Trivariate solid T-spline construction from boundary triangulations with arbitrary genus topology

Wenyan Wang a, Yongjie Zhang a,*, Lei Liu a, Thomas J.R. Hughes b

a Department of Mechanical Engineering, Carnegie Mellon University, Pittsburgh, PA 15213, USA
b Institute for Computational Engineering and Sciences, The University of Texas at Austin, Austin, TX 78712, USA

A comprehensive scheme is described to construct rational trivariate solid T-splines from boundary triangulations with arbitrary topology. To extract the topology of the input geometry, we first compute a smooth harmonic scalar field defined over the mesh, and saddle points are extracted to determine the topology. By dealing with the saddle points, a polycube whose topology is equivalent to the input geometry is built, and it serves as the parametric domain for the trivariate T-spline. A polycube mapping is then used to build a one-to-one correspondence between the input triangulation and the polycube boundary. After that, we choose the deformed octree subdivision of the polycube as the initial T-mesh, and make it valid through pillowing, quality improvement and applying templates to handle extraordinary nodes and partial extraordinary nodes. The T-spline that is obtained is $C^2$-continuous everywhere over the boundary surface except for the local region surrounding polycube corner nodes. The efficiency and robustness of the presented technique are demonstrated with several applications in isogeometric analysis.

1. Introduction

For tight integration of Design-Through-Analysis, isogeometric analysis [1] was proposed which utilizes spline functions as a basis. The current state-of-the-art in engineering design and analysis is built on disparate geometric foundations. Spline representation is popular in design while polygonal mesh representation is generally used in analysis. This leads to many translational difficulties which affect the efficiency and accuracy of the entire process. Isogeometric analysis utilizes the same basis functions as the geometry representation, and, consequently, analysis can be carried out on an exact spline representation of the geometry. Similar to other physically based analyses, trivariate solid models, which can represent both the boundary shape and the interior volume, are required for many applications in isogeometric analysis. A fundamental step for the unified Design-Through-Analysis technologies is to automatically construct trivariate spline models from boundary representations.

The advent of T-splines [2] gives more flexibility for geometric modeling, allowing local refinement, non-rectangular domains in 2D and non-cubic domains in 3D. T-splines can represent a complicated design with complex topology as a single watertight geometry, avoiding splitting the model into several patches. T-splines are a superior alternative to and also are compatible with NURBS, which is the current geometry standard in CAD systems. The flexibility of T-splines makes them an ideal discretization technology for isogeometric analysis.

Several methods have been developed recently to construct trivariate T-splines. A trivariate T-spline generation method was described for genus-zero geometries [3]. In [4], trivariate T-splines were defined based on the generalized polycube parameterization. Another spline scheme based on polycubes, called restricted trivariate polycube splines, was developed in [5]. This algorithm is based on semi-standard T-splines. It requires calculation of weights, and the T-spline elements obtained are uniform. For all these methods, the T-spline model that is constructed may contain some negative Jacobian elements, which is unsuitable for analysis.

In our earlier work, we developed a mapping-based rational trivariate solid T-spline construction method for genus-zero geometry from the boundary surface triangulation [6]. To extend this algorithm to more general geometry, we first extract the topology of the input geometry and build a polycube which approximates the input geometry, by computing a harmonic field and dealing with its critical points. Due to its regular structure, the polycube is suitable for serving as the parametric domain of the tensor-product spline representations. Here, a trivariate solid T-spline is built directly upon the generated polycube, instead of being based on generalized polycube parameterization [4] or extending the polycube to a box domain and then restricting this...
domain [5]. We then build a parametric mapping between the triangulation and the boundary of the polycube. In the following steps, an octree subdivision is applied to the polycube and the initial T-mesh is a deformation of its subdivision. The subdivision continues until the surface approximation error is less than a given tolerance. After that, the valid T-mesh is obtained through pillowing, T-mesh quality improvement, and applying templates to handle extraordinary nodes and partial extraordinary nodes. Finally, Bézier elements are extracted with all positive Jacobians, which are suitable for isogeometric analysis.

Our main contribution lies in using polycubes to automatically construct analysis-suitable trivariate solid T-splines with arbitrary topology. Compared with other existing methods, our trivariate solid T-spline construction scheme has several attractive properties: (1) it is built directly upon the polycube and works for arbitrary topology, and the polycube is created automatically; (2) the trivariate T-spline obtained has a one-piece representation, and it contains very few irregular nodes where the continuity degenerates; (3) it employs rational T-spline basis, which guarantee partition of unity by definition; (4) it produces high-quality analysis-suitable T-spline elements with all positive Jacobians; and (5) with an adaptive refinement scheme, the resulting T-spline is an efficient representation for analysis.

