# Computing a High-Dimensional Euclidean Embedding from an Arbitrary Smooth Riemannian Metric

ZICHUN ZHONG, Wayne State University
WENPING WANG, The University of Hong Kong
BRUNO LÉVY, INRIA Nancy - Grand Est
JING HUA, Wayne State University
XIAOHU GUO, University of Texas at Dallas

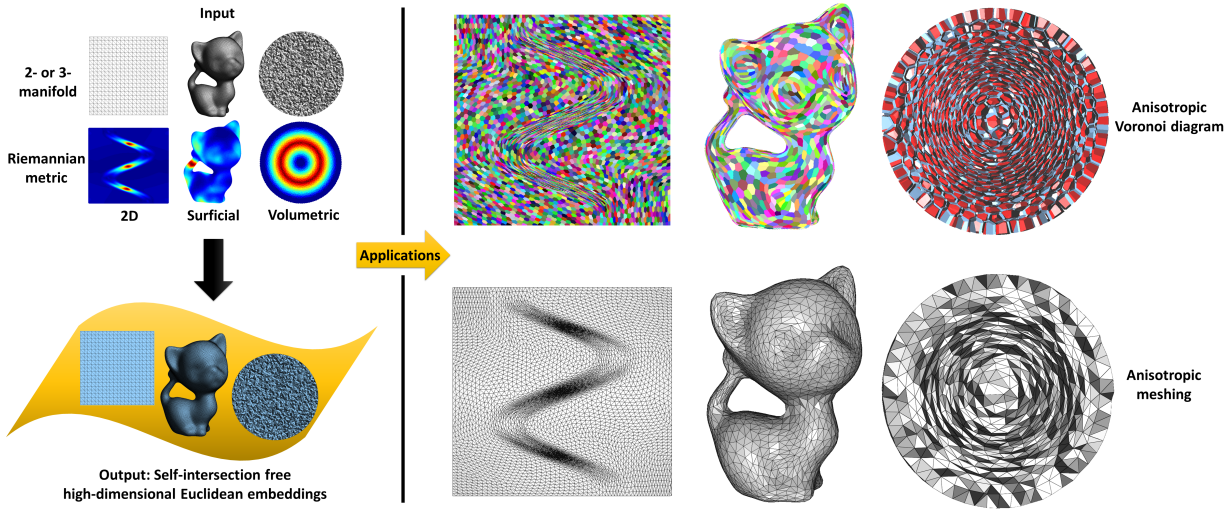


Fig. 1. Illustration of self-intersection free high-dimensional Euclidean embedding (SIFHDE$^2$) for anisotropic Restricted Voronoi Diagram and meshing on Riemannian 2- and 3-manifolds. Left: For some examples of 2- and 3-manifolds with Riemannian metric fields in different scenarios, such as metrics defined by a 2D analytic function, 3D surface curvature, and 3D analytic function, we can compute SIFHDE$^2$s of them in a high-dimensional Euclidean space. Right: Then, we can generate the Voronoi diagrams and meshes in such embedding with Euclidean metric. Finally, when they are mapped to the original domains, the anisotropic Voronoi diagrams (Top Right) and the anisotropic meshes (Bottom Right) will exhibit the desired anisotropy.

This article presents a new method to compute a *self-intersection free high-dimensional Euclidean embedding (SIFHDE$^2$)* for surfaces and volumes equipped with an arbitrary Riemannian metric. It is already known that given a high-dimensional (high-d) embedding, one can easily compute an anisotropic Voronoi diagram by back-mapping it to 3D space. We show here how to solve the inverse problem, i.e., given an input metric, compute a smooth intersection-free high-d embedding of the input such that the pullback metric of the embedding matches the input metric. Our numerical solution mechanism matches the deformation gradient of the 3D → higher-d mapping with the given Riemannian metric. We demonstrate the applicability of our method, by using it to construct anisotropic Restricted Voronoi Diagram (RVD) and anisotropic meshing, that are otherwise extremely difficult to compute. In SIFHDE$^2$-space constructed by our algorithm, difficult 3D anisotropic computations are replaced with simple Euclidean computations, resulting in an isotropic RVD and its dual mesh on this high-d embedding. Results are compared with the state-of-the-art in anisotropic surface and volume meshings using several examples and evaluation metrics.

