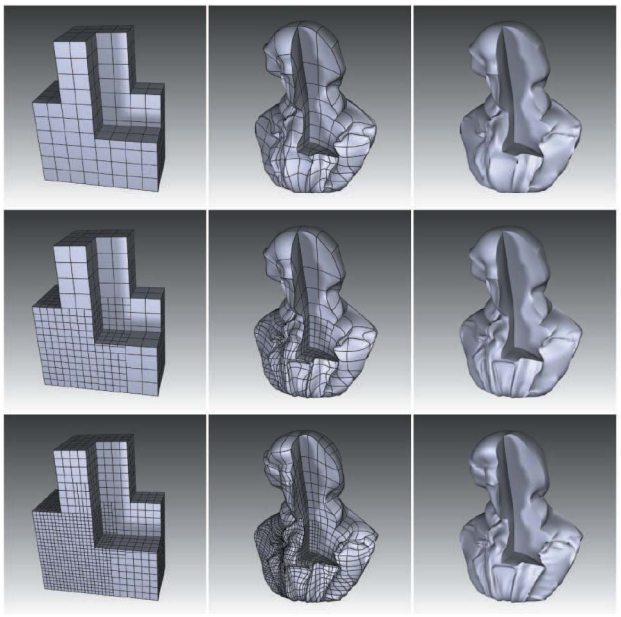


Fig. 10. Adaptive fitting of Beethoven. From top to bottom: fitting with 0, 1, and 2 levels of subdivision.

use the same source codes of RTP-splines to compute blending functions and derivatives for traditional T-splines, by including calculation of denominators. The comparison results are given in Table 3. As a result, the costs of the calculations of $\hat{B}$, $\hat{B}'$, and $\hat{B}''$ using traditional T-splines are roughly reduced by 47, 46, and 58 percent, respectively, if RTP-splines are used instead.

We can model other attributes in addition to geometry in RTP-splines by increasing the vector sizes of control points. In one of our experiments, we synthesize a scalar field on the head model, and then successfully recover a single RTP-spline representation of both the geometry and scalar values as shown in Fig. 12. Two kinds of scalar fields are involved in the experiment. One is the distance field to both the head surface and the brain surface inside (see Fig. 12d). The fitting results for the distance field and the corresponding fitting error map are demonstrated in Figs. 12e and 12f, respectively. Note that the fitting errors shown here are also normalized RMS errors as the distances are related to the model geometry. The other type of scalar field is a synthesized procedural 3D texture, generated using the fractal sum of Perlin noise [34] as $T(\mathbf{p}) = \sum_{i=1}^{4} \frac{1}{i} noise(i\mathbf{p}), \mathbf{p} \in \mathbb{R}^3$) (see Fig. 12g). In the experiment, the value of $T(\mathbf{p})$ varies from 0.8 to -1.33. And the absolute RMS fitting error to $T(\mathbf{p})$ is 7.3×10^{-4} (see Figs. 12h and 12i).

As an RTP-spline function is continuously and smoothly defined over a polycube, we can evaluate any properties that depend on function values and derivatives anywhere over the domain. If we interpret the RTP-spline function $\mathbf{F}$ obtained from data fitting as a deformation from a polycube to the shape of the fitted solid model, the deformation gradient tensor is $\mathbf{G} = \mathbf{F} \otimes \nabla$ and its Jacobian $det(\mathbf{G})$ measures the volume changes produced by the deformation. In Fig. 9, the Jacobian values for the hand and Bimba model are directly evaluated from function value and derivatives of $\mathbf{F}$ and there are no abrupt changes in color due to the smoothness and continuity of RTP-splines.

Linear independence of blending functions. It has been proven in [35] that linear independence is not a guaranteed property on general T-splines, and the property of linear independence can be inherited from T-splines with coarser T-meshes through refinement operations, as long as new anchors are added in certain ways.

