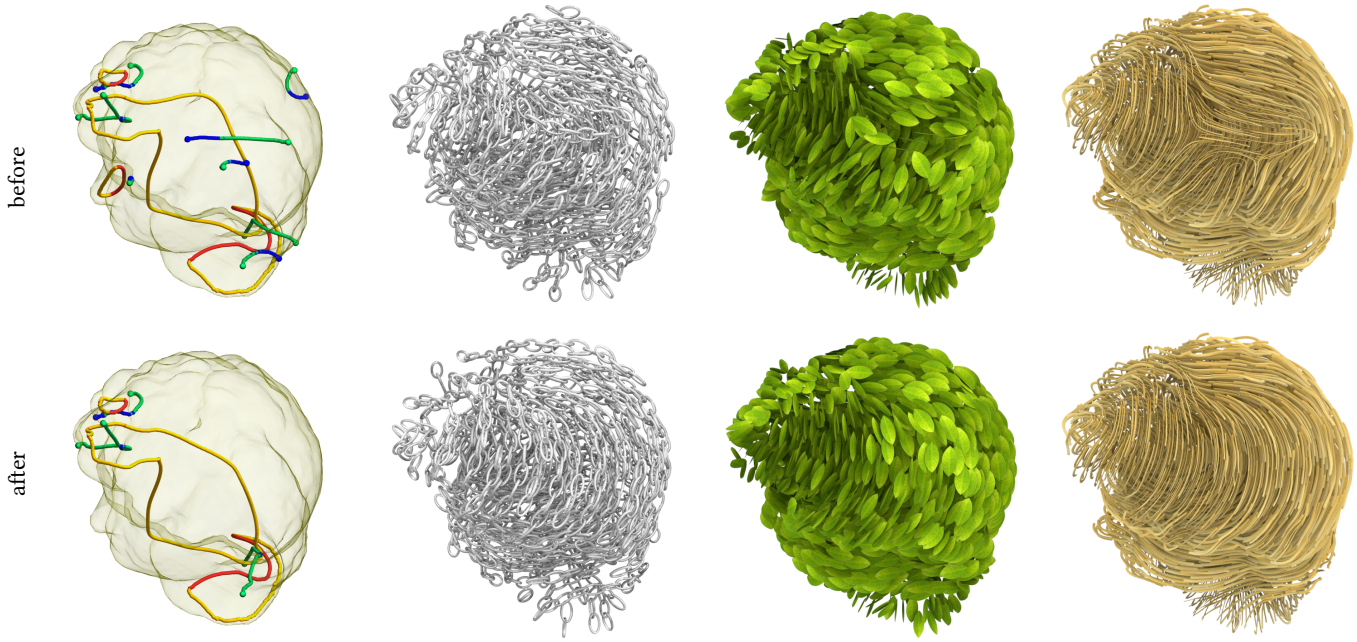
Tensor Field Design in Volumes

JONATHAN PALACIOS, LAWRENCE ROY, and PRASHANT KUMAR, Oregon State University
CHEN-YUAN HSU, Bournemouth University
WEIKAI CHEN, University of Hong Kong and USC Institute for Creative Technologies
CHONGYANG MA, Snap Inc.
LI-YI WEI, University of Hong Kong
EUGENE ZHANG, Oregon State University



(a) degenerate curves (b) chains (c) leaves (d) noodles

Fig. 1. *Different tensor fields can lead to different element synthesis results.* With our tensor field design system, users can create any tensor field and control the topology of the tensor field by deforming, removing, and reconnecting degenerate curves in the field (colored curves in (a)). The modified tensor field topology can lead to improved element synthesis results (before the topological editing (top) and after (bottom) in (b), (c), and (d)). The underlying 3D model is David's head, which is viewed from the top.

3D tensor field design is important in several graphics applications such as procedural noise, solid texturing, and geometry synthesis. Different fields can lead to different visual effects. The topology of a tensor field, such as degenerate tensors, can cause artifacts in these applications. Existing 2D tensor field design systems cannot be used to handle the topology of a 3D tensor field. In this paper, we present to our knowledge the first 3D tensor field design system. At the core of our system is the ability to edit the topology of tensor fields. We demonstrate the power of our design system with applications in solid texturing and geometry synthesis.

CCS Concepts: • **Computing methodologies** → *Texturing*; *Parametric curve and surface models*;

Additional Key Words and Phrases: 3D tensor fields, tensor field design, topology, texture synthesis, geometry synthesis, element synthesis

ACM Reference format:
Jonathan Palacios, Lawrence Roy, Prashant Kumar, Chen-Yuan Hsu, Weikai Chen, Chongyang Ma, Li-Yi Wei, and Eugene Zhang. 2017. Tensor Field Design in Volumes. *ACM Trans. Graph.* 36, 6, Article 188 (November 2017), 15 pages.
https://doi.org/10.1145/3130800.3130844

0730-0301/2017/11-ART188 $15.00
https://doi.org/10.1145/3130800.3130844

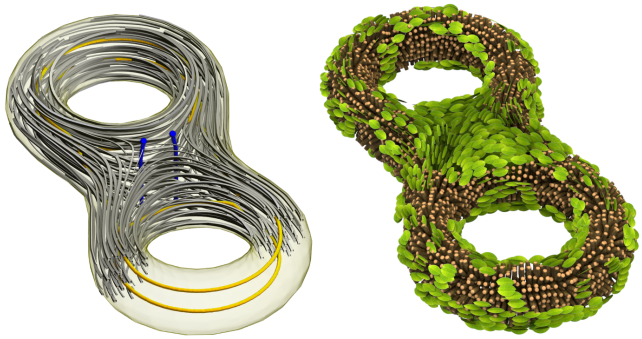


Fig. 2. *Tensor field design naturally leads to line field design and frame field design.* Given the tensor field designed by our system for a double torus (left: yellow and blue curves are degenerate curves and grey curves are hyperstreamlines following the minor eigenvector field), the tensor field can be used to generate element synthesis (right) with line-type elements (wood sticks) and elements with box-type symmetry (leaves). Moreover, a tensor field has both linear regions and planar regions (Section 3), naturally enabling two different elements to be synthesized with the same field. These properties are *not* shared by cross-frame fields.

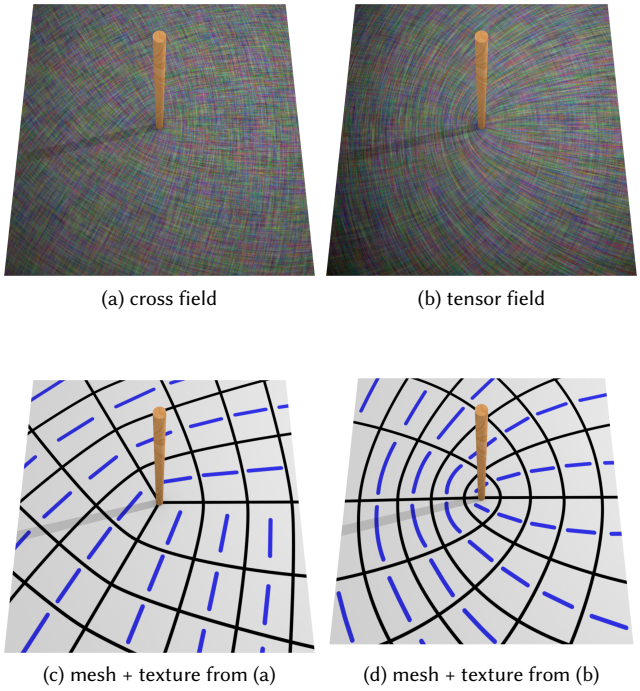


Fig. 3. *This figure compares the difference in the types of first-order singularities in a cross-frame field (left) and a tensor field (right).* Both fields have a line of singularities (top row), and we show the pattern of the fields projected on a plane perpendicular to the singularity line. The projection of the cross-frame fields on the plane is a 4-RoSy field (a), while the projection of the tensor field is a 2D tensor field (b) (both major and minor eigenvector fields shown). Notice that a cross-frame field singularity has an index of $\frac{k}{4}$ ($k \in \mathbb{Z}$), while a tensor field singularity has an index of $\frac{k}{2}$ ($k \in \mathbb{Z}$). This is because cross-frames have a larger symmetry group than tensors. In quadrangular remeshing (c)-(d) (mesh elements highlighted by the black curves), valence 3 and 5 vertices cannot be modeled by a tensor field singularity, making 2D tensor fields inadequate for quadrangular remeshing. Similarly, 3D tensor fields are inadequate for hexahedral remeshing. On the other hand, for line type of objects (c)-(d) (blue stripes), 4-RoSy fields can lead to visual artifact (c), while 2D tensor fields do not have such artifacts (d).

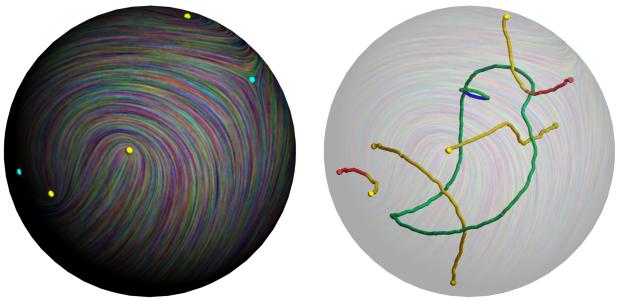
1 INTRODUCTION

Many applications rely on the ability to locally model anisotropy inside a volume, such as texturing [Kopf et al. 2007; Lagae and Dretakis 2011], meshing [Ray et al. 2016], modeling [Ma et al. 2011], and deformation [von Funck et al. 2006]. In these applications, the quality of the results heavily depends on the properties of the guiding anisotropy field, such as continuity and smoothness. Thus, field design has been a key focus in computer graphics and geometry processing [Vaxman et al. 2016].

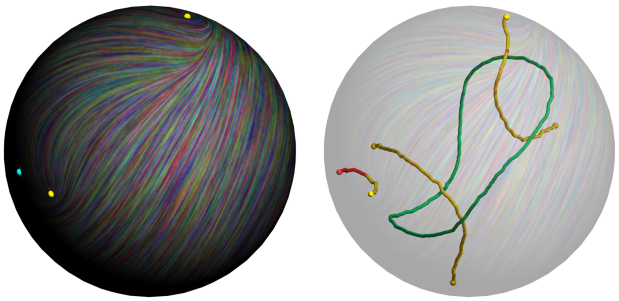
There are various types of volumetric fields, each with different symmetry properties and consequently suitable for different applications. For example, cross-frame fields (with octahedral symmetry) are ideal for hexahedral remeshing [Nieser et al. 2011] as they are able to properly model fundamental singularities (Figure 3a). On the other hand, the design and processing of cross frame fields is inadequate for the line-type and box-type of objects (e.g. Figure 3c). In contrast, tensor fields can model such objects (Figure 3d and Figure 2 – leaves (box-type) and wood sticks (line-type)). This is because a 3D tensor contains not only directional information (eigenvectors) but also anisotropic sizing information (eigenvalues). Moreover, due to anisotropy, a tensor field can be considered as a collection of three mutually perpendicular line fields. In contrast, such a distinction is not available for cross-frame fields. Consequently, cross-frame field processing does not apply to line field design, while tensor field design naturally leads to line field design. Moreover, a tensor field naturally contains a linear region and a planar region (Section 3), which enables the use of two different elements in the same field (see Figure 2). A cross-frame field does not have this capability. On the other hand, tensor field singularities do not properly model basic irregular vertices and edges in hex meshes (Figure 3). Consequently, tensor field design is inadequate for hexehedral remeshing.

We address the problem of designing volumetric tensor fields. To our knowledge, there has been little work on 3D tensor field design. Existing work on the processing of cross-frame fields in hexahedral remeshing [Huang et al. 2011; Li et al. 2012; Nieser et al. 2011] does not naturally extend to the processing of tensor fields. Furthermore, existing work on cross-frame fields mostly focuses on the automatic generation and modification of a cross-frame field based on the boundary of the domain, while we strive for a user design system for flexibility. The user can generate any tensor field, whether aligned with the boundary or not.