The remainder of this paper is organized as follows. Section 2 reviews related work. Section 3 presents an overview of the construction algorithm. The polycube construction algorithm is presented in Section 4. The T-mesh construction algorithm from the polycube is described in Section 5, and trivariate T-spline construction is presented in Section 6. Section 7 presents examples, and Section 8 draws conclusions.

2. Previous work

Harmonic fields. A harmonic field is the solution to the Laplace equation with given boundary conditions, and it defines a scalar or vector field over the domain. For a given mesh, harmonic fields can be computed by solving a linear system of algebraic equations with imposed boundary conditions. Harmonic fields have certain desirable properties, such as smoothness, and are free of extraneous critical points. Due to these properties, harmonic fields have been shown to provide an effective tool for a number of geometry processing problems. Dong et al. [7] traced the integral lines through the gradient and orthogonal vector fields of a harmonic field for quadrilateral remeshing of arbitrary manifolds. Based on a harmonic map, a 3D geometric metamorphosis method was developed for any two objects which are topologically equivalent to a sphere or a disk [8]. Joshi et al. [9] utilized harmonic coordinates, which are generalized barycentric coordinates, in volume deformation.

Surface parameterization. A surface parameterization is a one-to-one mapping from one surface in three dimensions to a suitable planar domain. Parameterization is a powerful tool, and it is necessary for many geometry processing tasks, including data fitting, texture mapping, and remeshing. Many significant advances have been made for surface parameterization [10–12]. In [13], a conformal mapping method was presented to map a genus-zero closed surface onto the unit sphere by minimizing the harmonic energy of the map. For parameterization of an arbitrary genus object to simpler surfaces of the same genus, the mesh is usually first segmented into disk-like patches and then each patch is mapped onto the corresponding plane. In [14], Gu et al. solved the problem of global conformal parameterization for surfaces of arbitrary topology, with or without boundaries.

Polycube generation and application. A polycube is a solid composed of cubes. It can be used to very roughly approximate the geometry of a 3D object while faithfully replicating its topology. Due to its highly regular structure, the polycube can be used as the parametric domain for surface parameterization and spline modeling. However, in practice, due to the complexity of shapes, polycubes are usually constructed manually, entailing considerable effort. To produce polycubes with less user intervention, an automatic method was developed to construct polycube maps via a feature-based mesh segmentation [15]. He et al. [16] developed another method to construct a 3D polycube by extruding the axis-aligned polygons which approximate the horizontal curved intersection contours. Based on the polycube, a global parameterization technique, the polycube map [17], was first used for seamless texture mapping. Wang et al. developed a technique to build the polycube splines upon the polycube map for surface modeling [18]. In [4], an algorithm was developed to construct trivariate T-splines over generalized polycubes with a global “one-piece” representation for general volumetric data. In [5], a theoretical volumetric modeling framework was presented to construct restricted trivariate polycube splines, in which the blending functions are strictly bounded within the solid polycube domain.

Trivariate spline modeling. Only a few works have been devoted to trivariate solid spline modeling. In [19], a trivariate simplex spline modeling method was developed based on a tetrahedral decomposition of any 3D domain with complicated geometry and arbitrary topology. In [20], a skeleton-based method was developed to construct solid NURBS for isogeometric analysis of arterial blood flow. A method was presented in [21] to generate NURBS parameterizations of swept volumes by sweeping a closed curve, and isogeometric analysis was applied to the generated NURBS model. Based on discrete harmonic functions, a volumetric parameterization was used to construct a single trivariate B-spline [22]. By using adaptive tetrahedral meshing and a mesh untangling technique, an algorithm was developed to construct a trivariate T-spline representation of genus-zero solids [3].

It is still a challenging problem to automatically create polycubes for high-genus geometry and use them in constructing analysis-suitable trivariate T-splines. In this paper, we utilize a harmonic field defined over the input mesh to build the polycube automatically, and then construct the rational trivariate solid T-spline over the polycube. We include pillowing and quality improvement techniques to guarantee that the solid T-spline obtained can be used for analysis directly.

3. Algorithm overview

As shown in Fig. 1, there are three main stages for constructing a trivariate solid T-spline from a given boundary triangle mesh with arbitrary genus topology. First, we compute a harmonic scalar field defined over the mesh, extract the geometry topology, and then generate the polycube with the same topology. Adopting the polycube as the parametric domain, we build the valid T-mesh in the second stage and construct trivariate solid T-splines in the last stage.

The polycube generation stage consists of two steps.