The RTP-spline is an extension of the trivariate T-spline, but its construction does not follow the rules proposed in [35] when new anchors are inserted. So, the question of whether the blending functions of an RTP-spline are linearly independent remains open. For this reason in this paper, we call B_i (in (7)) a *blending function* instead of a *basis function*.

The volume fitting problem is solved in a least-squares system, which does not require the linear equations involved to be linearly independent. Therefore, the fitting results will not be adversely affected even if the RTP-spline blending functions are linear independent or not. In our experiment, we observed that if there is no subanchor introduced in the refinement step as discussed in Section 4.3.1, the blending functions of the RTP-splines thus constructed would remain linear independent. So, it is possible to have a modified RTP-spline construction scheme using adaptive subanchor insertion that guarantees the linear independence property. We will explore this direction in the near future.

7 Conclusion

In this paper, we have proposed the concept and construction algorithm of RTP-splines and presented an effective framework to transform volumetric data (both geometries and associated attributes of solid objects) into representation of RTP-splines. Because of the topological flexibility of the polycube domain, RTP-splines can naturally model solid objects with bifurcations and high genus as a single-piece smooth function with restricted boundary, while ensuring lower parameterization distortion in comparison with traditional splines defined over standard box domains.

TABLE 3
Computational Costs in Calculating Blending Functions and Their Derivatives: RTP-Splines versus General T-Splines

Model	Sample Points	Polycube Spline			General T-spline		
		$\hat{B}(u,v,w)$	$\hat{B}'(u,v,w)$	$\hat{B}''(u,v,w)$	$\hat{B}(u,v,w)$	$\hat{B}'(u,v,w)$	$\hat{B}''(u,v,w)$
Bimba	2512	0.18s	0.6s	1.12s	0.35s	1.14s	2.62s
Kitten	23076	1.61s	5.21s	9.59s	2.95s	9.75s	23.1s
2-Torus	9768	0.71s	2.42s	4.36s	1.37s	4.43s	10.2s

The time includes total computation spent on blending functions values and their derivatives at all sample points. The blending functions used in comparison are defined as $\hat{B}_i(u,v,w) = w_i B_i(u,v,w)$ for RTP-splines, and $\hat{B}_i(u,v,w) = w_i B_i(u,v,w) / \sum_{j=1}^{|\mathcal{B}|} w_j B_j(u,v,w)$ for T-splines, respectively.