A major challenge facing tensor field design is the existence of *singularities*, where the tensor field has repeating eigenvalues and the orthonormal frames have discontinuity. Similar to 2D tensor field design, the set of singularities in a 3D tensor field plays an important role in shaping the behavior of the tensor field. Unlike 2D tensor fields, where the singularities (referred to as degenerate points) are isolated points under structurally stable conditions, in



(a) a surface tensor field (b) volume tensor field for (a)



(c) singularity pair cancellation of (a) (d) volume tensor field for (c)

Fig. 4. *The topology of a surface tensor field (a) does not provide a complete picture of the corresponding volume tensor field (b).* The 2D tensor field in (a) contains eight degenerate points on the boundary surface (yellow and cyan points). However, as a 3D tensor field in (b) there are two degenerate loops completely in the interior of the volume and thus not visible from the surface field in (a) Notice that it is difficult to predict which surface degenerate points belong to the same degenerate curves. Also, there can be degenerate loops which do not intersect the boundary surface. In addition, topological editing operations applied on the boundary tensor field can lead to unpredicted behaviors in the topology of the corresponding volume tensor field. For example, when removing a singularity pair in the surface tensor field (from (a) to (c)), two degenerate curves in the volume tensor field are joined, one of the degenerate loops is eliminated, and the shape of the other degenerate loop is altered (compare (b) with (d)).

3D tensor fields, degenerate points form curves (*degenerate curves*). Such curves can intersect the domain or reside inside the volume and form degenerate loops (Figure 4).

While it is conceivable to control the topology of a volume tensor field by controlling its behavior on the boundary surface, thus reusing 2D tensor field editing operations from existing research, this strategy can lead to undesired effects due to the following reasons. The topology of the surface tensor field does not provide a complete picture of the topology of the corresponding 3D tensor field. Thus when simplifying the topology of the 3D tensor field by 2D tensor field topological editing [Zhang et al. 2007], unpredicted changes may occur. An example is shown in Figure 4. Identifying possible, fundamental operations to modify the topology of 3D tensor fields, i.e., only targeted degenerate curves are impacted, is a challenging yet important problem for 3D tensor field design.

In this paper, we present to our knowledge the first interactive, 3D tensor field design system. The user can create a tensor field by specifying desired tensor values and local patterns inside the volume, or on the boundary surface, or in both places. The field can be made boundary-conforming, i.e., the surface normal direction is aligned with one of the eigenvectors of the tensor field everywhere on the surface.

At the core of our system is a set of topological editing operations such as *degenerate curve removal*, *degenerate curve deformation*, and *degenerate curve reconnection*, which we have identified and used to control the number, location, and shape of degenerate curves. We also provide robust algorithms to accomplish these degenerate curve editing operations. Figure 1 shows the results of geometry synthesis guided by a tensor field with degenerate points.

We demonstrate the efficacy of our system by applying it to the control of anisotropic Gabor noise, data-driven solid texturing, and geometry element synthesis.

2 PREVIOUS WORK

There has been much work in the design of orientations fields in both 2D and 3D, and it is beyond the scope of this work to thoroughly review all of them. Instead, we will focus on work that is most relevant to this research and refer interested readers to a recently published survey [Vaxman et al. 2016] for a comprehensive review.

Vector field design on surfaces starts with applications in texture and geometry synthesis [Nieser et al. 2012; Praun et al. 2000; Turk 2001; Wei and Levoy 2001], fluid simulation [Stam 2003], and non-photorealistic rendering [Hertzmann 1998; Hertzmann and Zorin 2000]. Zhang et al. [2006] present a vector field design system in which they use *Conley index theory* to control the number and location of the singularities in the vector field. Fisher et al. [2007] use discrete exterior calculus to design a vector field with control over the singularities in the field. Palacios and Zhang [2007] address the more generic problem of designing 2D orientations under N-way rotational symmetries, which they refer to as N-RoSy's. Ray et al. [2008] also develop a design system for such orientation fields under rotational symmetries with a focus on the control of location and type of the singularities in the field. They later automate this process to generate a field whose directions conform to the geometry of the surface [Ray et al. 2009]. Bommes et al. [2009] integrate field generation and quadrangular remeshing into the same system which they solve by a mixed-integer solver.

The analysis and design of 3D orientation field is a relatively new topic. Applications of 3D orientation fields include solid texture and geometry synthesis [Bénard et al. 2010; Kopf et al. 2007; Lagae and Drettakis 2011; Lagae et al. 2009; Landes et al. 2013; Ma et al. 2011; Takayama et al. 2008; Wei et al. 2009; Zhang et al. 2011] and hexahedral remeshing [Gregson et al. 2011; Huang et al. 2011; Lévy and Liu 2010; Li et al. 2012; Nieser et al. 2011]. However, it is difficult to specify and control the topology of the orientation fields, such as the number, location, shape, type, and connectivity of degenerate features [Ray et al. 2016; Solomon et al. 2017].

One exception is 3D symmetric tensor fields, for which there has been some work on topological analysis. Delmarcelle and Hesselink extend the notion of degenerate points from 2D tensor fields to 3D tensor fields [Delmarcelle and Hesselink 1994; Hesselink et al.

1997]. Zheng et al. [2004; 2005b] point out that numerically stable topological features in a 3D tensor field form curves and provide three methods to extract them. They further point out that tensor patterns near a degenerate point on a degenerate curve exhibits 2D tensor degenerate patterns such as wedges and trisectors [Zheng et al. 2005a]. Palacios et al. [2016b] introduce the notions of *eigenvalue manifold* and *neutral surfaces* into 3D symmetric tensor field topology.

In this paper we focus on the design of 3D tensor fields, which naturally leads to line field design and frame field design (Section 1). We have identified a set of editing operations that allows the topology of a 3D tensor field to be modified in an isolated fashion, i.e., without impacting other singularities. To do so, we rely on a number of observations on 3D tensor field topology that have not been made or reported previously. We also develop a unified framework in which these editing operations can be carried out. A preliminary version of this work is described in [Palacios et al. 2016a].

3 BACKGROUND

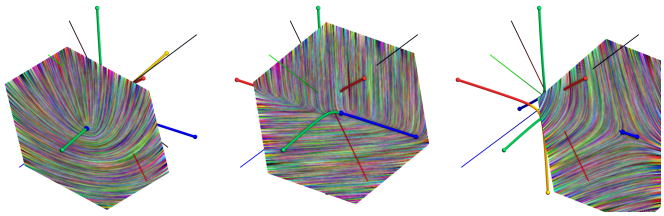
In this section we review background on 3D symmetric tensor fields mostly relevant to this work. This section is based on published works [Zhang et al. 2007; Zheng and Pang 2004]. As this paper *only* deals with symmetric tensors, we will omit the word *symmetric*.

A 3D (symmetric) tensor $\mathbf{T}$ has three real-valued *eigenvalues*: $\lambda_1 \geq \lambda_2 \geq \lambda_3$. They are referred to as the *major*, *medium*, and *minor* eigenvalues, respectively. An eigenvector belonging to the major eigenvalue is referred to as a *major eigenvector*. Medium and minor eigenvectors can be defined similarly. Eigenvectors belonging to different eigenvalues are mutually perpendicular.

The trace of a tensor $\mathbf{T}$ is $trace(\mathbf{T}) = \sum_{i=1}^{K} \lambda_i$. $\mathbf{T}$ can be uniquely decomposed as $\mathbf{D} + \mathbf{A}$ where $\mathbf{D} = \frac{trace(\mathbf{T})}{K}\mathbb{I}$ ($\mathbb{I}$ is the K-dimensional identity matrix) and $\mathbf{A} = \mathbf{T} - \mathbf{D}$. The *deviator* $\mathbf{A}$ is a *traceless* tensor, i.e., $trace(\mathbf{A}) = 0$. Note that $\mathbf{T}$ and $\mathbf{A}$ have the same set of eigenvectors. Consequently, the *topology* of a tensor field can be defined in terms of the topology of its deviator tensor field. Another nice property of the set of traceless tensors is that it is closed under matrix addition and scalar multiplication, making it a linear subspace of the set of tensors. Given these considerations, we will focus on the analysis and design of traceless tensor fields and also omit the term *traceless* in the remainder of the paper.

When $K = 3$, a tensor can be classified as either *linear* (**L**), *planar* (**P**), or *neutral* (**N**), corresponding to $\lambda_1 - \lambda_2 > \lambda_2 - \lambda_3$, $\lambda_1 - \lambda_2 < \lambda_2 - \lambda_3$, and $\lambda_1 - \lambda_2 = \lambda_2 - \lambda_3$, respectively. For traceless tensors, the above conditions are equivalent to $\lambda_2 < 0$ (linear), $\lambda_2 > 0$ (planar), and $\lambda_2 = 0$ (neutral).

A tensor is said to be *degenerate* if it has repeating eigenvalues. When $K = 2$, the only degenerate (traceless) tensor is the zero matrix. When $K = 3$, there are three types of degenerate tensors: neutral ($\lambda_1 = \lambda_2 = \lambda_3 = 0$), linear ($\lambda_1 > \lambda_2 = \lambda_3$), and planar ($\lambda_1 = \lambda_2 > \lambda_3$). The neutral degeneracy is also referred to as the *triple degeneracy*, for which all non-zero vectors are an eigenvector. The linear and planar degeneracies are called *double degeneracies*. The non-repeating eigenvalue is referred to as the *non-degenerate eigenvalue*, while the repeating eigenvalues are referred to as the *degenerate eigenvalues*. The non-degenerate eigenvalue is the major eigenvalue for **L**-type degenerate tensors and the minor eigenvalue for the **P**-type degenerate tensors. The eigenvectors corresponding to the non-degenerate eigenvalues are referred to as *non-degenerate eigenvectors*. The plane perpendicular to a non-degenerate eigenvector is referred to as a *degenerate plane*. Any vector inside this plane is an eigenvector corresponding to the degenerate eigenvalue.



(a) wedge (b) transition (c) trisector

Fig. 5. *The projection of a tensor field onto the non-repeating planes.* Along a degenerate curve, the projection can exhibit 2D degenerate patterns such as a wedge (left) and a trisector (right). Between segments of wedges (green) and trisectors (blue), transition points can appear (middle).

We now discuss tensor fields and their topology. A *tensor field* $\mathbf{T}(p)$ ($p \in \Omega$) is a tensor-valued function defined over some domain $\Omega \subset \mathbb{R}^K$. A tensor field can be thought of as K eigenvector fields, corresponding to the K eigenvalues. A point $p_0 \in \Omega$ is a *degenerate point* if $\mathbf{T}(p_0)$ is degenerate. The topology of a tensor field consists of its degenerate points.

In 2D, the set of degenerate points of a tensor field are isolated points under numerically stable configurations, i.e., when the topology does not change given sufficiently small perturbation in the tensor field. An isolated degenerate point can be measured by its *tensor index* [Zhang et al. 2007], defined in terms of the *winding number* of one of the eigenvector fields on a loop surrounding the degenerate point. The most fundamental types of degenerate points are *wedges* and *trisectors*, with a tensor index of $\frac{1}{2}$ and $-\frac{1}{2}$, respectively. The total tensor index of a continuous tensor field over a two-dimensional manifold is equal to the *Euler characteristic* of the underlying manifold. Consequently, it is not possible to remove one degenerate point. Instead, a pair of degenerate points with opposing tensor indexes (a wedge and trisector pair) must be removed simultaneously [Zhang et al. 2007].

In 3D, the situation is more complicated. Zheng et al. [2004; 2005a] point out that stable topological features consist of degenerate curves, each of which is either linear or planar. The projection of the tensor field onto the degenerate plane exhibits either a wedge or trisector pattern. Consequently, a degenerate curve can be divided into consecutive segments of purely wedge points and trisector points. The boundary points between these segments are *transition points*, where the non-degenerate eigenvector is perpendicular to the tangent of the degenerate curve. Figure 5 illustrates this with an example. Along degenerate curves, we can observe two 2D degenerate patterns such as wedges (left) and trisectors (right). At the transition point (middle), the projected pattern is neither a wedge nor a trisector.