(i) Harmonic field calculation — We build a smooth harmonic scalar field defined over the input mesh. Based on this field, we compute its gradient field and an orthogonal vector field.

(ii) Handling critical points — From the harmonic field, we determine all the critical points where the first partial derivatives vanish. These critical points include extreme points and saddle points. We design different methods to deal with each type of point in order to build the polycube. The polycube edges are traced along the gradient and isocontour directions.
Fig. 1. An overview of the trivariate solid T-spline construction algorithm from the given boundary triangle mesh with arbitrary topology.

Based on the polycube, the T-mesh is constructed. There are five different kinds of node in the solid T-mesh: regular nodes, partial extraordinary nodes, extraordinary nodes, edge T-junctions and face T-junctions. A regular node is a node around which the local T-mesh is a structured mesh, like node A in Fig. 2(a). Both partial extraordinary nodes and extraordinary nodes are irregular, and they can be distinguished using reflection edges. Reflection edges are a pair of adjacent edges with one common node, and the set formed by all the elements sharing one edge is topologically symmetric with the set of elements sharing the other. For example, AB and AC in Fig. 2(b) are a pair of reflection edges. A partial extraordinary node is an irregular node about which some but not all of its adjacent edges have reflection edges, like node A in Fig. 2(b). An extraordinary node is an irregular node about which none of its adjacent edges have reflection edges, such as node A in Fig. 2(c). A T-junction terminates a row of control points in one or more parametric directions, which may lie on an edge or a face. In solid T-splines, an edge T-junction is a T-junction which lies on one edge, such as node M in Fig. 2(d). A face T-junction is a T-junction lying on one face, such as node P in Fig. 2(e).

The T-mesh construction stage consists of four steps.

(i) Parametric mapping — We build a parametric mapping between the input triangle mesh and the boundary of the generated polycube.

(ii) Octree subdivision and projection — An initial T-mesh is obtained by an octree subdivision of the polycube and each node on the polycube boundary is projected onto the boundary surface based on the mapping.

(iii) Pillowing and quality improvement — We insert one pillowed layer on the boundary and improve the T-mesh quality by smoothing and optimization.

(iv) Handling extraordinary nodes and partial extraordinary nodes — In order to obtain a gap-free T-mesh, we apply templates to each extraordinary node and partial extraordinary node in the initial T-mesh.

In this stage, we use the polycube as the parametric domain for the trivariate solid T-spline construction. In the octree subdivision step, we choose the existing T-junction parametric values to subdivide each octree cell as much as possible, instead of always using the central parametric value. Based on the valid T-mesh, the knot vectors for each node are determined by traversing T-mesh faces and edges [23], and the trivariate T-spline is constructed based on the rational T-spline definition [24]. Bézier elements are extracted to serve as the primary computational objects in isogeometric analysis. See [25,26] for elaboration.

4. Polycube generation

In this section, we discuss the detailed algorithm of the polycube generation from the input boundary triangulation. The polycube must be constructed in a geometrically approximate and topologically equivalent way. To achieve this goal, we first compute a harmonic function defined over the input mesh \( \mathcal{T} \), derive two orthogonal fields from the harmonic function, and extract the topology structure of \( \mathcal{T} \) by examining critical points.

4.1. Harmonic field calculation

To extract the topological structure of the input mesh \( \mathcal{T} \subset \mathbb{R}^3 \), we construct a harmonic function \( f : \mathcal{T} \rightarrow \mathbb{R} \), such that

\[
\triangle f = 0
\]  

(1)

Fig. 2. Four types of node in solid T-meshes. (a) Regular node; (b) partial extraordinary node; (c) extraordinary node; (d) edge T-junction; and (e) face T-junction.
subject to the boundary condition that vertices in the predefined set \( S_{\text{min}} \) and \( S_{\text{max}} \) have the minimum and maximum values. \( \Delta \) is the Laplace operator, \( S_{\text{min}} \) and \( S_{\text{max}} \) are either given by the user, or they can also be determined by selecting the bottom-most and topmost points on \( \mathcal{T} \). Basically, computing such a harmonic function aims to assign a scalar value to each vertex in \( \mathcal{T} \). For a triangle mesh, we use the discretization of the Laplace operator

\[
\Delta f_i = \sum_{j \in N_i} w_{ij}(f(V_j) - f(V_i)) = 0, \tag{2}
\]

where \( V_i, V_j \in \mathcal{T} \), \( w_{ij} \) is the weight and \( N_i \) is the number of vertices adjacent to \( V_i \). Here, we choose the weights, \( w_{ij} = \cot \alpha_i + \cot \beta_i \), where \( \alpha_i \) and \( \beta_i \) are the opposite angles of the edge \( V_i - V_j \) [11].