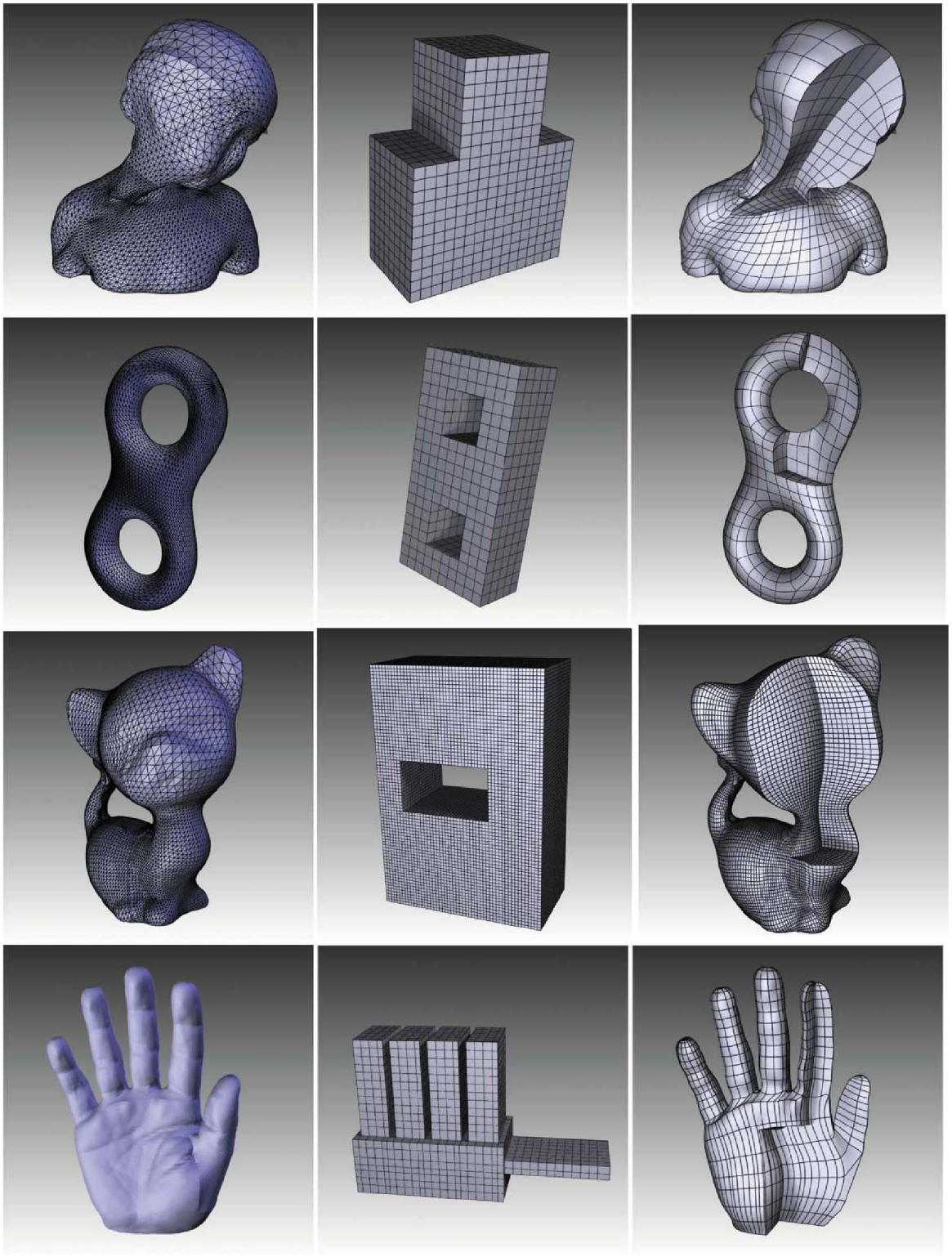


Fig. 11. From left to right: original solid models represented by tetrahedral meshes, polycube domains, and hex-mesh generated by RTP-spline functions. (The edges of the hand tetrahedral model are omitted without being displayed due to the fact that their significantly large number will deteriorate the rendering quality.)

The semistandardness on the initially constructed RTP-spline, which allows the efficient computation of spline functions and their derivatives, without any division overhead. The proposed RTP-spline supports local refinement, and a refinement algorithm has been developed to preserve the semistandardness when the RTP-splines undergo anchor insertion and local subdivision. The particular restricted boundary requirement of RTP-splines prevents control points from affecting domain regions spanning across nearby boundaries.

We demonstrate the efficacy of our RTP-splines as a powerful solid modeling tool in various experiments. This