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

Additional Key Words and Phrases: anisotropic Voronoi diagram, anisotropic meshing, high-dimensional (high-d) embedding, self-intersection free

**ACM Reference Format:**
Zichun Zhong, Wenping Wang, Bruno Lévy, Jing Hua, and Xiaohu Guo. 2018. Computing a High-Dimensional Euclidean Embedding from an Arbitrary Smooth Riemannian Metric. *ACM Trans. Graph.* 37, 4, Article 62 (August 2018), 16 pages. https://doi.org/10.1145/3197517.3201369

Authors' addresses: Zichun Zhong, Wayne State University, zichunzhong@wayne.edu; Wenping Wang, The University of Hong Kong, wenping@cs.hku.hk; Bruno Lévy, INRIA Nancy - Grand Est, bruno.levy@inria.fr; Jing Hua, Wayne State University, jinghua@wayne.edu; Xiaohu Guo, University of Texas at Dallas, xguo@utdallas.edu.


0730-0301/2018/8-ART62 $15.00
https://doi.org/10.1145/3197517.3201369

## 1 INTRODUCTION

Anisotropic meshes are important for not only improving the accuracy of numerical simulations [Alauzet and Loseille 2010; Narain et al. 2012] but also better approximating shapes [Simpson 1994].

Unlike isotropic meshes, where the elements are chosen to be as regular and uniform as possible, anisotropic meshes are designed with elongated mesh elements that follow user specified orientations and aspect ratios or Riemannian metric tensor fields. With the same number of budgeting elements, the anisotropic meshing representation for the surface / volume of the original model is preferred due to the smaller approximation errors compared with the correspondent isotropic counterpart [Heckbert and Garland 1999; Simpson 1994].

Anisotropic meshes are notoriously difficult to compute. A possible strategy to overcome the difficulty is to map the (complicated) anisotropic 3D space onto a higher-d space, where geometric computations are made simpler (Euclidean / isotropic). In a certain sense, "anisotropy is traded for additional dimensions". This idea is inspired by the celebrated Nash embedding theorem, which states that every Riemannian manifold can be isometrically embedded into some high-dimensional (high-d) Euclidean space [Kuiper 1955; Nash 1954]. In such high-d embedding space, the metric is uniform and isotropic. When the isotropic mesh is computed in this embedding space and back-projected, the mesh elements on the original manifold will exhibit the desired anisotropic property. Recently, a few methods investigated this problem on surface meshing. Zhong et al. [2013] introduced a particle-based meshing approach that conceptualizes the inter-particle energy optimization in a high-d "embedding space". This embedding space was used in their energy and force formulation without computing such a space explicitly. However, their method has some limitations due to lack of explicit embedding. Specifically, without an explicit embedding, K-Nearest Neighbors (K-NN) search of particles and anisotropic Restricted Voronoi Diagram (RVD) computation under a Riemannian metric space are very challenging and time-consuming. Another group of research focused on computing explicit embeddings, such as 3D embedding [Panozzo et al. 2014], 5D or 6D embedding [Dassi et al. 2014, 2015; Lévy and Bonneel 2012]. However, these methods have their own limitations, such as exhibiting self-intersections [Panozzo et al. 2014], working only on specific dimensions [Dassi et al. 2014, 2015; Lévy and Bonneel 2012; Panozzo et al. 2014], or using only specific embeddings (i.e., not considering arbitrary input Riemannian metrics) [Dassi et al. 2014, 2015; Lévy and Bonneel 2012].

The novelty introduced by our method is that our input is a smooth arbitrary user-defined metric, from which the embedding is deduced, whereas previous work observed that some predefined shapes of the embedding result in some specific anisotropies when back-mapped to 3D space. Our *main contribution* is to provide a numerical method that solves the "inverse problem". It makes our method much more general in terms of supported metrics. Besides that, previous work considered the surficial case, whereas we also experiment with the (significantly more challenging) *volume* meshing problem. Our algorithm works with arbitrary embedding dimension and avoids self-intersections. On the computed embedded surface or volume, we can directly optimize for a uniform, isotropic particle distribution and generate a high-quality isotropic RVD and its dual mesh on it. When mapped back to the original surface or volume, an anisotropic RVD as well as a simplicial mesh are obtained. Fig. 1 illustrates the method in flat 2D, 3D surficial and volumetric settings.