From the discretization of the Laplace operator, the scalar function \( f \) can be obtained by solving a linear system. Fig. 3(a) shows the scalar function defined over the “Eight” model. The red region has the maximal scalar value and the blue region has the minimal scalar value.

From the scalar field \( f \), we compute both the gradient \( g_1 = \nabla f \) and one orthogonal vector field \( g_2 \) (the isocontour field). Due to the properties of a harmonic scalar field, the direction fields obtained are guaranteed to be smooth and free of extraneous critical points. For one triangle \( \{V_i, V_j, V_k\} \), suppose \( \vec{n} \) is the unit normal vector. The gradient vector \( g_1 = \nabla f \) is obtained by solving the following linear system [7]:

\[
\begin{bmatrix}
V_i - V_j \\
V_j - V_k \\
\vec{n}
\end{bmatrix}
\begin{bmatrix}
g_1 \\
g_2 \\
f_1 - f_2 \\
f_1 - f_3
\end{bmatrix} = \begin{bmatrix}
f_1 - f_1 \\
f_2 - f_2 \\
0
\end{bmatrix}. \tag{3}
\]

The field \( g_2 \) is along the isocontour directions of \( f \). Hence, for one scalar value, we simply find out its isocontour to obtain the vector field \( g_2 \) for the triangle mesh. Once we obtain the two orthogonal vector fields \( g_1 \) and \( g_2 \), we can trace along the flow lines. A flow line is a piecewise-linear curve defined over the mesh whose edges are along one of the vector fields. There are two cases for tracing the gradient flow: the regular case and the edge case. As shown in Fig. 3(b–c), vertex \( A \) is the starting point, the green arrows represent the gradient direction for each incident triangle, and we wish to trace the flow line from \( A \) by walking across the incident triangles along the gradient direction. For the regular case in (b), starting from \( A \) we extend the gradient line by crossing one of its incident triangles. In this case, we add one new vertex \( B \) to advance the flow line. For the edge case in (c), the flow field converges on an edge \( AB \) and we simply follow this edge. In Fig. 3(b–c), the red edges are the newly obtained flow line segments. The next step is to consider vertex \( B \) and trace the flow line from it using the same procedure.

In addition, based on the scalar field, we can then determine all the critical points of \( f \), that is, those points whose partial derivatives vanish. These points include the following:

- Minimal points — Points in set \( S_{\text{min}} \).
- Maximal points — Points in set \( S_{\text{max}} \).
- Splitting saddle points — Points where the geometry splits.
- Merging saddle points — Points where the geometry merges.

Fig. 4 shows the configurations of a regular point, a minimal point, a maximal point, and a saddle point. The red points denote the vertices with a larger scalar value compared with the center point, and the blue points denote the vertices with a smaller scalar value. As shown in Fig. 4(a), around one regular point, vertices on one side all have a larger scalar value and vertices on the other side all have a smaller scalar value. For a minimal/maximal point, all the vertices surrounding it have a larger or smaller scalar value. For a saddle point, the sign changes alternately along its circumferential direction.

Discussion. The harmonic field is controlled by the user-defined minima and maxima constraints; it then affects the polycube generation and alignment of the trivariate T-spline. To ensure that the harmonic field follows the geometric shape of the input surface, generally it is better to place these constraints on the tips of the geometry and also consider symmetry. For example, in the “Eight” model (Fig. 3(a)), we assign the constraints at the top and the bottom. Choosing minima and maxima without considering geometry symmetry may yield asymmetric parameterization results.

4.2. Handling critical points

We need to handle various critical points: minimal, maximal, splitting saddle and merging saddle points. We first compute all the scalar levels or isocontours at which there are critical points. Let \( f_i \) denote the isovalue of \( I_i \) and \( C_i \) denote the corresponding isocontour. For level \( I_i \) with saddle points, two sets of isocontours,
and $C^+_i$ are computed by using the isovalue $f_i$ with a small perturbation $\delta$. For the minimal or maximal level, only one isocontour is computed.