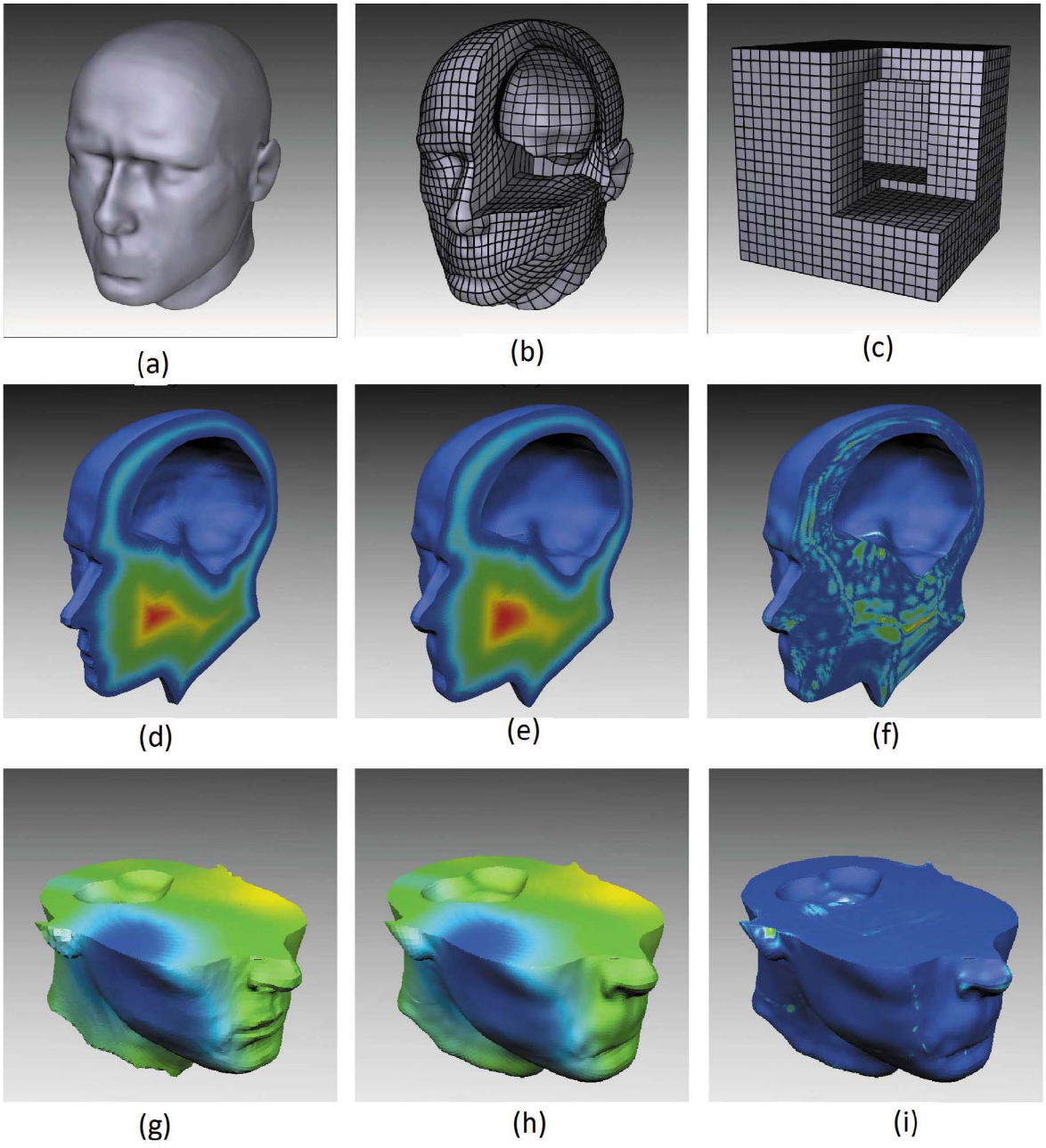


Fig. 12. Fitting results for the head model associated with synthesized scalar field (red denotes high value while blue denotes low value). (a) Polycube in parametric domain; (b,c) are the volumetric meshes reproduced from fitted RTP-splines; (d) synthesized distance field texture; (e) texture generated from the fitted RTP-spline; (f) fitting error map, where the maximum error is $0.92*10^{-2}$ and the average is $6.0*10^{-4}$; (g) texture synthesized as a Perlin noise function. In addition, (h,i) show the fitting errors and the error map, respectively. The maximum fitting error for noise texture is 0.066 and the average is $7.3*10^{-4}$.

unified paradigm enables the transformation from discrete solid models (represented by tetrahedral meshes) into continuous RTP-spline representations, accurately modeling both geometry and possibly multidimensional attributes.