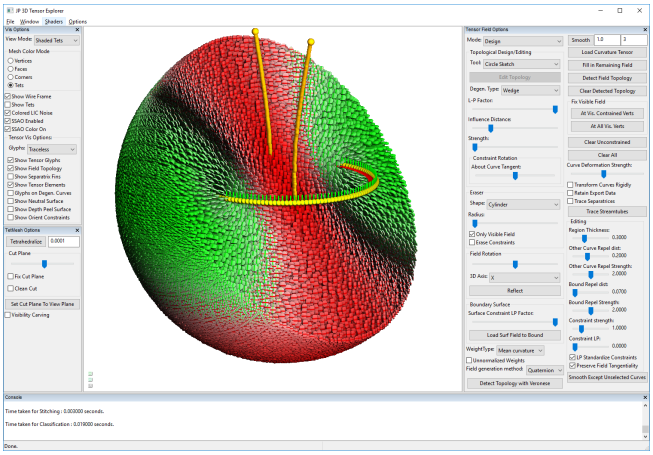


Fig. 6. *The interface of our tensor field design system.* Our UI includes a graphics display sub-window (middle), a visualization option panel (left), and a design option control (right). Please refer to the accompany video for live actions.

4 OVERVIEW

We now describe our 3D tensor field design system, which consists of two steps. First, the user generates a tensor field by providing constraints inside the volume and/or on the boundary surface. Second, the user performs editing operations to individual degenerate curves of the tensor field. The user interface of our system is shown in Figure 6.

The computational setup of our system is as follows. The input domain is a tetrahedral mesh $M \subset \mathbb{R}^3$. Quite often the input is actually a triangle mesh representing a closed, orientable surface. In this case the volume bound by the triangle mesh is tetrahedralized using existing software such as TetGen [Si 2015]. A 3D tensor field is represented as a set of tensor values, one per vertex of M. Piecewise linear interpolation is used to extend the tensor values from a set of points (vertices of M) to a continuous tensor field over M. When performing tensor field design and editing, we modify the tensor field by modifying its values on the vertices of M. After each modification to the tensor field, the degenerate curves are extracted based on the method of Zheng and Pang [2004]. The **LP**-type as well as wedge/trisector classification along degenerate curves are also performed and color-coded. As shown in Figure 5, an **L**-type wedge degenerate point is colored in green, while an **L**-type trisector point is colored in blue. A **P**-type degenerate point is colored in yellow if a wedge, or red if a trisector. We will describe the first step, i.e., the specification of tensor fields through a set of user constraints, in Section 5. The discussion of topological editing of tensor fields is in Section 6.

5 3D TENSOR FIELD SPECIFICATION

Our specification system is based on a number of requirements. First, it is important to be able to specify the tensor value at a given point in space. Second, if the desired tensor value is degenerate, it is usually important to also specify the degenerate tensor pattern near the point of interest. Third, we would like the interface to be both intuitive and easy-to-use.

Our specification system consists of the following steps. First, the user places *design elements* using our design system, each of which is a desired tensor value at a given point, usually inside a tetrahedron in the mesh. Second, the design elements are used to generate tensor values at a set of vertices in the mesh, usually the vertices of the tetrahedrons that contain the design elements. Each of such vertices are referred as a *fixed vertex*. Finally, the tensor values at the fixed vertices are propagated to the remaining mesh vertices as follows.

Propagation. We treat each entry in the tensor field $T_{i,j}$ as a scalar field. A 3D symmetric, traceless tensor field can be considered as five scalar fields defined on the same mesh. The problem of propagating tensor values from the fixed vertices to the remaining vertices is thus converted into computing five scalar fields over the mesh. For each scalar field the values are given at the fixed vertices. To find the values at the remaining vertices, one can use the so-called Laplacian smoothing framework. Basically, for each scalar field, we create a *harmonic* vertex-based scalar field ψ on M given the boundary conditions (from the values at the fixed vertices). Note that ψ is harmonic if it minimizes the following energy,

$$\int_{x \in M} \frac{1}{2} \|\nabla \psi\|^2 dx = \sum_{t \in \mathcal{T}} \frac{1}{2} V_t \|\nabla \psi\|^2 \tag{1}$$

where $\mathcal{T}$ is the set of M's tetrahedrons, V_t is the volume of the tetrahedron t, and $\nabla\psi$ is the gradient vector of ψ. When ψ minimizes this energy we say that it solves the Laplace equation. This energy is equivalent to using cotangent edge weights [Jacobson 2013].

Once all five scalar fields have been obtained over the whole mesh, they are assembled into a 3×3 symmetric, traceless tensor field, which satisfies the boundary condition (fixed vertices). Note that this framework has been used successfully in 2D tensor field processing [Alliez et al. 2003; Zhang et al. 2007].

Design. Next, we focus on the details of the second step, i.e., computing the set of fixed vertices and tensor values at these vertices based on user-specified design elements.

A design element can be either degenerate or non-degenerate. In the latter, it is in the form of a non-degenerate tensor. A degenerate design element is in the form of

$$T_0 + (x - x_0)T_x + (y - y_0)T_y + (z - z_0)T_z \tag{2}$$

Here (x_0, y_0, z_0) is the 3D coordinates of the vertex, T_0 is a degenerate tensor, while T_x, T_y, and T_z are traceless tensors that may be degenerate or non-degenerate. Together, the coefficients of T_x, T_y, and T_z form the Jacobian of the tensor field and are responsible for the local tensor degenerate patterns (wedges, trisectors) around the point where the constraint is placed.

We have found that it is easier and often more intuitive to specify not just a single design element, but a set of design elements along a curve. This is especially true when specifying degenerate tensor elements, which naturally form curves. Given a user-specified curve, our system first generates a spline that best captures the sketched curve. The spline curve is then subsampled at a set of evenly spaced

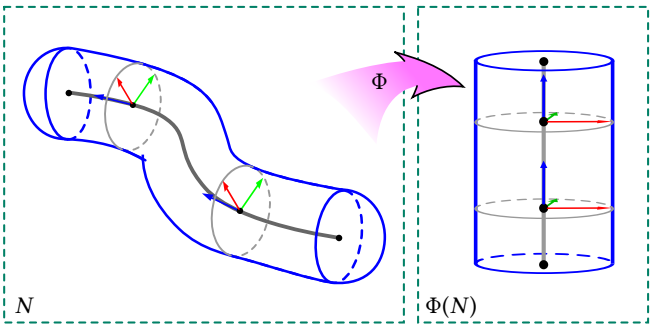


Fig. 7. *Space-warping parameterization for constraint generation.* Left: a tubular neighborhood N surrounding the user's placed curve c (dark gray with black endpoints in the center) is found using the fast marching method. This region is then mapped to a cylindrical parameter space $\Phi(N)$, so that we can easily generate a 3D tensor field in N. To further illustrate the mapping locally, we show two local frames along c with the X, Y, Z axes colored in red, green, and blue.

points on the curve including both end points. At each sample point p, we compute a frame based on the method of Bergou et al. [2008]. The first vector in the frame, i.e., the curve tangent, is assumed to be a non-degenerate eigenvector.

For a non-degenerate design element, the other two vectors in the frame are assumed to be the remaining eigenvectors. The user can change the eigenvectors by freely rotating the frame in 3D. The user can also specify the eigenvalues along the degenerate curves. The default maximal eigenvalue is 1 for **L**-type, and the default minimal eigenvalue is -1 for **P**-type of constraints.

For a degenerate design element, the other two vectors in the frame give the degenerate plane. Again, the user can change the non-degenerate eigenvector and the degenerate plane by rotating the frame. The default eigenvalues are $2, -1, -1$ for an **L** constraint and $1, 1, -2$ for a **P** constraint. The default values for the Jacobian are

$$T_x = \begin{pmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 0 \end{pmatrix}, T_y = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, T_z = 0 \tag{3}$$

for a wedge constraint. A default Jacobian for a trisector constraint simply negates T_y. The user can change the 2D tensor pattern inside the degenerate plane by changing the 2×2 sub-blocks in T_x and T_y. The user can also rotate the non-degenerate eigenvector direction freely in 3D, which will lead to the rotation of the 3D frame but the Jacobian coefficients do not change with respect to the new local coordinate system. This can lead to change in the angle between the non-degenerate eigenvector and the curve tangent.

Parameterization. Notice that the subsamples along a user-specified curve are usually not the vertices of the mesh. Recall that in our framework, the fixed vertices are at the vertices of the mesh, not the middle of the tets. While it is possible to find the set of tets containing the user-specified curve and somehow transfer the tensor constraints from the curve to the vertices of these tets, we have found that such an approach leads to poor control over tensor patterns (especially degenerate patterns) near the degenerate curves.

Consequently, we compute a tubular neighborhood for the specified curve (cylinder if open curve and torus if a loop). An illustration is shown in Figure 7 (left). We then compute a volumetric parameterization of this neighborhood with respect to the curve, which is equivalent to deforming the neighborhood into a canonical neighborhood as shown in Figure 7 (right). Then for each vertex v inside the neighborhood, we locate the closest vertex v_0 on the curve and applies the displacement vector $v - v_0$ into v_0's local linearization (tensor value and Jacobian) to obtain the desired tensor value at v. When two user-specified curves have intersecting neighborhoods, a conflict occurs for vertices inside the intersection. Consequently, we allow the desired tensor values from two curves to be blended, with each tensor value weighted by its distance to the respective curve. This way, a vertex closer to one curve will receive a larger influence from that curve. In our system, the radius of the neighborhood can be adjusted by the user. Generally speaking, the larger the radius, the better control over local tensor patterns around degenerate curves.

Boundary-conforming tensor fields. The users often wish the designed tensor field to conform to the boundary surface of the volume, i.e., one of the eigenvector fields is aligned with the surface normal everywhere on the boundary. However, tensor fields generated from the aforementioned Laplacian system are in general not boundary-conforming. To handle this difficulty, we perform one more Laplacian smoothing using some additional fixed vertices. Let $\mathbf{T}$ be the tensor field generated from design elements as described earlier. For each vertex v on the boundary surface of the domain, we modify the tensor value at v to be aligned with the boundary normal. Specifically, let δ be a local basis at v such that the normal at v, N_v is the third vector in δ. Under this basis

$$\mathbf{T}(v) = \begin{pmatrix} M_B & M_{BN} \\ M_{BN}{}^{\mathsf{T}} & M_N \end{pmatrix} \tag{4}$$

where M_B is a 2×2 matrix corresponding to the projection of $\mathbf{T}(v)$ in the tangent plane at v, B_v. M_{BN} is a 2×1 matrix, and M_N is a 1×1 matrix. To make $\mathbf{T}(v)$ boundary-conforming, we simply change it by setting $M_{BN} = 0$. We now add these boundary vertices to the set of fixed vertices (from design elements) and perform Laplacian smoothing a second time. The resulting tensor field is boundary-conforming and respects design elements.

Tensor fields from curvature tensor. The user may wish the tensor field to be aligned with the curvature tensor field on the boundary surface. We reuse the idea above, except that we first need to convert the curvature tensor, a 2×2 tensor, into a 3×3 tensor. This is achieved by adding an eigenvalue corresponding to the new eigenvector, the surface normal. The user can also specify the new eigenvalue. Once the curvature tensor has been converted into a 3D tensor field, it is included in the boundary condition.

Local field smoothing. It is often important to reduce the topological and geometric complexity of the tensor field in a region R with the field outside R unchanged. We refer to this operation as *local field smoothing*. This is achieved with the same tensor-valued Laplacian smoothing framework with different boundary conditions.