Suppose level $L_i$ contains one minimal point. At level $L_i$, four seed points $P^i_{l,j}$, $j = 0, \ldots, 3$ are chosen for each closed curve, as shown in Fig. 5(a–b), which correspond to the four lower corners of the cube $C_i$ with unit parametric length. The parametric value of the other vertices lying on this isocontour will be computed using the chord-length parameterization. From these seed points $P^i_{l,j}$ we trace the gradient flow line until it intersects with the isocontour $C^+_{i-1}$ of the next level $L_{i+1}$ at $P^i_{l,j}$ ($L_{i+1}$ may contain a saddle or maximal point). The red curves in Fig. 5 are traced gradient lines, and the black ones are isocontours. The four vertices $P^i_{l,j}$ $(j = 0, \ldots, 3)$ correspond to the four upper corners of $C_i$. The four traced curves are then mapped onto the four vertical edges of the cube, and $P^i_{l,j}$ and $P_{l,j}$ serve as the eight corners. Then the polycube construction process advances to the next level.

For level $L_i$ with a splitting saddle point $S$ as shown in Fig. 5(c–d), we do the following.

(i) Parameterize the lower isocontour $C^+_i$ using $P^i_{l,j} = P^i_{l,j-1}$ as four corners (assuming the associated cube before splitting is $C_{i-1}$).

(ii) Find the shortest path from the splitting saddle point $S$ to $C^+_i$, get the intersection node $Q_0$ on $C^+_i$, and calculate point $Q_1$ on the opposite edge with the same parametric value.

(iii) Compute the shortest path between $Q_0$ and $Q_1$ (see the blue curve in Fig. 5(c); the path cannot contain any edge on this isocontour).

(iv) Determine two sets of points on $C^+_i$, $Q_0^0 = Q_0^1$ and $Q_1^0 = Q_1^1$, which have the shortest distance from $Q_0$ to $Q_1$ to $C^+_i$.

(v) Construct two cubes $C^0_i$ and $C^1_i$ by using $P^0_{l,j-1}$-$Q_0$-$Q_1$-$P^1_{l,j}$ and $Q_0$-$P^1_{l,j}$-$Q_1$ as the lower corners, respectively.

(vi) Continue tracing the gradient flow until the flow line intersects the isocontour at the next level.

Basically, for one splitting saddle point, we aim to find one isoparametric line to split one cube into two. Here, the isoparametric line connecting $Q_0$ and $Q_1$ is used to split the upper face of the cube $C_{i-1}$, as shown in Fig. 5(c–d). Similarly, for each merging saddle point, we use the same procedure to find one isoparametric line to merge two neighboring cubes and ensure they match with each other seamlessly.

By dealing with all the critical points, the polycube is constructed level by level. We always map the isocontour onto the horizontal isoparametric edges of the polycube and map the gradient flow lines onto the vertical edges. Fig. 6 shows one polycube construction result for the “Eight” model. The red lines in Fig. 6(b) denote the curves corresponding to the edges of the polycube. This polycube construction algorithm does not consider the symmetry property of the input geometry. However, if $S_{\min}$ and $S_{\max}$ are given symmetrically, the polycube obtained and the trivariate T-spline that is constructed can be symmetric for a symmetric input geometry.

Discussion. Here, we only consider Morse saddle points whose multiplicity is 1. Morse saddle points are handled by splitting one cube into two, or merging two cubes into one. In general, for saddle points of any multiplicity, one cube may be split into an arbitrary number of cubes. If there are two or more saddle points on one level, multiple sets of $Q_0$–$Q_1$ need to be computed.

5. T-mesh construction

The T-mesh construction stage aims to build one valid T-mesh from the given boundary triangulation and the constructed polycube. There are four main steps in this stage: parametric mapping, adaptive octree subdivision and projection, pillowing and quality improvement, and handling extraordinary and partial extraordinary nodes.

5.1. Parametric mapping

This step aims to build a one-to-one parametric mapping between the input triangle mesh $T$ and the boundary of the polycube obtained, which serves as the parametric domain of the trivariate T-spline. From the constructed polycube, we have the correspondence between the traced gradient or isocontour lines and the edges of the polycube. Based on the traced lines on the triangle mesh, we can divide the input mesh into $N$ submeshes, $T^i (i = 0, \ldots, N)$, where $N$ is the number of boundary faces of the polycube. Each submesh is associated with one face of the polycube, $F^i_C (i = 0, \ldots, N)$. We then use the surface parameterization to map each submesh $T^i$ to its corresponding polycube face $F^i_C$.
For each submesh, we first map its boundary to the boundary of $F_i^l$ by a chord length parameterization. The parameterization for the interior vertices is calculated by solving a linear system formed by the harmonic equation $\sum_{j=0}^{3} w_{ij} (f(V_j) - f(V_i)) = 0$, where $w_{ij} = \cot \alpha_{ij} + \cot \beta_{ij}$. For the curve shared by two adjacent submeshes, we use the same parameterization. Fig. 6(c) shows the mapping result for the “Eight” model. To guarantee a conformal boundary between two neighboring cubes, we always choose the same parameterization for all the edges shared by them. Note that the two neighboring cubes $A$ and $B$ do not share faces. There are two duplicated faces in the parametric space but they are separate in the physical space.