The embedding problem is formulated as an optimization that minimizes the deviation between the given metric and the *deformation gradient* of a map from the original surface / volume to the high-d embedded surface / volume. Our empirical experiments detailed further demonstrate that our method constructs an embedding with a small metric deviation error. The benefits of the high-d embedding are as follows:

- the so-constructed high-d space can be used to compute anisotropic Voronoi diagrams for surfaces and volumes with a prescribed anisotropy. To our knowledge, this is the first time that an effective computational algorithm is proposed to construct such objects in the arbitrary volumetric metric;
- it introduces new computational strategies for some recent anisotropic meshing algorithms, such as particle optimization, RVD, and dual meshing in the high-d Euclidean embedding space. Our empirical evaluations in several scenarios demonstrate that high-d embedding is a practical utility.

## 2 RELATED WORK

### 2.1 Anisotropic Triangulation

*Anisotropic Centroidal Voronoi Tessellation.* In order to compute well sampled anisotropic meshes, Du et al. [2005a] further generalized the concept of Centroidal Voronoi Tessellation (CVT) to the anisotropic case – Anisotropic CVT (ACVT), and then computed its dual mesh graph. An Anisotropic Voronoi Diagram (AVD) with the given Riemannian metric needs to be constructed in each Lloyd iteration [Lloyd 1982], which is a time-consuming operation. To make the computation faster, Valette et al. [2008] described a discrete approximation of ACVT by clustering the vertices of a dense pre-triangulation of the domain, at the expense of degraded mesh quality. Recently, Zhong et al. [2014] provided a method to solve the anisotropic meshing by conformally mapping the metric surface to an appropriate 2D parametric domain and then compute CVT on it. But the limitation of the conformal embedding method is that it cannot handle surfaces with complicated topologies.

*Surface Meshing in Higher Dimensional Space.* Several solutions were proposed for certain classes of surface meshing problems by embedding them in high-d spaces [Boissonnat et al. 2008a; Cañas and Gortler 2006; Kovacs et al. 2010; Lévy and Bonneel 2012]. Lévy and Bonneel [2012] extended the computation of CVT to a 6D space in order to achieve the curvature-adaptation. The main idea of their method is to transform the anisotropic meshing problem on a 3D surface to an isotropic one embedded in 6D space described by vertex positions and normals. Recently, Dassi et al. [2014; 2015] used Lévy and Bonneel's framework to compute remeshing, but instead of optimizing the CVT in 6D, they used local operations (i.e., edge flips). It should be noted that both Lévy and Bonneel's and Dassi et al.'s work does not provide user's flexibility to control the anisotropy through an arbitrary input metric field.

*Particle-Based Anisotropic Meshing.* Particle-based approaches for anisotropic meshing have been proposed during the past two decades. Bossen and Heckbert [1996] simulated the repulsion and attraction forces between particles based on a distance function with the metric tensor. Shimada et al. [1995; 1997] physically modeled the inter-bubble forces by a bounded cubic function of the distance, and

further extended it to anisotropic meshing by converting spherical bubbles to ellipsoidal ones. Both Bossen et al. and Shimada et al.'s work requires dynamic population control schemes, that is to adaptively insert or delete particles / bubbles in certain regions. Zhong et al. [2013] showed that by formulating the inter-particle energy optimization in a high-d space, such optimizations have very fast convergence without any need for the explicit control of particle population. But they still needed to compute the anisotropic RVD on the surface in 3D space in their final step of mesh generation, which is less robust and efficient. Another bottleneck in their framework is the search speed of neighboring particles. A comparison with our approach is shown in Sec. 6 and Appendix D.

*Refinement-Based Delaunay Triangulation.* Delaunay refinement-based approaches involve iterative point insertion in a Delaunay triangulation. Many researchers have extended these approaches to anisotropic triangulations [Borouchaki et al. 1997b,a; Dobrzynski and Frey 2008]. Cheng et al. [2001; 2006] applied Delaunay refinement to anisotropic surface meshing. Boissonnat et al. [2015a; 2008b; 2015b] introduced a Delaunay refinement framework to add new vertices gradually to reach the final anisotropic meshes. Recently, Rouxel-Labbé et al. [2016] introduced an algorithm to compute discrete approximations of Riemannian Voronoi diagrams on 2-manifolds by using the numerical computation of geodesic distances. The main strength of these methods is that they come with theoretical proofs, both on termination (finite number of inserted points) and existence of a straight dual. However, since they use *local* criteria to insert points, the final result is often of lower quality than when using a *global* optimization [Fu et al. 2014; Zhong et al. 2013]. Since they used the same representation of the metric as in [Lévy and Bonneel 2012] and as us (piecewise constant in the simplices), their method can be used as a *post-processing* after a global optimization, in order to benefit from both advantages (quality of global optimization and robustness / theoretical guarantees of Delaunay refinement).