If R is strictly in the interior of the volume, we use the tensor values on the boundary of R, which are in the interior of the volume, as the boundary condition for the Laplacian system. If R intersects

the boundary of the volume, we also use $K = R \bigcap \partial M$ as a boundary condition.

6 TOPOLOGICAL EDITING OPERATIONS

The tensor fields generated in the first step (Section 5) often contain degenerate curves in addition to the ones specified by the user (Figure 8). As mentioned earlier, excessive and misbehaving degenerate curves can lead to visual artifacts in the applications of tensor fields. In this section, we describe three *fundamental* topological editing operations that we have identified as part of this research:

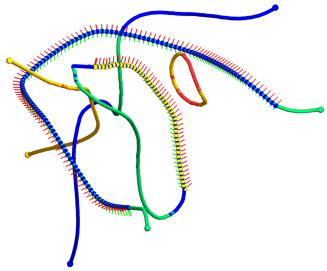


Fig. 8. *Tensor field design based on purely user-specified constraints (curves with spikes) can lead to excess degenerate curves (without spikes).*

Degenerate curve deformation which refers to deforming part of a degenerate curve (Section 6.1).

Degenerate curve reconnection which refers to cutting open two degenerate curve segments and stitching together pieces from different segments, thus resulting in two new degenerate curve segments (Section 6.2).

Degenerate curve removal which refers to removing either one degenerate curve or simultaneously two degenerate curves, under respective conditions (Section 6.4).

These operations are designed to impact the least number of degenerate curves, while, together, can provide enough flexibility to modify tensor field topology. Figure 9 shows a sequence of topological editing operations on a tensor field. Next we describe our analysis and algorithms that enable these editing operations.

6.1 Degenerate Curve Deformation

Deforming a degenerate curve γ by itself is straightforward, but we need to resample tensor values at the surrounding mesh M vertices while keeping the rest intact (boundary condition). To deform part of a degenerate curve γ, our system identifies a region M (Figure 10 (a-b): cylinder), a topological ball that encloses the segment $\gamma_1 \subset \gamma$ (Figure 10 (b): curve) and its deformed version γ_2 (Figure 10 (a): curve) but no other degenerate points. Next, we strive for a self-homeomorphism ϕ of M that maps γ_2 to γ_1. Finally, we use the map ϕ to generate a deformed tensor field. Figure 9 shows the result of deforming the longest degenerate curve near Moai's ear.

Region M. We build the region surrounding the degenerate curves via a sequence of topology-aware morphological dilation and erosion operations. We start with the set of tetrahedrons containing γ_1 and γ_2. Typically the topology of this set, referred to as U, is not a ball. We then compute the distance function d_U of any tetrahedron in the mesh to U. Based on d_U we iteratively add one tetrahedron at a time to U' (initially the same as U) until U' is a ball. We then shrink U' by iteratively removing tetrahedrons from U' while keeping both

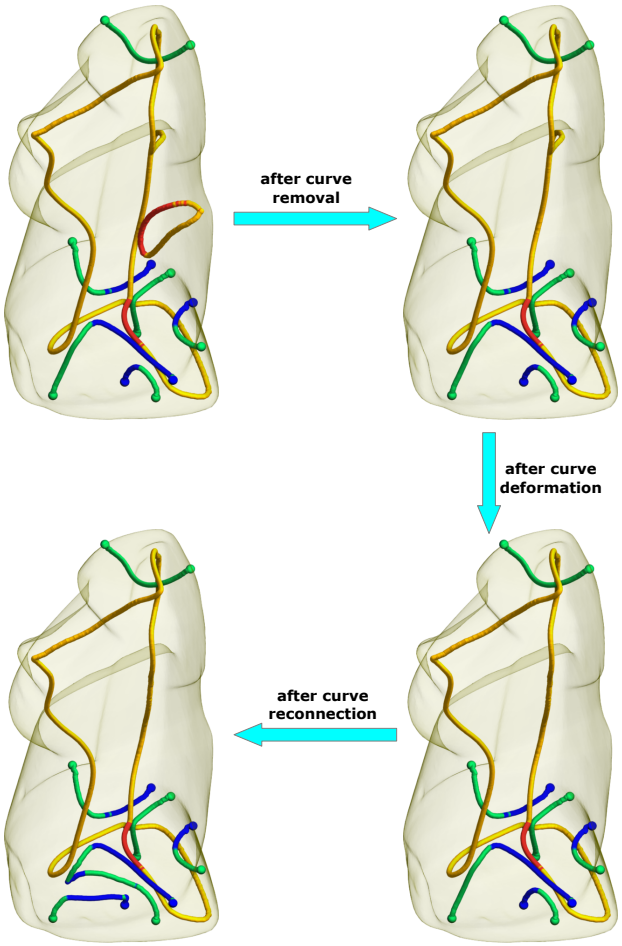


Fig. 9. *A sequence of topological editing operations applied to a 3D tensor field.* (upper-left) the input tensor field, (upper-right) remove a mixed degenerate loop inside Moai's body (yellow/red), (lower-right) deform part of the longest degenerate curve near Moai's ear (yellow), and (lower-left) reconnect two mixed degenerate curves near Moai's bottom (both green/blue).

γ_1 and γ_2 inside U' and requiring U' to remain a topological ball. Once this process terminates, we compute a new distance function, $d_{U'}$, which measures the distance to the closest tetrahedron in U', and use it to grow the region M from U' out to a user specified distance (30% of the mesh radius works well in our experiments). A tetrahedron with the minimal d_U (or $d_{U'}$) will not be added if it contains part of a degenerate curve not intended by this editing operation.

Mapping ϕ. We construct the map ϕ by treating it as a volumetric parameterization problem with the conditions that the parameterization is the identity map on the boundary of M and maps the desired degenerate curve segment γ_2 to the original curve γ_1. Note that degenerate curves are usually inside tetrahedrons. Consequently, we first establish a bijective map of γ_2 to γ_1 by sampling both curves with the same number of sample points. Next, we compute the minimal tetrahedral envelope for γ_2, i.e., any tetrahedron in the envelope

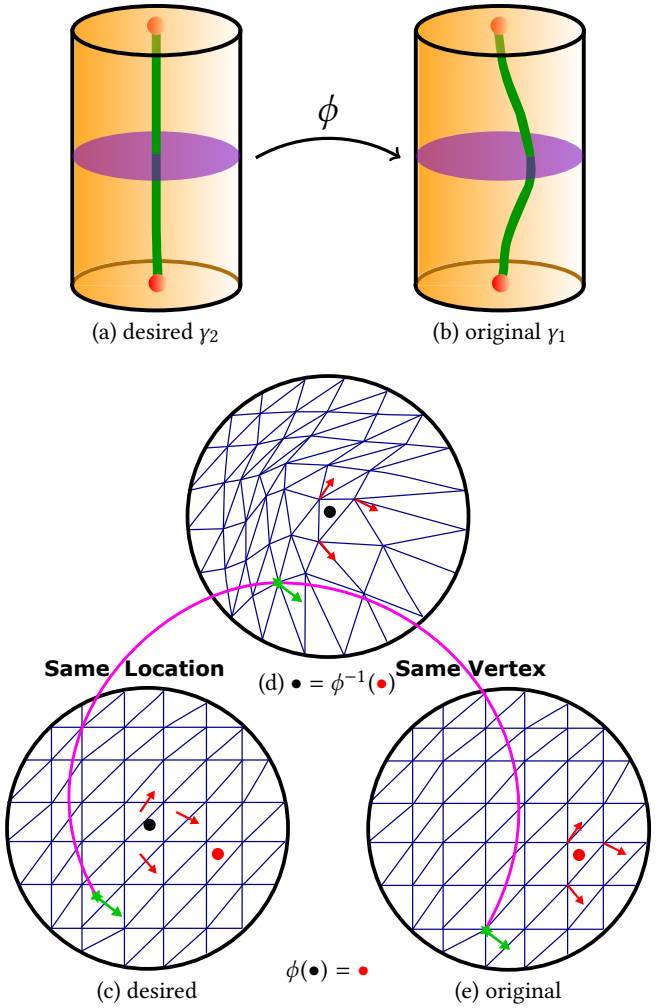


Fig. 10. *Our deformation algorithm modifies the tensor field to make the degenerate curve move from its location in the input (b) to a user designed output curve (a).* The procedure is shown in (c)-(e) for the purple cross section in (a) and (b). We compute a map ϕ that deforms the neighboring vertices of the target location (black dot) (c) into the neighborhood of the original singularity (red dot) (e) for interpolation.

contains at least part of γ_2. For each vertex v_i in the envelope, we find the closest sample point p_i on γ_2 and compute the displacement vector $dv_i = v_i - p_i$. We then set the constraint that $\phi(v_i) = q_i + dv_i$ where q_i is the sample point on γ_1 corresponding to p_i. With these constraints, we solve the parameterization of the remaining vertices in M by solving the Laplacian given by Equation (1) on the coordinates of the points. Note that since our map takes γ_2 to γ_1, we are essentially assigning a pre-deformation location to each vertex.

Note that the parameterization may have fold-overs and usually has suboptimal distribution of stretches. This can cause the resulting tensor field to be less smooth. If the fold-over tetrahedrons cover the degenerate curve, the editing operation may fail as there can be extra degenerate curves after the deformation. However, we find that this relatively simple method works well in practice.

Interpolation. Once the parameterization is given, we can look up the new tensor value at each vertex $v_i \in M$ by using its pre-image under the parameterization, $\phi(v_i)$. We then set the tensor value at v_i to be that of $\phi(v_i)$. Figure 10 (c-e) shows the process of computing ϕ and deforming the tensor field with a 2D illustration.

6.2 Degenerate Curve Reconnection

Let γ_1 and γ_2 be two segments of degenerate curves that do not share any common degenerate points. Furthermore, the endpoints of γ_1 are p_1 and q_1 while the endpoints of γ_2 are p_2 and q_2. A degenerate curve reconnection of γ_1 and γ_2 will result in a new tensor field whose degenerate curves are the same as the original tensor field except that γ_1 and γ_2 are replaced with γ_1' and γ_2' where γ_1' has endpoints of either p_1 and p_2 or p_1 and q_2 while γ_2' is bounded by the other two points. Figure 11 shows the atomic scenarios.

Note that $\{p_1, q_1, p_2, q_2\}$ can be on the same degenerate curve or two different degenerate curves. Reconnecting them can change the number of degenerate curves in the field. For example, if these points are on a degenerate loop, after reconnection the loop can be broken up into two. Reconnection allows us to change the connectivity of the degenerate curves to facilitate degenerate curve deformation and removal. Figure 9 shows a reconnection of two mixed degenerate curves (both green/blue) near Moai's bottom.

Uniqueness of reconnection. When two degenerate curves are reconnected, there appear to be two ways of doing so, i.e., (1) p_1, p_2 and q_1, q_2, and (2) p_1, q_2 and p_2, q_1. Consequently, it seems that the user would need to specify how the reconnection occurs. The following analysis shows that this is not necessary. Once the user specifies the curves to reconnect, there is only one way to reconnect them. To see this, we need the following result from [Markus 1955], which is also referred to as *combing* [Vaxman et al. 2016]. Given a line field $\mathcal{L}$ defined in $\mathbb{R}^3$, let B be a simply connected, finite region such that $\mathcal{L}|_B$ has no singularities/zeros. Then $\mathcal{L}$ can be turned into a continuous vector field without singularities.

We now consider a region B where the non-repeating eigenvector field (a line field) has no singularities, then the non-repeating eigenvector field can be turned into a continuous vector field in B. Given a chosen orientation of the non-repeating eigenvector field inside B, the boundary of B is divided into the *incoming* region, where the vector field V points into B, and the *outgoing* region, where V points to the outside of B (Figure 11). The following observation is useful.