At present, one unclear property of the RTP-spline is the linear independence of its basis functions, and we shall explore possible constraints during the RTP-spline construction and propose necessary or sufficient conditions that could guarantee such property. When the linear independence problem is solved, we would also like to explore the isogeometric analysis founded upon RTP-splines. Moreover, the particular polycube domains of RTP-splines can be naturally decomposed into a set of regular structures, which will enable GPU-friendly computing and image-based geometric shape processing. We are planning to investigate the aforementioned topics.

ACKNOWLEDGMENTS

This research is supported in part by US National Science Foundation (NSF) grants IIS-1047715, IIS-1049448, IIS-0949467, and IIS-0710819, Louisiana Board of Regents Research Competitiveness Subprogram (RCS) LEQSF(2009-12)-RD-A-06, PFund: NSF(2009)-PFund-133, and LSU Faculty Research Grant 2010.

REFERENCES

[1] T.W. Sederberg, D.L. Cardon, G.T. Finnigan, N.S. North, J. Zheng, and T. Lyche, "*T*-Spline Simplification and Local Refinement," *ACM Trans. Graphics,* vol. 23, no. 3, pp. 276-283, 2004.

[2] J. Hua, Y. He, and H. Qin, "Trivariate Simplex Splines for Inhomogeneous Solid Modeling in Engineering Design," *J. Computing and Information Science in Eng.,* vol. 5, no. 2, pp. 149-157, 2005.

[3] H. Si, "Tetgen: A Quality Tetrahedral Mesh Generator and Three-Dimensional Delaunay Triangulator," *Web Intelligence and Agent Systems: An Int'l J.,* 2005.

[4] X. Zhou and J. Lu, "Nurbs-Based Galerkin Method and Application to Skeletal Muscle Modeling," *Proc. ACM Symp. Solid and Physical Modeling,* pp. 71-78, 2005.

[5] T.J. Hughes, J.A. Cottrell, and Y. Bazilevs, "Isogeometric Analysis: CAD, Finite Elements, NURBS, Exact Geometry, and Mesh Refinement," *Computer Methods in Applied Mechanics and Eng.,* vol. 194, nos. 39-41, pp. 4135-4195, 2005.

[6] T. Martin, E. Cohen, and R.M. Kirby, "Volumetric Parameterization and Trivariate *B*-Spline Fitting Using Harmonic Functions," *Computer Aided Geometric Design,* vol. 26, no. 6, pp. 648-664, 2009.

[7] T.W. Sederberg, J. Zheng, A. Bakenov, and A. Nasri, "*T*-Splines and *T*-NURCCs," *ACM Trans. Graphics,* vol. 22, no. 3, pp. 477-484, 2003.

[8] B. Schmitt, A. Pasko, and C. Schlick, "Constructive Modeling of FRep Solids Using Spline Volumes," *Proc. Sixth ACM Symp. Solid Modeling and Applications,* pp. 321-322, 2001.

[9] W. Martin and E. Cohen, "Representation and Extraction of Volumetric Attributes Using Trivariate Splines," *Proc. Sixth ACM Symp. Solid Modeling and Applications,* pp. 234-240, 2001.

[10] D.L. Cardon, "*T*-Spline Simplification," master's thesis, Brigham Young Univ., 2007.

[11] H. Ipson, "*T*-Spline Merging," master's thesis, Brigham Young Univ., 2005.

[12] Y. Bazilevs, V. Calo, J. Cottrell, J. Evans, S. Lipton, M. Scott, and T. Sederberg, "Isogeometric Analysis Using *T*-Splines," *Computer Methods in Applied Mechanics and Eng.,* vol. 199, nos. 5-8, pp. 229-263, 2010.

[13] W. Song and X. Yang, "Free-Form Deformation with Weighted *T*-Spline," *The Visual Computer,* vol. 21, no. 3, pp. 139-151, 2005.