*Optimal Delaunay Triangulation.* Another idea for anisotropic meshing is through Delaunay-based variational approach, by minimizing the error between a lifted quadratic target function and the linear interpolation of the constructed mesh. Chen et al. [2007; 2004] proposed Optimal Delaunay Triangulation (ODT) method to minimize the error function for both isotropic and anisotropic meshing. Alliez et al. [2005] presented a robust approach for isotropic tetrahedral meshing by exploiting the ODT formalism. In the context of varying densities, the objective function was carefully analyzed and improved in [Chen et al. 2014]. ODT can be considered as lifting the points onto a paraboloid and estimating volume integrals on this paraboloid. Budninskiy et al. [2016] presented a variational method to take into account an anisotropy by using a *convex* function instead of a paraboloid. While elegant and interesting from a theoretical point of view, their method is limited to a small class of anisotropies stemming from convex functions. The constraint is not reasonable in practical applications (for instance, it could not account for the anisotropy in Fig. 1-Right, nor with any non-convex geometry). Fu et al. [2014] proposed a Locally Convex Triangulation (LCT) method to compute the anisotropic meshes based on constructing convex functions that locally match the specified Riemannian metric. We compare the quality of our generated mesh with Fu et al.'s method in Sec. 6.

## 2.2 Anisotropic Tetrahedralization

Theoretically, it is possible to extend ACVT-based method for anisotropic meshing from 2D domain or surface [Du and Wang 2005a; Valette et al. 2008; Zhong et al. 2014] into 3D volume, but this group of methods needs to compute AVD iteratively in the ambient 3D space with specified metrics, which makes the computation very complicated and impractical. In computer graphics and geometric modeling areas, there is few literature in anisotropic tetrahedral meshing: Dobrzynski and Frey's local mesh adaptation method based on an extension of Delaunay kernel [Dobrzynski and Frey 2008], Klingner et al.'s aggressive optimization [Klingner 2008; Klingner and Shewchuk 2007] and Fu et al.'s LCT method [Fu et al. 2014] on some simple 3D models. Yamakawa and Shimada [2000] proposed an ellipsoidal bubble packing method, but inserting or deleting particles / bubbles is necessary in the computation. In mechanical and biomedical engineering, anisotropic meshing is widely used in computational fluid dynamics (CFD) [Alauzet and Lozeille 2016; Frey and Alauzet 2005; Loseille and Löhner 2016] and blood flow simulations [Marchandise et al. 2013; Sauvage et al. 2014]. However, their methods are numerical and error estimation-based approaches. The complicated algorithms and implementations are becoming the major bottleneck and impeding the effective utilization and better understanding of acquired 3D anisotropic volume models.

## 2.3 Other Related Embedding Work

In the mathematical community, there are many theoretical work on isometric embedding for Riemannian manifolds [Gromov 2017; Gromov and Rokhlin 1970; Han and Hong 2006; Hong 1993]. Recently, Borrelli et al. [Borrelli et al. 2012] proposed to convert the convex integration theory into an algorithm and implementation that produces isometric maps of flat tori. Bronstein et al. [2000; 2005] demonstrated the performance of multi-dimensional scaling (MDS). Instead of raising the dimensions, they mapped the original metric domain into a parameter domain (similar to a conformal embedding). Verma [2012] proposed the distance preserving embeddings for general n-dimensional manifolds in the machine learning field. It applies "spiraling perturbations / corrections" for globally isometric mapping. This cannot compute a smooth high-d embedding for our computer graphics research and applications. Recently, Panozzo et al. [2014] proposed a 3D embedding framework with self-intersections to compute a surface deformation that warps a frame field into a cross field, and used it for the adaptive quad meshing. In their method, the parameterization is used to handle the final meshing with self-intersections. However, our proposed method does not need an extra step to parameterize the embedding for meshing, since we can directly produce the Voronoi diagrams and meshes on the computed SIFHDE$^2$. Another limitation of their method is that the frame-driven deformation for 3D embedding computation may deteriorate the quality of the input triangulation, especially when the stretching ratio is high, such as $\geq 10$, which leads to failure. However, our method theoretically can compute the embedding for any high-stretching ratio case, since we provide enough degrees of freedom in higher dimensions. We compare the embedding quality

with Panozzo et al.'s method in Sec. 6. Besides surface cases, our high-d embedding approach can work on volume cases, which is an unexplored topic in the previous work.