Lemma 6.1. *A pure wedge (or trisector) curve intersecting the boundary of the above region B must have its two endpoints in different regions, i.e., one in the incoming region, and the other outgoing region. In contrast, a degenerate curve with one wedge segment and one trisector segment must have its endpoints in the same region, i.e., either both incoming or both outgoing.*

Proof. Trace the degenerate curve from one end so that the dot product between the forward curve tangent and the oriented non-degenerate eigenvector field is positive at the starting point. Notice that every time a transition point occurs, the dot product changes its sign, i.e., a turn has occurred. To travel from a point

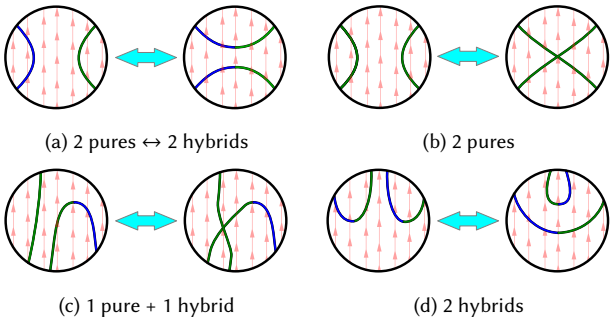


Fig. 11. *Different fundamental reconnection scenarios.* The non-repeating eigenvector field in the regions are pointing upward, dividing the boundary of the regions, a topological sphere, into incoming regions (below the equator) and outgoing regions (above the equator). A green segment is wedge, and a blue segment is a trisector. Notice that there are two fundamental cases in (a), while one fundamental case each for the rest.

inside the incoming region to the outgoing region, there must be an even number of turns, i.e., transition points. To connect two points in the same region (either incoming or outgoing), there must be an odd number of transition points. See Figure 11 for illustration. □

Theorem 6.2. *Given two degenerate curve segments γ_1 and γ_2 to be reconnected inside a simply-connected region M, there is only one way to reconnect them.*

Proof. Assume that p_1 and p_2 are in the incoming region, and q_1 and q_2 are in the outgoing region (e.g. Figures 11a and 11b left). This means that $\mathbf{T}(p_1)$ and $\mathbf{T}(q_1)$ have the same wedge/trisector type, and $\mathbf{T}(p_2)$ and $\mathbf{T}(q_2)$ have the same wedge/trisector type. Without the loss of generality, assume that $\mathbf{T}(p_1)$ and $\mathbf{T}(q_1)$ are both wedge types. If $\mathbf{T}(p_2)$ and $\mathbf{T}(q_2)$ are both wedges, then p_1 and p_2 cannot be reconnected. This is because if they could be reconnected, then there must be an even number of transition points between two wedges. On the other hand, since both p_1 and p_2 are in the incoming region, Theorem 6.1 states that there must be an odd number of transition points between p_1 and p_2, a contradiction. In this case, it is only possible to reconnect p_1 with q_2, and p_2 with q_1. Now, assume that p_2 and q_2 are trisectors. Using similar arguments (Theorem 6.1) it is straightforward to verify that p_1 cannot be reconnected with q_2. The only possibility is to reconnect p_1 with p_2, and q_1 with q_2. Similar analysis can show that reconnection is unique for other cases. □

There are only five fundamental reconnection scenarios as shown in Figure 11. Note in case (a) there are two scenarios: from left (1 wedge and 1 trisector) to right (two hybrid wedge/trisector), and from right to left. In other cases, the reconnection from left to right and from right to left are equivalent.

Reconnection algorithm. The steps can be summarized as follows:

(1) Compute a topological ball M that contains γ_1 and γ_2 but no other degenerate points (Figure 12a). The process is similar to computing the envelope B for degenerate curve deformation (Section 6.1).

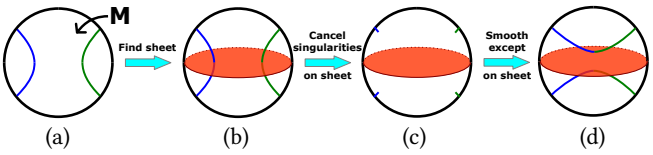


Fig. 12. *Steps for reconnecting degenerate curves for Figure 11a.* From left to right: compute the containing region M, find an insulating sheet (red), over which generate a singularity-free tensor field, and propagate it to M.

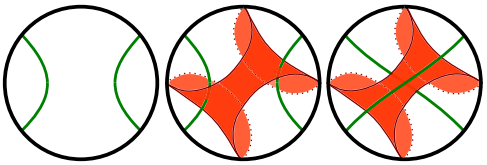


Fig. 13. *The sheet for Figure 11b is more complex than the sheet shown in Figure 12 for Figure 11a.*

(2) Determine which pair of endpoints should be connected after reconnection.
(3) Compute a sheet (Figure 12b: red plate) inside M that separates M into two regions, each of which contains precisely two endpoints that are connected after the reconnection.
(4) Reassign tensor values on the interior vertices of the sheet so that there are no degenerate points on the sheet.
(5) Recompute the tensor values for the remaining interior vertices of M that are not already on the sheet. This is achieved by simultaneously solving Equation (1) for each entry in the tensor field.

Below we provide more details on steps 2, 3, and 4.

Step 2. To decide how to pair the endpoints after reconnection, we need to convert the non-repeating eigenvector field into a continuous vector field in M. This is achieved as follows. We first choose a vertex in the region M and arbitrarily choose a forward direction from two of its unit non-repeating eigenvectors. We then parallel transport this vector to all the other vertices in M and use it to choose the forward direction of each of the non-repeating eigenvectors. This will lead to the conversion of the non-repeating eigenvector field on M to a vector field since it is free of singularities as a line field. Depending on the type of degenerate points the endpoints of the curves are (wedge or trisector) and where they situate on the boundary of M with respect to the vectorized non-repeating eigenvector field (incoming or outgoing), we determine which pair of endpoints to reconnect after reconnection using the idealized illustrations in Figure 11.

Step 3. To generate the sheet, we compute two curves, one for each pair of endpoints after reconnection, by using Fast Marching [Kimmel and Sethian 1998] to find the shortest path between the desired endpoints. We require that the two curves do not intersect, which is generally the case based on our observation. If there are intersections, we perform local adjustment in M to remove the intersections. Next, we compute the distance function from these

two curves to the rest of the vertices in the region M and record which curve is the closest to the vertices. We then extract all the edges in M whose two vertices have different closest curves. The set of tetrahedrons that contains these edges is a thickened version of the sheet.

Step 4. To strive for a degenerate point free tensor field in the sheet, we consider a more generic problem: given a simply-connected, topological ball over which the tensor field's non-repeating eigenvector field has no singularities, how to modify the tensor field in the region to be free of degenerate points with tensor values fixed at part of or the whole boundary of the region? The algorithm to achieve this is not only useful for degenerate curve reconnection, but also degenerate curve removal (Section 6.4). We provide the details of this in Section 6.3.

6.3 Singularity-Free Tensor Field Generation

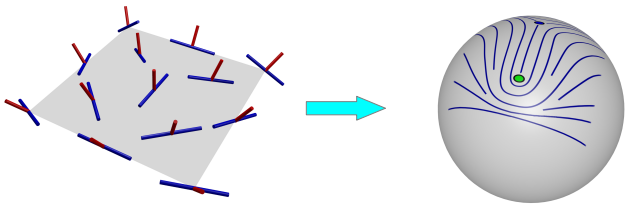
Given a simply-connected region M where the non-repeating eigenvector field has no singularities, any degenerate point must be a singularity in the medium eigenvector field. This implies that to compute a degenerate-point-free tensor field inside M, we only need to ensure that the medium eigenvector field has no singularities. In addition, as pointed out earlier (Section 6.2), the non-repeating eigenvector field in M can be turned into a continuous vector field. Consequently, this vector field can be considered as a map ϕ from M to the unit sphere $\mathbb{S}^2$. Given a point $p_0 \in M$, a unit medium eigenvector at p_0 can be considered as a tangent vector at $\phi(p_0)$. Figure 14 illustrates this map.

Angle field. We further model the medium eigenvector field as a scalar field, i.e. its angular component, and perform Laplacian smoothing (Equation (1)) on this scalar field with the given boundary conditions. The resulting scalar field (angular component of the medium eigenvector field) will then be combined with the non-repeating eigenvector field (not changed in the process) to generate the new tensor field inside region that is free of degenerate points.

Note that the angular component of a vector depends on the coordinate system. Give two points q_1 and q_2 on $\mathbb{S}^2$, their tangent planes are not the same, nor are their respective coordinate systems correlated. We thus need a continuous, singularity-free vector field defined on M that is also perpendicular to the non-repeating eigenvector field everywhere in the domain. This vector field, denoted by U, serves as the references with which we compute the angular component of the medium eigenvector field.

Parallel transport. To compute this reference vector field U, one can start with a seed vertex in the region M and assign a reference vector there. This reference vector is then iteratively propagated to the remaining vertices in M through its edges (e.g. breadth-first search). Given an edge $e = (v_0, v_1)$ where the reference vector $U(v_0)$ is available, we obtain $U(v_1)$ by parallel transporting $U(v_0)$ along the geodesic connecting $\phi(v_0)$ and $\phi(v_1)$ (Figure 15).

Jump. Results from classical differential geometry [Lai et al. 2010] state that when parallel transporting a vector along a simple loop on the sphere, the resulting vector may differ from the initial vector by a 2D rotation. The angle of the rotation is proportional to the area of the region enclosed by the loop. This means that when



(a) tensor field (b) line field

Fig. 14. *Mapping a 3D tensor field to a tangential vector field on the* Gauss sphere. A point in the plane can be mapped to the sphere based on its non-repeating eigenvector (red), and the tensor field (a) is now reduced to a tangential line field on the sphere (b) where the line field corresponds to the medium eigenvectors in the tensor field (blue). Singularities in the medium eigenvector field (green dot) are degenerate points in the tensor field, which we strive to remove.

parallel transporting the reference vector $U(v_1)$ from a vertex v_0 to v_1 and then to v_2 where v_0, v_1, and v_2 form a spherical triangle (Figure 15), the resulting reference vector $U(v_2)$ will be different from directly parallel transporting $U(v_0)$ to v_2 along the edge v_0v_2. The angular difference α is proportional to the *signed* area of the spherical triangle $\phi(v_0)\phi(v_1)\phi(v_2)$. We refer to $-\alpha$ as the jump from v_0 to v_2. Note that in our reference propagation process, the jump is zero for edges used in the propagation and usually non-zero for the ones not used.

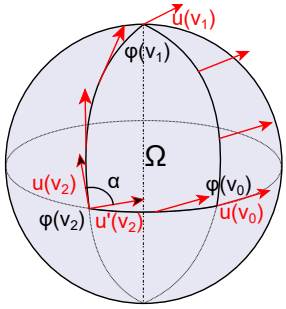
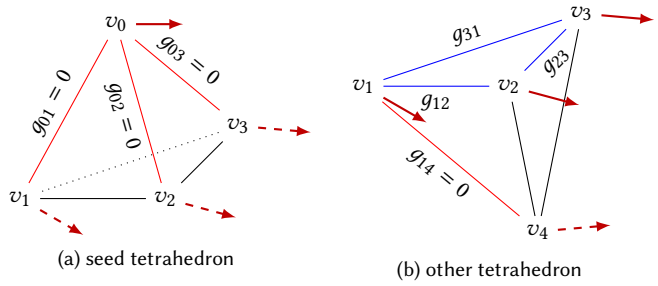


Fig. 15. *Parallel transport and jump.*

Propagation. If jumps are not properly accounted for, singularities can occur in the medium eigenvector field [Lai et al. 2010; Li et al. 2006]. Consequently, our reference propagation algorithm also simultaneously computes jumps for edges not used in the propagation.

As illustrated in Figure 16a, we start with a seed vertex v_0 where the reference vector is already defined. We then select one of the tetrahedrons incident to v_0 as the seed tetrahedron. Denote this tetrahedron as t_0, which has three other vertices v_1, v_2, v_3. We first propagate the reference vector from v_0 to v_1 and set the jump to zero on the edge v_0v_1. We handle v_2 and v_3 in a similar fashion. Next, we set the jump on edge v_1v_2 to be the negative of the signed area of the spherical triangle $\phi(v_0)\phi(v_1)\phi(v_2)$. The jumps for the edges of v_2v_3 and v_3v_1 are computed in the same way.

