Abstract—This paper develops a new trivariate hierarchical spline scheme for volumetric data representation. Unlike conventional spline formulations and techniques, our new framework is built upon a novel parametric domain called Generalized PolyCube (GPC), comprising a set of regular cubes being glued together. Compared with the conventional PolyCube (PC) that could serve as a “one-piece” 3-manifold domain, GPC has more powerful and flexible representation ability. We develop an effective framework that parameterizes a solid model onto a topologically equivalent GPC domain, and design a hierarchical fitting scheme based on trivariate T-splines. The entire data-spline-conversion modeling framework provides high-accuracy data fitting and greatly reduce the number of superfluous control points. It is a powerful toolkit with broader application appeal in shape modeling, engineering analysis, and reverse engineering.

Keywords—Trivariate Spline; Generalized PolyCube; Volumetric Parameterization.

I. INTRODUCTION

The engineering design industry frequently pursues data conversion from discrete 3D data to compact and continuous spline formulations in scientific computing and industrial applications (e.g., reverse engineering). Compared with surface splines, trivariate splines can represent both boundary shape and real volumetric physical/material attributes. This is vital and highly desirable in many physically-based applications including mechanical analysis [4], physically-based shape editing, virtual-surgery training, etc.

To model an arbitrary 3-D manifold using conventional trivariate splines, current approaches decompose the model to many simple solid primitives first, and then design trivariate spline representations for each sub-region respectively. Separate splines must glue together along shared boundaries in order to ensure continuity of certain degree. For models with non-trivial topology and complicated geometry, the entire partitioning and patching/gluing process is primarily performed manually, and it requires intensive labor from users with domain knowledge. To overcome this difficulty in modeling general data, we forge ahead with our research efforts in PolyCube mapping, and construct the trivariate splines over the volumetric Generalized PolyCube (GPC) domain. The solid PolyCube-shaped domain is regular and offers a cuboid structure. It has many advantages over other parametric domains (See Table II and Section IV for more discussions) and is ideal for trivariate spline construction.

In this paper, we design algorithms to construct trivariate T-splines over GPC for general volumetric data and demonstrate their efficacy as the global “one-piece” representation with hierarchical fitting capability.

Contribution and Overview. The main contributions of this work include:
(1) We develop an effective framework to compute the Generalized PolyCube (GPC) parameterization. Compared with the conventional PC, GPC is a more natural parametric domain to represent 3-manifolds with complicated topology.
(2) We present a global “one-piece” trivariate spline scheme without stitching/trimming for general volumetric models. GPC provides parametric representation for topologically-complex models using very few cubes.
(3) We design an efficient trivariate T-spline fitting algorithm, that supports hierarchical refinement with improved accuracy and reduced number of control points.

II. GENERALIZED POLYCUBE

A global one-piece parametric representation for shape with nontrivial topology is highly desirable for many geometric modeling and processing tasks, because it prevents artifacts caused by stitching/trimming over shared boundaries of solid primitives. The PolyCube representation is of particular interest toward this ambitious goal since it has perfect local regularity (i.e., local homogeneity) for tensor-product spline design. The idea of parameterizing a shape onto a conventional PolyCube (PC) is introduced by Tarini et al. [9]. A PolyCube is glued by a set of unit cubes. Therefore, consistent sets of knot intervals can be devised on a PolyCube straightforwardly. PolyCube parameterization has been studied by different researchers for various shape modeling tasks [9], [10]. In this work, we propose a novel concept called Generalized PolyCube (GPC), where the gluing direction of two faces shall be explicitly identified and a cube is even allowed to glue to itself (see Section II-A for details). Moreover, cube primitives are not enforced to have a true 3D embedding with principal axis alignment. In essence, a GPC domain consists of a set of cubes with connectivity information.

What inspires the design of GPC is that not all volumetric data can be parameterized effectively by PC. Figure 1 illustrates some examples. If we look at a solid torus ($S^1 \times D^2$) (a), topologically, the most concise representation is simply a cube with two opposite sides...
Conventional PC typically consists of cubes of uniform size, hence each cube face glues to another cube face in its entirety. This limits the valence of a cube cell to six. It is more flexible to allow the size of cubes to be non-uniform and a small cube can be glued to a part of a face of a big cube. Adjacent sub-regions with different solid volumes can then be parameterized to two cubes of different sizes, guaranteeing coherent parameterization. For example, in Fig. 2, cubes of fingers are partially glued to the cube of the palm. The parameterization and gluing are also clearly illustrated on the GPC-graph. In the interest of space, we refer interested readers to [5] for more implementation details on partial gluing.