[14] R. Feichtinger, M. Fuchs, B. Jüttler, O. Scherzer, and H. Yang, "Dual Evolution of Planar Parametric Spline Curves and *T*-Spline Level Sets," *Computer-Aided Design,* vol. 40, no. 1, pp. 13-24, 2008.

[15] H. Yang and B. Jüttler, "Evolution of *T*-Spline Level Sets for Meshing Non-Uniformly Sampled and Incomplete Data," *The Visual Computer,* vol. 24, no. 6, pp. 435-448, 2008.

[16] H. Yang and B. Jüttler, "Meshing Non-Uniformly Sampled and Incomplete Data Based on Displaced *T*-Spline Level Sets," *Proc. IEEE Int'l Conf. Shape Modeling and Applications,* pp. 251-260, 2007.

[17] H. Yang and B. Jüttler, "3D Shape Metamorphosis Based on *T*-Spline Level Sets," *The Visual Computer,* vol. 23, no. 12, pp. 1015-1025, 2007.

[18] M. Tarini, K. Hormann, P. Cignoni, and C. Montani, "Polycube-Maps," *ACM Trans. Graphics,* vol. 23, no. 3, pp. 853-860, 2004.

[19] H. Wang, Y. He, X. Li, X. Gu, and H. Qin, "Polycube Splines," *Computer Aided Design,* vol. 40, no. 6, pp. 721-733, 2008.

[20] H. Wang, M. Jin, Y. He, X. Gu, and H. Qin, "User-Controllable Polycube Map for Manifold Spline Construction," *Proc. ACM Symp. Solid and Physical Modeling,* pp. 397-404, 2008.

[21] J. Lin, X. Jin, Z. Fan, and C. Wang, "Automatic Polycube-Maps," *Proc. Fifth Int'l Conf. Geometric Modeling and Processing,* pp. 3-16, 2008.

[22] Y. He, H. Wang, C.-W. Fu, and H. Qin, "A Divide-and-Conquer Approach for Automatic Polycube Map Construction," *Computers and Graphics,* vol. 33, no. 3, pp. 369-380, 2009.

[23] Y. Wang, X. Gu, T.F. Chan, P.M. Thompson, and S.T. Yau, "Volumetric Harmonic Brain Mapping," *Proc. IEEE Int'l Symp. Biomedical Imaging,* pp. 1275-1278, 2004.

[24] X. Li, X. Guo, H. Wang, Y. He, X. Gu, and H. Qin, "Meshless Harmonic Volumetric Mapping Using Fundamental Solution Methods," *IEEE Trans. Automation Science and Eng.,* vol. 6, no. 3, pp. 409-422, July 2009.

[25] T. Martin, E. Cohen, and M. Kirby, "Volumetric Parameterization and Trivariate B-Spline Fitting Using Harmonic Functions," *Proc. ACM Symp. Solid and Physical Modeling,* pp. 269-280, 2008.

[26] T. Martin and E. Cohen, "Volumetric Parameterization of Complex Objects by Respecting Multiple Materials," *Computers and Graphics,* vol. 34, no. 3, pp. 187-197, 2010.

[27] J. Xia, Y. He, X. Yin, S. Han, and X. Gu, "Direct-Product Volumetric Parameterization of Handlebodies via Harmonic Fields," *Proc. Int'l Conf. Shape Modeling,* pp. 3-12, 2010.

[28] J. Xia, Y. He, S. Han, C.-W. Fu, F. Luo, and X. Gu, "Parameterization of Star-Shaped Volumes Using Green's Functions," *Proc. Sixth Int'l Conf. Geometric Modeling and Processing,* pp. 219-235, 2010.

[29] C.-Y. Yao, H.-K. Chu, T. Ju, and T.-Y. Lee, "Compatible Quadrangulation by Sketching," *Computer Animation and Virtual Worlds,* vol. 20, nos. 2/3, pp. 101-109, 2009.