Once t_0 has been processed, we perform region growing from t_0 until all the tetrahedrons in M have been covered as demonstrated in Figure 16b. In every iteration, a new tetrahedron t in M that shares a face with the set of visited tetrahedrons is included in the set. Moreover, we require that at any given moment the set of visited tetrahedrons forms a topological ball. Note that there is at most one vertex of t that has not been assigned a reference vector, and at most



(a) seed tetrahedron (b) other tetrahedron

Fig. 16. *Reference vector field propagation.* Given the seed vertex v_0 and seed tetrahedron (a), the reference vector $U(v_0)$ is propagated to v_1, v_2, and v_3 (dashed lines) along edges with a jump zero (red edges). The jumps for the remaining edges (black) will be computed based on areas of the triangles with two red edges and one black edge. For other tetrahedron (b), three vertices have been assigned a reference vector and three edges (blue) have their jumps computed. The reference vector at the new vertex v_4 can be obtained from one of the three vertices where the reference vectors are available, with the corresponding edge given a zero jump (red). The other edges (black) will have their jumps computed based on the areas of triangles with only one black edge.

three edges whose jumps have not been computed. We assign the new vertex a reference vector by parallel transporting the reference vector from that of one of the other three vertices in the tetrahedron and set the jump to zero for the corresponding edge. If there are still edges in t for which the jump has not been computed, we identify a triangle in t incident to the edge that has two edges with already computed jumps, and subtract the area of the spherical triangle from the sum of the two already computed jumps to get the third jump. Note that this is always possible since our algorithm requires that at the end of each iteration the visited tetrahedrons form a topological ball.

Conversion. Once the reference vector field U is generated and the jumps computed, we convert the medium eigenvectors at the fixed vertices of the region M as follows. Starting from a seed fixed vertex v_0, we compute $\theta(v_0)$ as the oriented angle from $U(v_0)$ to the medium eigenvector there. To compute $\theta(v_1)$ where v_1 is adjacent to v_0, we first assign $\theta(v_1)$ to be the oriented angle from $U(v_1)$ to the medium eigenvector at v_1. Let g_{01} be the jump from v_0 to v_1. We then find $k_0 \in \mathbb{Z}$ so that

$$|\theta(v_1) + 2k_0\pi - (\theta(v_0) + g_{01})| < \frac{\pi}{2} \tag{5}$$

and update $\theta(v_1)$ to be $\theta(v_1) + 2k_0\pi$. This process repeats until θ has been computed for all the fixed vertices.

Next, we compute θ for vertices in M that are not part of the boundary condition by using the θ values at the fixed vertices. To solve for the new angular field, we need to modify Equation (1) to take into account the jumps. That is, given a vertex v_0 and the set of vertices $\{v_1, ..., v_k\}$ adjacent to v_0, the Laplace of the angular component is

$$\sum_{j=1}^{k} w_j \left(\theta(v_j) - \theta(v_0) - g_{0j}\right) \tag{6}$$

where g_{0j} is the jump from v_0 to v_j and w_j is the cotagent weight [Jacobson 2013].

Finally, the θ field is converted back to a line field by using the reference vector field U. This is the singularity-free medium eigenvector field, which, by being combined with non-repeating eigenvector field (not changed during the process), generates the new eigenvector fields in M that is free of degenerate points.

In addition to the eigenvector fields, we also need to compute the eigenvalue fields in order to obtain the tensor field in M free of degenerate points. This is achieved as follows. We consider the major eigenvalue field as a scalar field. Given the major eigenvalues at the fixed vertices of M, we find the major eigenvalues elsewhere in M by solving Equation (1). Similarly, we compute the minor eigenvalues in M. Finally, since the tensor field is traceless, the medium eigenvalue field can be obtained from the major and minor eigenvalue fields. This leads to a tensor field in M free of degenerate points.

6.4 Degenerate Curve Removal

The third topological editing operation is *degenerate curve removal*, which is responsible for reducing the number of degenerate curves in the field. Figure 9 shows the removal of a degenerate loop inside Moai's body (yellow/red).

Recall that in 2D tensor fields, at least two degenerate points with opposite tensor indices (wedge and trisector) must be removed simultaneously. However, in 3D tensor fields, there does not exist a known index to measure degenerate points or curves. A degenerate curve may consist of both wedge and trisector segments, and each degenerate curve has an additional linear-planar (**L−P**) classification. These additional complexities make it more challenging to decide the necessary and/or sufficient conditions for degenerate curve removal.

Empirically, we have identified the following four scenarios of degenerate curve removal (Figure 17):

One-curve removal One open degenerate curve with opposite wedge/trisector types at its ends is removed (Figure 17a).

Two-curve removal One open wedge degenerate curve is cancelled with one open trisector degenerate curve of the same linear/planar type (Figure 17b).

One-loop removal One degenerate loop with both wedge and trisector sections is removed (Figure 17c).

Two-loop removal A pure wedge degenerate loop is cancelled with a trisector degenerate loop of the same linear/planar type (Figure 17d).

For all but the last case, the enclosing region is a topological ball (simply connected). For each case the removal operation is always theoretically possible if an enclosing topological region can be found that intersects no other degenerate curves. Our degenerate curve removal algorithm for these cases again makes use of the fact that inside these enclosing regions (if they exist), the non-repeating eigenvector field is a vector field. Therefore, we employ a similar framework to that of degenerate curve reconnection (Section 6.2).

We first compute a region M of the appropriate type given the type of the above scenarios using a method similar to region growing

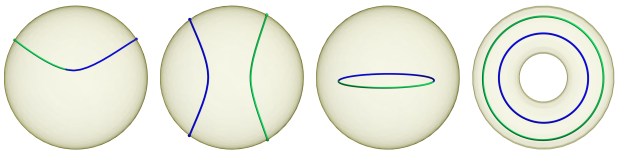


Fig. 17. *Fundamental removal scenarios.* Note that the enclosing region is a topological ball for cases (a)-(c) and a topological torus for case (d).

for degenerate curve deformation and reconnection. Next, we use the tensor values on the boundary M as constraints and generate the tensor field inside M using the angle-based method in Section 6.3.

For two-loop removal, the enclosing region is a topological solid torus (not simply-connected). Consequently, the angle-based method (in particular, the jump assignment step) would not always result in a singularity free continuous tensor field inside the region. Therefore, we implement two-loop removal operation as a composite operation. By first reconnecting the two loops, we obtain a single loop with both wedge and trisector segments that can be handled with the one-loop removal.

For all our topological editing operations, the success depends on the shape and size of the region M. The larger and rounder the region, the higher the success rate is. We have conducted 47 experiments of applying topological editing operations to fields defined on various 3D models such as the bunny, Moai, buddha, and dragon. The fields were generated either through manual design or, more commonly in our experiments, extrapolation from the curvature tensor on the boundary surfaces. Our algorithms have a success rate of over 90%.

7 PERFORMANCE

Our tool has been tested on a system with Intel(R) Xeon(R) CPU with 3.40 Ghz speed with a RAM of 64 GB and an Nvidia Quadro K420 graphics card. We have tested our design system on various models, with a resolution varying from $150,000$ tets to $1,000,000$ tets. These models can be mathematically defined (e.g., sphere/torus), organic (e.g., Stanford bunny, feline, fertility), or CAD models (e.g., rocker arm and fan disk). The boundary of these tet meshes have a range of $10,000$ to $60,000$ triangles. The tet meshes are usually generated from the boundary mesh using TetGen, which typically leads to adaptive meshes where the tets are much smaller around sharp features. Our system can handle such meshes since we do not assume a minimal tet mesh resolution anywhere in our algorithm and implementation. Our system can also be used to generate tensor fields on meshes with sharp edges, such as the rocker arm. Solving for the field from boundary or interior constraints takes less than five seconds on all of our test models. All of the editing operations finish within 10 seconds for most models, and within 60 seconds for larger models ($> 500,000$ tets).

8 APPLICATIONS

In this section we demonstrate applications of our method in solid texture and geometry synthesis.

8.1 Solid Texturing

Example based solid texture. In 2D texture synthesis the goal is to take a small input exemplar texture and generate larger output texture with the same visual characteristics; this problem is well-explored as surveyed in [Wei et al. 2009]. Solid texture synthesis is the logical extension of this problem to the voxel case, and relatively fewer methods have been proposed [Dong et al. 2008; Owada et al. 2004; Pietroni et al. 2007; Takayama et al. 2008; Zhang et al. 2011]. Interestingly, with the exception of [Takayama et al. 2008] most methods still rely on 2D exemplars as the direct input, as they are easier to acquire. When the input texture exhibits anisotropy, an orientation field is needed to guide the orientation and placement of the exemplar texture [Takayama et al. 2008; Zhang et al. 2011]. As generating smooth fields that naturally follow shapes can easily be accomplished by our method (nearly automatically, when using the curvature tensor), our system readily benefits this application. The only difference is that a tensor field is sampled instead of a vector or frame field, and the eigenvectors must be computed and used as the local orientation for each neighborhood. As long as the texture is symmetric (or at least nearly so), the algorithms work without other modification. We adapt the method of [Zhang et al. 2011] to use symmetric tensor fields. See Figure 18 for results.

Anisotropic procedural noise. Procedural noise is a powerful way to model natural phenomena and enrich visual details, with particular advantages such as fast evaluation, low memory consumption, virtually infinite resolution, and appearance control [Ebert et al. 2002; Lagae et al. 2010]. Of particular interest to us is solid anisotropic noise, such as that introduced in [Lagae et al. 2009] and extended in [Bénard et al. 2010; Lagae and Drettakis 2011], which requires an axis at each point in the target domain to guide the orientation of the corresponding anisotropic noise pattern. Vector fields are unable to model all of the features that anisotropic Gabor noise is capable of exhibiting. As far as we know, the results generated in [Bénard et al. 2010; Lagae and Drettakis 2011; Lagae et al. 2009] use only surface tensor fields for 2D noise and constant or simple 3D vector field functions to orient solid noise, likely because of the lack of methods for easily producing 3D tensor fields. Our system addresses this need, and allows the generation of much more complex effects. Also, we find that Gabor noise makes for an apt texture-based method of interactively visualizing 3D tensor fields, because of its fast computation. Examples of our artistic results using Gabor noise can be seen in Figure 18.

8.2 Geometry Synthesis

Placing and orienting 2D image/text and 3D geometry elements has shown to be an important application for many graphical and interactive tasks [Gal et al. 2007; Hausner 2001; Ijiri et al. 2008; Landes et al. 2013; Ma et al. 2011; Maharik et al. 2011; Ostromoukhov and Hersch 1999; Pedersen and Singh 2006; Praun et al. 2001; Roveri et al. 2015; Xu and Kaplan 2007]. These applications require good orientation fields for the corresponding domains for maximal visual benefits. However, as discussed in Section 2, the design of 3D fields has been much less explored than the 2D counterparts. Thus, our method offers benefits for applications that require the placement of 3D elements as exemplified by [Ma et al. 2011].