5.2. Adaptive octree subdivision and projection

An initial T-mesh is generated by applying an adaptive octree subdivision to the polycube $C$ and projecting to the boundary. For each cube in $C$, we create one hexahedral element, using the same parametric length and considering the physical length difference in three directions. Then we obtain a root T-mesh for the whole polycube and treat it as one single piece, instead of treating each cube separately. Starting from the root T-mesh, we subdivide one element into eight smaller ones recursively to get the refined initial T-mesh after projection. For each boundary element, we check the local distance from the T-mesh boundary to the input triangular mesh, and subdivide the element if the distance is greater than a given threshold $\epsilon$.

Each node obtained has both parametric and physical coordinates. The parametric coordinates represent its position in $C$. For each boundary node, the physical coordinates are its associated position on the triangular mesh based on the polycube mapping. The physical coordinates of each interior node are calculated by linear interpolation. Note that the three isoparametric planes are not always in the middle. If one element contains T-junctions, the parametric values of the T-junctions are used to subdivide the element. If there are more than two T-junction parametric values in one direction, the one closest to the middle is used. For example, the purple element in Fig. 7 has one T-junction on the $\xi$-edge and two T-junctions on the $\eta$-edge. For the $\xi$ direction, the parametric value $\xi_1$ is used, while for the $\eta$ direction $\eta_2$ is used during subdivision, because it is closer to the parametric middle. The dashed lines
are inserted to refine this element. In this way we can minimize the number of T-junctions in the initial T-mesh. The octree subdivision and projection processes continue until the local distance from each boundary element to the input mesh is less than $\varepsilon$ and each element has at most one edge T-junction for an edge, or one face T-junction for a face. Fig. 8 gives one result. (a) shows the parametric coordinates for the nodes in the T-mesh and (b) shows their physical coordinates.

5.3. Pillowing and quality improvement

To improve the T-mesh quality, we adopt pillowing, smoothing and optimization techniques. Pillowing is a sheet-insertion technique that inserts one layer around the boundary [6]. Here, we insert one pillowed layer for the initial T-mesh, which helps to improve the T-mesh quality and the T-spline surface continuity.

Fig. 9 illustrates the pillowing operation for the polycube. The cubes $C_i$ are rendered in different colors. As mentioned earlier, $C_1$ and $C_2$ do not share a face in the parametric space. In (b), we use two separate faces. The dark blue lines show the pillowed layer. In pillowing, each boundary face is duplicated to form one pillowed element, and each pillowed element has at most one face lying on the boundary. For the pillowed layer, the edge knot interval is a predefined constant along the pillowing direction, which stays the same for the other two directions.

As shown in Fig. 9(b), after pillowing the yellow corner nodes become interior extraordinary nodes. The nodes on the blue polycube edges become interior partial extraordinary nodes. The red corner nodes, pillowed from the yellow nodes, are extraordinary nodes on the boundary surface. The T-spline surface that is constructed is $C^0$-continuous around these red boundary extraordinary nodes up to the 2-ring neighborhood, $C^1$-continuous from the 2-ring to the 3-ring neighborhood, and $C^2$-continuous...
Fig. 13. Kitten model with genus one. (a) The input boundary triangle mesh; (b) the harmonic scalar field (the bottom plane is chosen as the minima and the two ear tips are chosen as the maxima); (c) the constructed polycube and the mapping result; (d) the subdivision result for the parametric domain; (e) the constructed trivariate solid T-spline and T-mesh; (f) the extracted solid Béziers; (g) the extracted solid Bézier mesh with some elements removed to show the interior (blue) and one pillowed layer (magenta); and (h) the isogeometric analysis result. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

everywhere else. Note that, after pillowing, the surface continuity across the polycube edges is improved from \( C^0 \) to \( C^2 \). The interior region is \( C^0 \)-continuous around each interior extraordinary node until the 3-ring neighborhood. For the interior region across the polycube edges, the continuity is \( C^1 \) from the 2-ring to the 3-ring neighborhood, and \( C^2 \) everywhere else. Since here we introduce some extraordinary and partial extraordinary nodes, we use a local parameterization for each element in the following steps.