C. GPC Construction and Parameterization

Given a solid model as a general input, the GPC-graph and GPC parameterization can be computed simultaneously in the following procedure:

1. **Topological Decomposition.** Partition the model to several components so that each component has trivial topology. A homology basis of a closed genus-g surface can be computed automatically [1], and the surface can be canonically decomposed into a set of $2g - 2$ pants patches [6]. The volumetric primitive bounded by each pants patch can be further decomposed into four topological cubes.

2. **Geometric Decomposition.** For each volumetric primitive, detect long and thin branches using the shape skeleton [11], then remove each long branch and parameterize it to a single cube domain.

3. **Cube Parameterization.** Parameterize each sub-region using a cube domain.

Intuitively, in Steps 1 and 2, each handle of the original model is parameterized to a “T”-shaped GPC, composed by four cubes. Long and thin branches (like antennae) are also parameterized to cube primitives separately. The algorithm of cube parameterization is as follows: (3.1) 8 vertices on the boundary are either selected or derived (from adjacent primitives to ensure the consistency), and they are mapped to the corners of the cube; shortest paths connecting pairs of corner vertices partition the boundary into 6 quadrilateral regions; (3.2) compute the harmonic surface map [2] for each quadrilateral region that can map to one of the cube’s faces, subject to all the curve boundary constraints; (3.3) solve the 3D Laplacian equation [12] to arrive at the interior volumetric parameterization.
III. Trivariate GPC-Splines

A. Point-Based Splines

Tensor-product trivariate B-splines are usually defined over the parametric cube domain. In order to allow hierarchical and adaptive fitting, without significantly increasing the number of control points, T-splines (that allows T-junctions for knots and control points) have been proposed [8]. T-splines are point-based splines whose control points form a T-mesh and have no regular connectivity with surrounding ones. In this paper, as a natural generalization of our work of designing bivariate T-splines on Polycube surface domain [10], our ambitious goal is to design the trivariate T-splines over GPC for general volumetric data. We shall highlight the idea of our spline construction by fitting $C^0$-continuous parametric solid, while generalizing it to globally $C^n$-continuous representation is straightforward.

Each control point $C_i$ (located in parametric cube $D_i^j$ with local coordinate $c^j_i$) is associated with three knot vectors along three principal axis directions: $r = [r_1, r_2, r_3, r_4, r_5], s = [s_1, s_2, s_3, s_4, s_5], t = [t_1, t_2, t_3, t_4, t_5]$, where $r_3 = 0$, $s_3 = 0$, and $t_3 = 0$. For any sample point with $(u, v, w)$ as its parameter, the blending function is

$$B_i(u, v, w) = N_r(u) \times N_s(v) \times N_t(w), \quad (1)$$

where $N_r$, $N_s$, and $N_t$ are cubic B-spline basis functions associated with the knot vector $r$, $s$, and $t$ respectively.

The formulation for a PB-spline on this point is

$$P(u, v, w) = \frac{\sum_i^n C_i B_i(u, v, w)}{\sum_i^n B_i(u, v, w)}. \quad (2)$$

By parameterizing the solid model to a GPC, PB-splines defined on cubes can achieve a globally continuous representation for any input model. The global parametric domain is equivalent to a collection of coordinate charts in all constituting local cube primitives via cube parameterization, and these local charts are then glued coherently to form the entire GPC parametric domain, enabled by the GPC-graph. As a result, the global PB-splines are piecewise rational polynomials defined on GPC, whose transition functions between adjacent cube primitives are compositions of translations and rotations of $n\pi/2$. Note that unlike the PolyCube surface splines, trivariate splines defined on GPC do not have singularities in solid interior.

Given an arbitrary parameter $u$ in cube $D^j_i$ (also denoted as $u^j_i$), the spline approximation can be carried out as follows:

1. Find all the neighboring cubes $\{D^j_i\}$ that support $u$ (i.e., it contains control points $C_k^j$ that may support $u$);
2. The spline function is:

$$P(u) = \frac{\sum_{k=0}^n C_k^j B_k(\phi^j_i(u^j_i) - c^j_k)}{\sum_{k=0}^n B_k(\phi^j_i(u^j_i) - c^j_k)} \quad (3)$$

where $u^j_i$ is the local parametric coordinate of point $u$ in the cube domain $D^j_i$, $\phi^j_i$ is the transition function from cube domain $D^j_i$ to $D^j_k$, $C_k^j$ denotes the control point $k$ in the cube domain $D^j_i$, and $c_k^j$ is its local coordinate.

The transition function $\phi^j_i$ from cube domains $D^j_i$ to $D^j_k$ is a composition of translations and rotations following the consecutive path where we glue cube $D^j_i$ to cube $D^j_k$. In the GPC-graph, if we assign the weight on each edge $[D^j_i, D^j_k]$ to be the length of the translation vector that transforms one local frame to the other, then this transition function can be easily derived by navigating through the shortest path $D^j_iD^j_k$ from node $D^j_i$ to $D^j_k$.