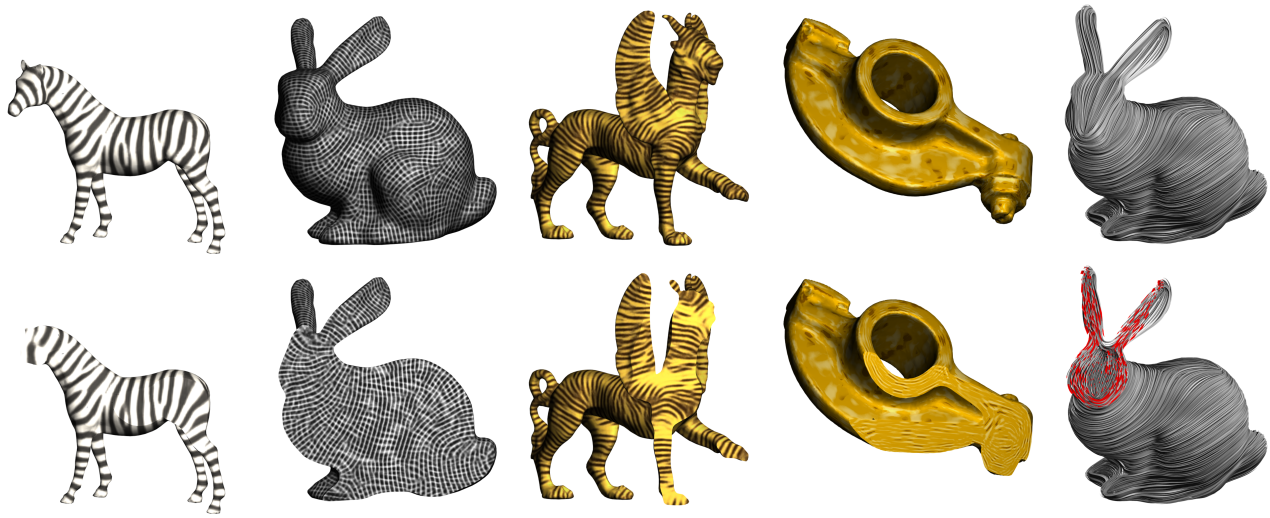
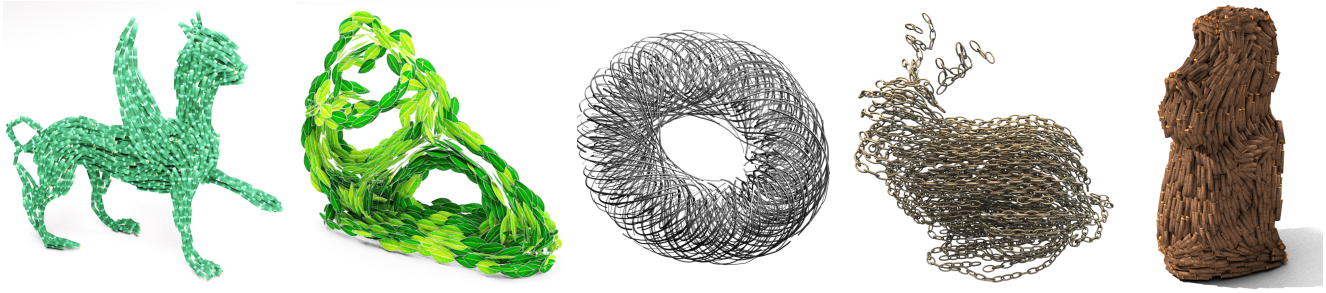


Fig. 18. *Solid texture synthesis based on examples (horse, bunny, and feline) and procedures (Gabor noise rocker arm and streamtubes bunny).* The output is rendered through different cross sections (bottom row) to illustrate its volumetric nature.



(a) bamboo feline (b) leaf fertility (c) wire torus (d) chain bunny (e) stick Moai

Fig. 19. *3D geometry element synthesis.* Bamboo, stick, and wire patterns are line-type elements while leaf and chain patterns have box-type symmetry.

Unlike prior methods which place elements according to a vector or frame field, our method deals with tensor fields which can generate line fields as well as frame fields satisfying the box symmetry (the Klein group [Conway et al. 2016]). Figure 19 shows various elements applied to a variety of 3D models with tensor fields designed using our tensor field design system. Moreover, due to the co-existence of linear/planar regions in a tensor field, we can have two different elements in the same field (Figure 2). Figure 20 shows the synthesis results using fields that were created completely from scratch and are not aligned with the boundary surface (the sphere in this case). Elements can also be used to visualize the effects of topological editing operations as shown in Figures 1 and 21.

9 LIMITATIONS AND FUTURE WORK

In this paper we introduce the problem of 3D tensor field design and have identified a number of graphics applications. We also provide the first 3D tensor field design system that is interactive, intuitive, and efficient. At the core of our system we provide the capability to design and control degenerate curves, which is tensor field topology. With these we have made a number of theoretical observations of 3D tensor field topology and identified a set of fundamental topological editing operations.

Our system has several limitations. First, our topological editing operations can fail. In the future, we plan to seek more theoretical understandings as well as explore improved algorithms that can lead to topological editing algorithms with guaranteed success. Second, our degenerate curve deformation algorithm relies on the ability to compute a bijective space warping of the region in which the operation is performed. However, we cannot guarantee that the warping is indeed bijective, i.e., foldovers may occur. We plan to explore volumetric parameterization techniques that are guaranteed to produce low-distortion and bijective maps. A series of recent work on the topic has the potential of benefiting our method [Aigerman and Lipman 2013; Campen et al. 2016; Kovalsky et al. 2015, 2016; Rabinovich et al. 2017; Schüller et al. 2013].

In the future, we plan to also improve the UI of our system to make it more accessible to non-expert users. Moreover, we plan

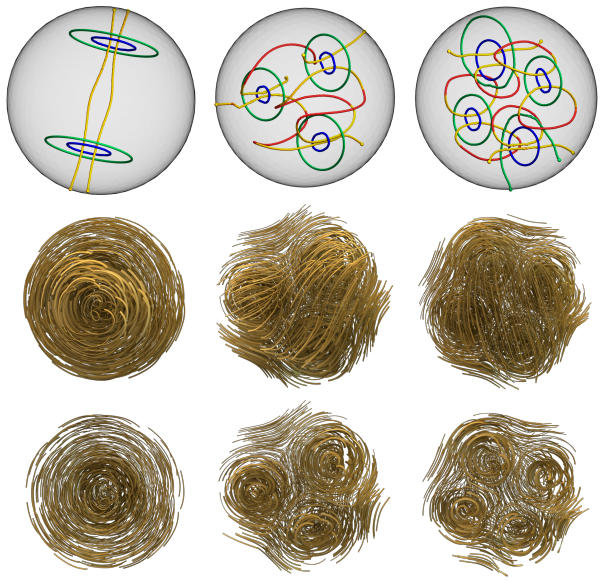


Fig. 20. *Tensor field visualization with a noodle element.* Three tensor fields (top row: only degenerate curves are shown) are applied with a noodle texture (middle row) where the noodles follow the major eigenvector directions. The bottom row shows the cutaway views.

to explore adding the design of trace into our tensor field design system. Our topological editing functions are also impacted by the quality of techniques for extracting tensor field topology, which is an area we plan to conduct further research in.

ACKNOWLEDGMENTS

We would like to thank Guoxin Zhang, Shimin Hu, and Kun Xu for sharing the code of [Zhang et al. 2011]; Yue Zhang, Amy Roy, and Tom Roy for their help in proofreading the paper; and the anonymous reviewers for their valuable suggestions. This work has been partially supported by US National Science Foundation awards (0546881, 0830808, 0917308, 1340112, and 1619383) as well as Hong Kong RGC general research fund 17202415.

REFERENCES

Noam Aigerman and Yaron Lipman. 2013. Injective and Bounded Distortion Mappings in 3D. *ACM Trans. Graph.* 32, 4, Article 106 (2013), 14 pages.

Pierre Alliez, David Cohen-Steiner, Olivier Devillers, Bruno Lévy, and Mathieu Desbrun. 2003. Anisotropic Polygonal Remeshing. *ACM Trans. Graph.* 22, 3 (2003), 485–493.

Pierre Bénard, Ares Lagae, Peter Vangorp, Sylvain Lefebvre, George Drettakis, and Joelle Thollot. 2010. A Dynamic Noise Primitive for Coherent Stylization. In *EGSR '10*. 1497–1506.

Miklós Bergou, Max Wardetzky, Stephen Robinson, Basile Audoly, and Eitan Grinspun. 2008. Discrete Elastic Rods. *ACM Trans. Graph.* 27, 3, Article 63 (2008), 12 pages.

David Bommes, Henrik Zimmer, and Leif Kobbelt. 2009. Mixed-integer Quadrangulation. *ACM Trans. Graph.* 28, 3, Article 77 (2009), 10 pages.

Marcel Campen, Cláudio T. Silva, and Denis Zorin. 2016. Bijective Maps from Simplicial Foliations. *ACM Trans. Graph.* 35, 4, Article 74 (2016), 15 pages.

John H Conway, Heidi Burgiel, and Chaim Goodman-Strauss. 2016. *The symmetries of things.* CRC Press.

Thierry Delmarcelle and Lambertus Hesselink. 1994. The Topology of Symmetric, Second-Order Tensor Fields. *IEEE Computer Graphics and Applications* (1994), 140–147.

Yue Dong, Sylvain Lefebvre, Xin Tong, and George Drettakis. 2008. Lazy solid texture synthesis. *Computer Graphics Forum* 27, 4 (2008), 1165–1174.

David S. Ebert, Kenton F. Musgrave, Darwyn Peachey, Ken Perlin, and Steven Worley. 2002. *Texturing & Modeling: A Procedural Approach.* Morgan Kaufmann.

Matthew Fisher, Peter Schröder, Mathieu Desbrun, and Hugues Hoppe. 2007. Design of Tangent Vector Fields. *ACM Trans. Graph.* 26, 3, Article 56 (2007).

Ran Gal, Olga Sorkine, Tiberiu Popa, Alla Sheffer, and Daniel Cohen-Or. 2007. 3D collage: expressive non-realistic modeling. In *NPAR '07*. 7–14.

James Gregson, Alla Sheffer, and Eugene Zhang. 2011. All-Hex Mesh Generation via Volumetric PolyCube Deformation. *Comput. Graph. Forum* 30, 5 (2011), 1407–1416.

Alejo Hausner. 2001. Simulating decorative mosaics. In *SIGGRAPH '01*. 573–580.

Aaron Hertzmann. 1998. Painterly rendering with curved brush strokes of multiple sizes. In *SIGGRAPH '98*. 453–460.

Aaron Hertzmann and Denis Zorin. 2000. Illustrating smooth surfaces. In *SIGGRAPH '00*. 517–526.

Lambertus Hesselink, Yuval Levy, and Yingmei Lavin. 1997. The Topology of Symmetric, Second-Order 3D Tensor Fields. *IEEE Transactions on Visualization and Computer Graphics* 3, 1 (1997), 1–11.

Jin Huang, Yiying Tong, Hongyu Wei, and Hujun Bao. 2011. Boundary Aligned Smooth 3D Cross-frame Field. *ACM Trans. Graph.* 30, 6, Article 143 (2011), 8 pages.

Takashi Ijiri, Radomír Mech, Takeo Igarashi, and Gavin Miller. 2008. An Example-based Procedural System for Element Arrangement. *Computer Graphics Forum* 27, 2 (2008), 429–436.

Alec Jacobson. 2013. *Algorithms and interfaces for real-time deformation of 2d and 3d shapes.* Ph.D. Dissertation. ETH.

Ron Kimmel and James A Sethian. 1998. Computing geodesic paths on manifolds. *Proceedings of the national academy of Sciences* 95, 15 (1998), 8431–8435.

Johannes Kopf, Chi-Wing Fu, Daniel Cohen-Or, Oliver Deussen, Dani Lischinski, and Tien-Tsin Wong. 2007. Solid Texture Synthesis from 2D Exemplars. *ACM Trans. Graph.* 26, 3, Article 2 (2007).

Shahar Z. Kovalsky, Noam Aigerman, Ronen Basri, and Yaron Lipman. 2015. Large-scale Bounded Distortion Mappings. *ACM Trans. Graph.* 34, 6, Article 191 (2015), 10 pages.

Shahar Z. Kovalsky, Meirav Galun, and Yaron Lipman. 2016. Accelerated Quadratic Proxy for Geometric Optimization. *ACM Trans. Graph.* 35, 4, Article 134 (2016), 11 pages.

Ares Lagae and George Drettakis. 2011. Filtering Solid Gabor Noise. *ACM Trans. Graph.* 30, 4, Article 51 (2011), 6 pages.