After pillowing, smoothing and optimization [6] are used to improve the T-mesh quality. For smoothing, each node is moved toward its mass center, and for optimization each node is moved toward an optimal position that maximizes the worst Jacobian. For one T-mesh element, the Jacobian is defined as

\[
J = \det(J_M) = \left| \begin{array}{ccc}
\sum_{i=0}^{7} x_i \frac{\partial N_i}{\partial \xi} & \sum_{i=0}^{7} x_i \frac{\partial N_i}{\partial \eta} & \sum_{i=0}^{7} x_i \frac{\partial N_i}{\partial \zeta} \\
\sum_{i=0}^{7} y_i \frac{\partial N_i}{\partial \xi} & \sum_{i=0}^{7} y_i \frac{\partial N_i}{\partial \eta} & \sum_{i=0}^{7} y_i \frac{\partial N_i}{\partial \zeta} \\
\sum_{i=0}^{7} z_i \frac{\partial N_i}{\partial \xi} & \sum_{i=0}^{7} z_i \frac{\partial N_i}{\partial \eta} & \sum_{i=0}^{7} z_i \frac{\partial N_i}{\partial \zeta}
\end{array} \right|,
\]

where \( N_i \) is a trilinear shape function. The scaled Jacobian is

\[
J_s = \frac{J}{\|J_M(\cdot, 0)\| \|J_M(\cdot, 1)\| \|J_M(\cdot, 2)\|},
\]

where \( J_M(\cdot, 0), J_M(\cdot, 1) \) and \( J_M(\cdot, 2) \) represent the first, second and last column of the Jacobian matrix, \( J_M \), respectively. To handle T-junctions during smoothing and optimization, we add some “virtual nodes” to refine the local region and convert the local T-mesh to a hexahedral mesh. Fig. 8(d) shows the result after pillowing and optimization for the “Eight” model.

5.4. Handling extraordinary and partial extraordinary nodes

Unlike regular nodes, extraordinary nodes or partial extraordinary nodes may introduce gaps to the trivariate T-spline. Similar to local refinement, we apply templates given in [24] to make the T-mesh gap free. Fig. 10(a) shows the template for a partial extraordinary node, in which the magenta edge has a reflection edge. Fig. 10(b) shows one general template for an extraordinary node.

6. Trivariate T-spline construction and Bézier extraction

In this stage, we aim to infer the local knot vectors for each node, build the rational trivariate solid T-spline from the T-mesh, and then extract embedded Bézier elements [25,26]. The concept of rational T-splines was given in [24], with basis functions satisfying a partition of unity by definition. The rational trivariate solid T-spline is defined as

\[
S(\xi, \eta, \zeta) = \frac{\sum_{i=0}^{n} w_i \bar{R}_i(\xi, \eta, \zeta)}{\sum_{i=0}^{n} w_i R_i(\xi, \eta, \zeta)}, \quad (\xi, \eta, \zeta) \in \Omega,
\]
Fig. 14. Sculpture model with genus two. (a) The input boundary triangle mesh; (b) the harmonic scalar field (the bottom plane is chosen as the minima and the two top-most vertices of the heads are chosen as the maxima); (c) the constructed polycube and the mapping result; (d) the subdivision result for the parametric domain; (e) the constructed trivariate solid T-spline and T-mesh; (f) the extracted solid Bézier elements; (g) the extracted solid Bézier mesh with some elements removed to show the interior (blue) and one pillowed layer (magenta); and (h) the isogeometric analysis result. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

\[
R_i(\xi, \eta, \zeta) = \frac{N_\xi^i(\xi)N_\eta^i(\eta)N_\zeta^i(\zeta)}{\sum_{j=0}^n N_\xi^j(\xi)N_\eta^j(\eta)N_\zeta^j(\xi)}
\]

(7)

where \(R_i(\xi, \eta, \zeta)\) is the rational B-spline basis function; \(N_\xi^i, N_\eta^i\) and \(N_\zeta^i\) are B-spline basis functions defined by the local knot vectors at node \(C_i\) which, for degree \(d = 3\), are given by \(\xi_i = [\xi_{i0}, \xi_{i1}, \xi_{i2}, \xi_{i3}, \xi_{i4}]\) and \(\eta_i = [\eta_{i0}, \eta_{i1}, \eta_{i2}, \eta_{i3}, \eta_{i4}]\) and \(\zeta_i = [\zeta_{i0}, \zeta_{i1}, \zeta_{i2}, \zeta_{i3}, \zeta_{i4}]\).

From the T-mesh, we first compute the knot intervals for each node by traversing T-mesh edges and faces. Note that we repeat knots whenever we meet an extraordinary node during the traverse. Then, for each domain, we determine all the nodes with non-zero basis functions and use them to build the solid T-spline element. The entire solid T-spline model is built by looping over all the local domains.

\[
B^e_t = M^e B^e_b.
\]

(8)

where \(B^e_t\) is the vector formed by the non-zero T-spline basis functions in this element and \(B^e_b\) is the vector formed by the trivariate Bézier basis functions.