Suppose $\tilde{D}^j_iD^j_k := D^j_{i+1} \rightarrow D^j_{i} \rightarrow \ldots \rightarrow D^j_1(=D^j_{i})$, and the transition function $\Phi[i+1](t)$ (derived by way of cube-gluing) from $D_{i+1}$ to $D_i$ is known, then $\phi^j_i$ is formulated by

$$u^i = \phi^j_i(u^j_i) = \Phi[1,2](\Phi[2,3](\ldots \Phi[n-1,n](u^j_i))).$$

B. Hierarchical Fitting

A key advantage for defining T-splines upon GPC is that the evaluation of each node relies on just a union of general cubes in nearby regions. Moreover, the hierarchical fitting and level-of-detail control can be efficiently developed, which have two attractive properties: eliminating a large percentage of superfluous control points by introducing T-junctions; and providing adaptive control to users for properly balancing between fitting accuracy and efficiency.

The spline fitting is defined as follows. We have the parameterized solid model being generated from the boundary representation. The sample point in the model is $f(u_i)$, where $u_i$ is the parametric coordinate for each sample point. We minimize the following equation:

$$E_{dist} = \sum_{j=0}^n ||P(u_i) - f(u_i)||^2, \quad (4)$$

where $P(u_i)$ is the approximation of each sample point acquired from Eq. (3). Given a sample parametric point $u$ in GPC, we measure the root-mean-square error (rms) $\sigma(u)$ between its spatial position $f(u)$ and its spline approximation $P(u)$.

In each cube domain, we start from a coarse representation, denoted as T-mesh $H_0$, and assign a control point for each vertex. Then from level $k$ to level $k + 1$, a cell on $H_k$ is subdivided into 8 sub-cells in $H_{k+1}$ if the fitting error of this cell is larger than a given tolerance. The cell fitting error is the maximal error $\sigma(u)$ of sample points $u$ in this cell. Algorithm 1 offers a high-level sketch for the hierarchical fitting procedure.
Algorithm 1 Hierarchical Spline Fitting

for all cube domain $D_i$ do 
  Initialize $H_0$ and control points, $k = 0$
end for
loop
  1. Traverse and get knot vectors for all control points.
  2. Evaluate $P(u)$ in Eq.(3) for all sample points.
  3. Spline Fitting: Minimize $E_{	ext{dist}}$ in Eq.(4).
  4. Compute the fitting error of each cell:
     for all cells $H_j^k$ in level k do
       if Error$(H_j^k) \geq$ tolerance then
         Subdivide $H_j^k$ and add new control points.
       end if
     end for
     if No cell is subdivided then
       Stop.
     end if
     $k = k + 1$
end loop

C. Implementation based on Octree Structure

Traversing Knot Vectors. We use the method of “Ray-Traversing” [8] to generate 3 knot vectors for each control point. Unlike surface splines, enabling T-junctions in volumetric domains can result in much more complicated data structure and very time-consuming knot computation. We implement the octree data structure for efficient knots traversing. However, we restrict that a cube can only be divided to 8 sub-cells of equal size during refinement (note that, originally [8] allowed dividing a cell to 2 or 4 cells of different sizes, which becomes extremely complicated in 3D), because it greatly reduces the implementation complexity and improves the efficiency of knots traversing.

In Fig. 4, we only highlight our idea for efficient knots traversal in a 2D layout using quad-trees. This idea can be directly generalized to 3D using octrees. In a quad-tree, let L, R, T, and B denote the current cell that locates at the left, right, top, and bottom (in 3D, also front and back) w.r.t. its parent cell, respectively. In (a), suppose we traverse a ray starting from the red point (on cell $C_s$) to right, and need to know the cell $C_t$ that contains the yellow point on its right face (because once we have $C_t$, the knots vector can be computed directly). The first step is to locate the smallest common ancestor $C_{st}$ (or up to level-0, the GPC-graph) of cells $C_s$ and $C_t$, $C_s$ and its parent cells should be always on the “Right” of their parent cells. This is because once it becomes a “Left” cell, that parent cell is the smallest common ancestor $C_{st}$.

We demonstrate the efficacy of our framework by converting several solid models into trivariate spline representations. Fig. 5 shows the spline representation of a solid Femur. A $3 \times 3 \times 3$ control mesh and control points. (b) The control point cloud of the refined T-mesh. (c) The close-up view of a volumetric T-junction. The path from $C_s$ to $C_{st}$, as the red path shown in (b), can be encoded as a string $T_s = \{RT, RB, RT, RT\}$. The path from $C_{st}$ to $C_t$ can be efficiently traced by first reversing $T_s$ (i.e., $\{RT, RT, RB, RT\}$), then replacing all $R$ with $L$ (because we are traversing towards right), resulting in the encoding $T_t = \{LT, LT, LB, LT\}$. $T_t$ indicates the traversal from $C_{st}$ to $C_t$, plotted as the blue path in (b). More implementation details can be found in [5].