Ares Lagae, Sylvain Lefebvre, Rob Cook, Tony DeRose, George Drettakis, D.S. Ebert, J.P. Lewis, Ken Perlin, and Matthias Zwicker. 2010. State of the Art in Procedural Noise Functions. In *Eurographics '10 State of the Art Report*. 2579–2600.

Ares Lagae, Sylvain Lefebvre, George Drettakis, and Philip Dutré. 2009. Procedural Noise Using Sparse Gabor Convolution. *ACM Trans. Graph.* 28, 3, Article 54 (2009), 10 pages.

Yu-Kun Lai, Miao Jin, Xuexiang Xie, Ying He, Jonathan Palacios, Eugene Zhang, Shi-Min Hu, and Xianfeng Gu. 2010. Metric-Driven RoSy Field Design and Remeshing. *IEEE Transactions on Visualization and Computer Graphics* 16, 1 (2010), 95–108.

Pierre-Edouard Landes, Bruno Galerne, and Thomas Hurtut. 2013. A Shape-Aware Model for Discrete Texture Synthesis. *Computer Graphics Forum* 32, 4 (2013), 67–76.

Bruno Lévy and Yang Liu. 2010. Lp Centroidal Voronoi Tessellation and Its Applications. *ACM Trans. Graph.* 29, 4, Article 119 (2010), 11 pages.

Wan-Chiu Li, Bruno Vallet, Nicolas Ray, and Bruno Lévy. 2006. Representing Higher-Order Singularities in Vector Fields on Piecewise Linear Surfaces. In *IEEE Transactions on Visualization and Computer Graphics (Proceedings Visualization '06)*.

Yufei Li, Yang Liu, Weiwei Xu, Wenping Wang, and Baining Guo. 2012. All-hex Meshing Using Singularity-restricted Field. *ACM Trans. Graph.* 31, 6, Article 177 (2012), 11 pages.

Chongyang Ma, Li-Yi Wei, and Xin Tong. 2011. Discrete Element Textures. *ACM Trans. Graph.* 30, 4, Article 62 (2011), 10 pages.

Ron Maharik, Mikhail Bessmeltsev, Alla Sheffer, Ariel Shamir, and Nathan Carr. 2011. Digital Micrography. *ACM Trans. Graph.* 30, 4, Article 100 (2011), 12 pages.

L. Markus. 1955. Line Element Fields and Lorentz Structures on Differentiable Manifolds. *Annals of Mathematics* 62, 3 (1955), pp. 411–417.

Matthias Nieser, Jonathan Palacios, Konrad Polthier, and Eugene Zhang. 2012. Hexagonal Global Parameterization of Arbitrary Surfaces. *IEEE Transactions on Visualization and Computer Graphics* 18, 3 (2012), 865–878.

Matthias Nieser, Ulrich Reitebuch, and Konrad Polthier. 2011. Cubecover–parameterization of 3d volumes. *Computer graphics forum* 30, 5 (2011), 1397–1406.

Victor Ostromoukhov and Roger D. Hersch. 1999. Multi-color and artistic dithering. In *SIGGRAPH '99*. 425–432.

Shigeru Owada, Frank Nielsen, Makoto Okabe, and Takeo Igarashi. 2004. Volumetric Illustration: Designing 3D Models with Internal Textures. *ACM Trans. Graph.* 23, 3 (2004), 322–328.

Jonathan Palacios, Chongyang Ma, Weikai Chen, Li-Yi Wei, and Eugene Zhang. 2016a. Tensor Field Design in Volumes. In *SIGGRAPH ASIA '16 Technical Briefs*. Article 18, 4 pages.

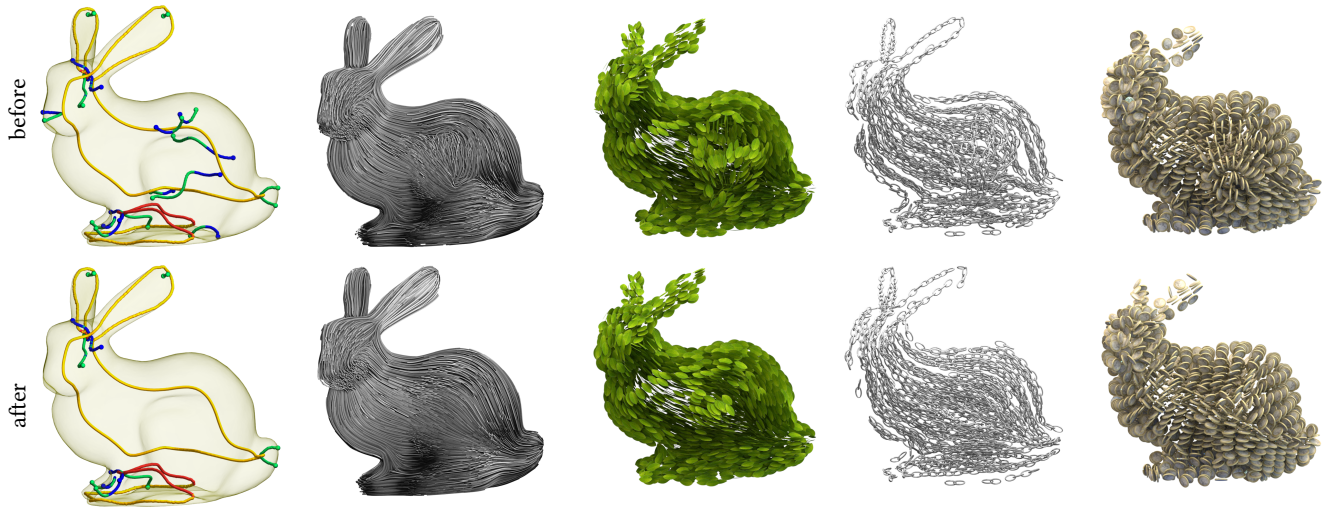


Fig. 21. *A degenerate curve removal example.* The top and bottom rows show respectively the tensor fields before and after the removal of some degenerate curves as visualized in (a) (degenerate curves only) and (b) (using streamtubes). The differences between the tensor fields are clearly reflected in the element synthesis results (c).

Jonathan Palacios, Harry Yeh, Wenping Wang, Yue Zhang, Robert S. Laramee, Ritesh Sharma, Thomas Schultz, and Eugene Zhang. 2016b. Feature Surfaces in Symmetric Tensor Fields Based on Eigenvalue Manifold. *IEEE Transactions on Visualization and Computer Graphics* 22, 3 (2016), 1248–1260.

Jonathan Palacios and Eugene Zhang. 2007. Rotational Symmetry Field Design on Surfaces. *ACM Trans. Graph.* 26, 3, Article 55 (2007).

Hans Pedersen and Karan Singh. 2006. Organic labyrinths and mazes. In *NPAR '06*. 79–86.

Nico Pietroni, Miguel A Otaduy, Bernd Bickel, Fabio Ganovelli, and Markus Gross. 2007. Texturing internal surfaces from a few cross sections. *Computer Graphics Forum* 26, 3 (2007), 637–644.

Emil Praun, Adam Finkelstein, and Hughes Hoppe. 2000. Lapped Textures. In *SIGGRAPH '00*. 465–470.

Emil Praun, Hugues Hoppe, Matthew Webb, and Adam Finkelstein. 2001. Real-time hatching. In *SIGGRAPH '01*. 581–.

Michael Rabinovich, Roi Poranne, Daniele Panozzo, and Olga Sorkine-Hornung. 2017. Scalable Locally Injective Mappings. *ACM Trans. Graph.* 36, 2, Article 16 (2017), 16 pages.

Nicolas Ray, Dmitry Sokolov, and Bruno Lévy. 2016. Practical 3D Frame Field Generation. *ACM Trans. Graph.* 35, 6, Article 233 (2016), 9 pages.

Nicolas Ray, Bruno Vallet, Laurent Alonso, and Bruno Levy. 2009. Geometry-aware Direction Field Processing. *ACM Trans. Graph.* 29, 1, Article 1 (2009), 11 pages.

Nicolas Ray, Bruno Vallet, Wan Chiu Li, and Bruno Lévy. 2008. N-symmetry direction field design. *ACM Trans. Graph.* 27, 2 (2008), 10:1–10:13.

Riccardo Roveri, A. Cengiz Öztireli, Sebastian Martin, Barbara Solenthaler, and Markus Gross. 2015. Example Based Repetitive Structure Synthesis. *Computer Graphics Forum* 34, 5 (2015), 39–52.

Christian Schüller, Ladislav Kavan, Daniele Panozzo, and Olga Sorkine-Hornung. 2013. Locally Injective Mappings. *Computer Graphics Forum (proceedings of EUROGRAPH-ICS/ACM SIGGRAPH Symposium on Geometry Processing)* 32, 5 (2013), 125–135.

Hang Si. 2015. TetGen, a Delaunay-Based Quality Tetrahedral Mesh Generator. *ACM Trans. Math. Softw.* 41, 2, Article 11 (2015), 36 pages.

Justin Solomon, Amir Vaxman, and David Bommes. 2017. Boundary Element Octahedral Fields in Volumes. *ACM Trans. Graph.* 36, 3, Article 28 (2017), 16 pages.

Jos Stam. 2003. Flows on Surfaces of Arbitrary Topology. *ACM Trans. Graph.* 22, 3 (2003), 724–731.

Kenshi Takayama, Makoto Okabe, Takashi Ijiri, and Takeo Igarashi. 2008. Lapped Solid Textures: Filling a Model with Anisotropic Textures. *ACM Trans. Graph.* 27, 3, Article 53 (2008), 9 pages.

Greg Turk. 2001. Texture Synthesis on Surfaces. In *SIGGRAPH '01*. 347–354.

Amir Vaxman, Marcel Campen, Olga Diamanti, David Bommes, Klaus Hildebrandt, Mirela Ben-Chen, and Daniele Panozzo. 2016. Directional Field Synthesis, Design, and Processing. In *SIGGRAPH ASIA '16 Courses*. Article 15, 30 pages.

Wolfram von Funck, Holger Theisel, and Hans-Peter Seidel. 2006. Vector Field Based Shape Deformations. *ACM Trans. Graph.* 25, 3 (2006), 1118–1125.

Li-Yi Wei, Sylvain Lefebvre, Vivek Kwatra, and Greg Turk. 2009. State of the art in example-based texture synthesis. In *Eurographics 2009, State of the Art Report, EG-STAR*. 93–117.

Li-Yi Wei and Marc Levoy. 2001. Texture Synthesis over Arbitrary Manifold Surfaces. In *SIGGRAPH 2001*. 355–360.

Jie Xu and Craig S. Kaplan. 2007. Calligraphic packing. In *GI '07*. 43–50.

Eugene Zhang, James Hays, and Greg Turk. 2007. Interactive Tensor Field Design and Visualization on Surfaces. *IEEE Transactions on Visualization and Computer Graphics* 13, 1 (2007), 94–107.

Eugene Zhang, Konstantin Mischaikow, and Greg Turk. 2006. Vector Field Design on Surfaces. *ACM Trans. Graph.* 25, 4 (2006), 1294–1326.

Guo-Xin Zhang, Song-Pei Du, Yu-Kun Lai, Tianyun Ni, and Shi-Min Hu. 2011. Sketch Guided Solid Texturing. *Graphical Models* 73, 3 (2011), 59–73.

Xiaoqiang Zheng and Alex Pang. 2004. Topological Lines in 3D Tensor Fields. In *VIS '04*. 313–320.

Xiaoqiang Zheng, Beresford Parlett, and Alex Pang. 2005a. Topological structures of 3D tensor fields. In *VIS '05*. 551–558.

Xiaoqiang Zheng, Beresford N. Parlett, and Alex Pang. 2005b. Topological Lines in 3D Tensor Fields and Discriminant Hessian Factorization. *IEEE Transactions on Visualization and Computer Graphics* 11, 4 (2005), 395–407.