7. Results and isogeometric analysis

We have applied the construction algorithm to several models (Figs. 11–14). The trivariate T-spline that is constructed is tricubic.
and C²-continuous except in the vicinity of partial extraordinary and extraordinary nodes. Statistics for all the tested models are shown in Table 1. The Bézier Jacobian is calculated using the scaled Jacobian at the eight Gauss quadrature points for each Bézier element. The algorithm is efficient, and all the results were computed on a PC equipped with an Intel X3470 processor and 8 GB main memory.

The Isis model has genus zero, and its polycurve only contains one single cube. For the kitten model, there are three saddle points (one merging and two splitting saddle points). We only consider two of them, and the splitting saddle point close to the maximum point is skipped to get a simplified polycurve. The sculpture model has genus two and five saddle points. Again, we skipped the one near the maximal level. The warrior model has genus four and eight saddle points. All of them are considered. The time used for each model not only depends on the input mesh size, but also depends on the T-mesh size, which is determined by the given surface error tolerance and the complexity of the topology and geometry. One advantage of this harmonic-function-based T-spline construction is that the isoparametric lines are basically aligned with the geometric structure of the model.

We have developed a 3D isogeometric analysis solver for static mechanics analysis, which uses rational T-splines as the basis, and we used it to test the trivariate T-spline models obtained. For all the models, we fix all the control points on the bottom and apply unit displacement on the top. The Young’s modulus $E = 72.4$ GPa, and the Poisson’s ratio $\nu = 0.3$. The displacement results obtained are given in Figs. 11(c), 12(d, m), 13(h) and 14(h). From these results, we can conclude that the rational T-splines obtained can be used in isogeometric analysis directly, and reasonable simulation results can be obtained.

8. Conclusions

We have presented a new algorithm to construct trivariate solid T-splines for arbitrary genus geometries from boundary triangulations. Our method is efficient, and the resulting trivariate T-spline is analysis suitable with C²-continuity except for the local region around a few irregular nodes. In this paper, we only consider geometries with Morse saddle points, and as part of our future work we will extend the algorithm to geometries with arbitrary saddle points. We will also consider engineering designs with sharp features.

Acknowledgments

Y. Zhang, W. Wang and L. Liu were supported by Y. Zhang’s ONR-YIP award N00014-10-1-0698 and an ONR Grant N00014-08-1-0653. T.J.R. Hughes was supported by ONR Grant N00014-08-1-0992, NSF GOALI CM-0700807/0700204, NSF CMMI-1101007 and a grant from SINTEF. The “Eight” model is from the AIM@SHAPE Shape Repository.

References


Table 1

<table>
<thead>
<tr>
<th>Model</th>
<th>Genus</th>
<th>Input mesh (vertices, elements)</th>
<th>T-mesh nodes</th>
<th>Extraordinary nodes (boundary, interior)</th>
<th>Interior partial extraordinary nodes</th>
<th>Bézier elements</th>
<th>Bézier Jacobian (worst, best)</th>
<th>Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Isis 0</td>
<td>0</td>
<td>(5,863, 11,722)</td>
<td>9,310</td>
<td>(8, 8)</td>
<td>244</td>
<td>5,335</td>
<td>(0.12, 1.00)</td>
<td>52.0</td>
</tr>
<tr>
<td>Kitten 1</td>
<td>1</td>
<td>(5,377, 10,754)</td>
<td>4,825</td>
<td>(12, 12)</td>
<td>140</td>
<td>2,883</td>
<td>(0.08, 1.00)</td>
<td>14.8</td>
</tr>
<tr>
<td>Eight 2</td>
<td>2</td>
<td>(766, 1,536)</td>
<td>5,735</td>
<td>(16, 16)</td>
<td>200</td>
<td>1,440</td>
<td>(0.10, 1.00)</td>
<td>8.5</td>
</tr>
<tr>
<td>Sculpture 2</td>
<td>(8,635, 17,276)</td>
<td>10,549</td>
<td>(16, 16)</td>
<td>252</td>
<td>7,072</td>
<td>(0.09, 1.00)</td>
<td>41.5</td>
<td></td>
</tr>
<tr>
<td>Warrior 4</td>
<td>4</td>
<td>(8,733, 17,478)</td>
<td>12,739</td>
<td>(24, 24)</td>
<td>309</td>
<td>6,835</td>
<td>(0.01, 1.00)</td>
<td>35.8</td>
</tr>
</tbody>
</table>