IV. EXPERIMENTAL RESULTS

Figure 5: Volumetric T-spline representation of the Femur model. (a) a $3 \times 3 \times 3$ control mesh and control points. (b) The control point cloud of the refined T-mesh. (c) The close-up view of a volumetric T-junction.

<table>
<thead>
<tr>
<th>Model</th>
<th>Bimba</th>
<th>Femur</th>
<th>Kitten</th>
<th>Greek</th>
<th>Homer</th>
</tr>
</thead>
<tbody>
<tr>
<td>S #</td>
<td>25,000</td>
<td>12,250</td>
<td>40,000</td>
<td>31,300</td>
<td>22,400</td>
</tr>
<tr>
<td>C #</td>
<td>1,905</td>
<td>1,430</td>
<td>2,304</td>
<td>4,365</td>
<td>3,070</td>
</tr>
<tr>
<td>rms (avg)</td>
<td>0.096%</td>
<td>0.077%</td>
<td>0.35%</td>
<td>0.53 %</td>
<td>0.21%</td>
</tr>
</tbody>
</table>

We demonstrate the efficacy of our framework by constructing several solid models into trivariate spline representations. Fig. 5 shows the spline representation of a solid Femur. A $3 \times 3 \times 3$ control grid is shown in (a), the refined control point cloud in the control grid is illustrated in (b), and a T-junction structure is shown in (c). Fig. 6 shows the spline representation of the solid Greek whose boundary surface is of genus-4. (a) shows the fitting result. (b) shows the hierarchical T-junctions on the model’s boundary surface, and (c) illustrates the side view from the right arm, highlighting the interior fitting result with T-junctions. Fig. 7 illustrates the T-spline representation of the solid Homer. (a) is the global fitting result, (b) offers a

Figure 6: Splines for the solid Greek. (a) The fitting result. (b-c) The close-up view of T-junctions on the boundary surface and at the solid interior.
close-up view of the head with T-junctions, and (c) shows its GPC-graph.

Fitting errors (root mean square errors after model normalization into a unit cube) on several models are color-coded and visualized in Fig. 8. The statistical results are given in Table I. In most of our experiments, approximation with good quality can be obtained after 3 levels of hierarchical refinement.

Table II: Comparisons of different parametric domains.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Regularity</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes, but Singularity</td>
<td>No</td>
</tr>
<tr>
<td>Connectivity</td>
<td>n</td>
<td>6</td>
<td>2</td>
<td>n</td>
</tr>
<tr>
<td>Hierarchical Refinement</td>
<td>Easy</td>
<td>Easy</td>
<td>Hard</td>
<td>Easy</td>
</tr>
<tr>
<td>Automatic Decomposition</td>
<td>Easy</td>
<td>Not Easy</td>
<td>Hard</td>
<td>Hard</td>
</tr>
</tbody>
</table>

A comparison among GPC and some other parametric domains is given in Table II. The grid structures of GPC, PC, and cylindrical domains are regular. In cylindrical parameterization [7], the center line has singularity, and in PC or GPC parameterization there is no singularity point in the solid interior. These three parameterizations have regular control nets, leading to simple and efficient evaluations. In contrast, simplicial complex splines are built upon triangular structure. Each cell of PC has up to six adjacent cells and cylindrical cell can have at most two adjacent cells, these factors limit the way of how different sub-regions can connect with each other and engender the technical challenge of designing effective segmentation on general models. Simplicial complex splines do not have such a limitation, and by allowing non-uniform cubes of different size, GPC also has the flexibility to glue many cubes. Simplicial complex supports local subdivision easily, while for regular structures such as cylinders, this is rather difficult. By allowing T-junctions, PC and GPC afford efficient hierarchical refinement.

V. CONCLUSION

We have presented a global “one-piece” trivariate hierarchical spline construction framework based on Generalized PolyCube (GPC) parameterization. The GPC concept enables a novel and desirable mechanism that facilitates the “one-piece” spline representation. We have articulated the decomposition and parameterization procedure. Global trivariate T-splines can be constructed on GPC and transition functions can be effectively computed using the GPC-graph. The entire spline construction framework affords hierarchical refinement and level-of-detail control. Our GPC trivariate T-splines have great potential in various shape design and physical analysis applications.

ACKNOWLEDGMENTS

This work is supported in part by NSF IIS-0949467, NSF IIS-0710819, Louisiana Board of Regents RCS LEQSF(2009-12)-RD-A-06, and PFund: NSF(2009)-PFUND-133.

REFERENCES