[30] X. Li, X. Guo, H. Wang, Y. He, X. Gu, and H. Qin, "Harmonic Volumetric Mapping for Solid Modeling Applications," *Proc. ACM Symp. Solid and Physical Modeling,* pp. 109-120, 2007.

[31] X. Li, H. Xu, S. Wan, Z. Yin, and W. Yu, "Feature-Aligned Harmonic Volumetric Mapping Using MFS," *Computers and Graphics,* vol. 34, no. 3, pp. 242-251, 2010.

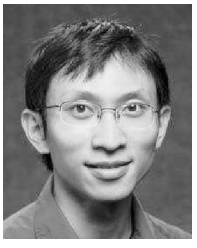
[32] X. Li, Y. Bao, X. Guo, M. Jin, X. Gu, and H. Qin, "Globally Optimal Surface Mapping for Surfaces with Arbitrary Topology," *IEEE Trans. Visualization and Computer Graphics,* vol. 14, no. 4, pp. 805-819, July 2008.

[33] Mosek, http://www.mosek.com/, 2011.

[34] K. Perlin, "An Image Synthesizer," *ACM SIGGRAPH Computer Graphics,* vol. 19, no. 3, pp. 287-296, 1985.

[35] A. Buffa, D. Cho, and G. Sangalli, "Linear Independence of the *T*-Spline Blending Functions Associated with Some Particular *T*-Meshes," *Computer Methods in Applied Mechanics and Eng.,* vol. 199, nos. 23/24, pp. 1437-1445, 2010.

Kexiang Wang received the BS and MS degrees in computer science from the University of Science and Technology of China in 1998 and 2001, respectively, and the PhD degree in computer science from Stony Brook University (SUNY) in 2010. He currently works as a data analyst at Renaissance Technologies, LLC.



Xin Li received the BS degree in computer science from the University of Science and Technology of China, and the MS and PhD degrees in computer science from Stony Brook University (SUNY). He is an assistant professor in the Department of Electrical and Computer Engineering and Center for Computational and Technology at Louisiana State University. His research interests include geometric modeling, computer graphics, vision, and visualization. His recent works include curve, surface and volumetric shape mapping/matching, and their applications in digital forensics, computational medicine, and robotics. He is a member of the IEEE Computer Society. More information about him can be found at http://www.ece.lsu.edu/xinli.

Bo Li received the BS degree in computer science from ZhongShan University (Sun Yet-sen University). He is working toward the PhD degree at the Department of Computer Science at Stony Brook University (SUNY). His research interests includes computer graphics, geometric modeling, computer-aided geometric design (CAGD), physical animation/simulation, and visualization. More information about him can be found at http://www.cs.stonybrook.edu/~bli.

Huanhuan Xu received the MSc degree from the University of Science Technology of China in 2006. She is working toward the PhD degree at the Department of Electrical and Computer Engineering at Louisiana State University. Her research interests include 3D computer graphics, geometric modeling, image processing, and pattern recognition.

Hong Qin received the BS and MS degrees in computer science from Peking University. He received the PhD degree in computer science from the University of Toronto. He is a professor of computer science in the Department of Computer Science at State University of New York at Stony Brook (Stony Brook University). In 1997, he was awarded NSF CAREER Award from the US National Science Foundation (NSF). He was also a recipient of Honda Initiation Award, and Alfred P. Sloan Research Fellow by the Sloan Foundation. He served as the general cochair for Computer Graphics International 2005. He was the conference cochair for ACM Solid and Physical Modeling Symposium in 2007. In 2008, he served as the conference chair for ACM Solid and Physical Modeling Symposium and IEEE International Conference on Shape Modeling and Applications. Currently, he is an associate editor for *Graphical Models*, *The Visual Computer*, and *Journal of Computer Science and Technology*. His research interests include geometric and solid modeling, graphics, physics-based modeling and simulation, computer-aided geometric design, human-computer interaction, visualization, and scientific computing. More information about him can be found at http://www.cs.sunysb.edu/~qin. He is a senior member of the IEEE.