## 3 HIGH-D EMBEDDING

### 3.1 Anisotropy and High-D Embedding

Anisotropy represents how distances and angles are distorted, which can be measured by the dot product in geometry. We consider that a metric $\mathcal{M}(.)$, i.e., a symmetric positive definite (SPD) bilinear form, is defined over the surface or volume domain $\Omega \subset \mathbb{R}^m$. In this case, at a given point $\mathbf{x} \in \Omega$, the *dot product* between two vectors $\mathbf{a}$ and $\mathbf{b}$ is denoted by $\langle \mathbf{a}, \mathbf{b} \rangle_{\mathcal{M}(\mathbf{x})}$, which is defined over the tangent space of the surface or the volume:

$$\langle \mathbf{a}, \mathbf{b} \rangle_{\mathcal{M}(\mathbf{x})} = \mathbf{a}^T \mathbf{M}(\mathbf{x}) \mathbf{b}, \tag{1}$$

where a symmetric $m \times m$ matrix $\mathbf{M}(\mathbf{x})$ represents the metric. Through Singular Value Decomposition (SVD), the metric matrix $\mathbf{M}(\mathbf{x})$ can be decomposed into:

$$\mathbf{M}(\mathbf{x}) = \mathbf{R}(\mathbf{x})^T \mathbf{S}(\mathbf{x})^2 \mathbf{R}(\mathbf{x}), \tag{2}$$

where the diagonal matrix $\mathbf{S}(\mathbf{x})^2$ contains its ordered eigenvalues (i.e., a scaling field), and the orthogonal matrix $\mathbf{R}(\mathbf{x})$ contains its eigenvectors (i.e., a rotation field). In the surface case [Du and Wang 2005a; Zhong et al. 2013], they are defined on the tangent spaces of the surface. $\frac{s_2}{s_1}$ is defined as *stretching ratio* in the surface metric field $\mathbf{M}(\mathbf{x})$, where $s_1$ and $s_2$ are the two diagonal items in $\mathbf{S}(\mathbf{x})$ and $s_1 \leq s_2$. Similarly, in volume case [Yamakawa and Shimada 2000], they represent the scaling and rotation in a 3D volume space.

*Metric through High-D Embedding.* For an arbitrary metric field $\mathbf{M}(\mathbf{x})$ defined on the surface or volume $\Omega \subset \mathbb{R}^m$ (i.e., Riemannian 2- or 3-manifold), Nash theorem [Nash 1954] states that there exists a high-d space $\mathbb{R}^{\overline{m}}$ (i.e., $m \leq \overline{m}$), in which the surface or volume can be embedded with Euclidean metric as $\overline{\Omega} \subset \mathbb{R}^{\overline{m}}$. In this article, $\overline{\Omega}$ denotes the "embedded surface or volume". Consider a mapping denoted by $\phi : \Omega \rightarrow \overline{\Omega}$. For a simple example of Fig. 2, the left image shows a 2D metric field $\mathbf{M}$. The unit circles and bean-shaped curves are iso-distance contours, and the colormap represents the stretching ratio. The right image shows that by embedding the flat 2D domain as a curved surface in 3D, one can recast the Riemannian manifold problem (such as in 2D) as the isotropic Euclidean manifold of a shape surface embedded in a higher dimensional space (such as in 3D). This theorem does also hold in the higher dimensions, other than 2D. In this work, we focus on Riemannian 2- and 3-manifold shapes.

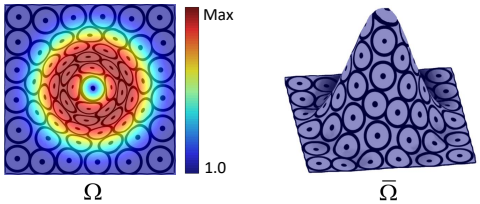


Fig. 2. The original 2D Riemannian manifold surface $\Omega$ (left) embedded in a higher dimensional space $\overline{\Omega}$ (right) (3D).

From the definition of anisotropy at the beginning of this section and the above Eq. (1), it can be seen that introducing anisotropy means changing the definition of the dot product. If we consider two vectors $\mathbf{a}$ and $\mathbf{b}$ in the tangent space of a given location $\mathbf{x} \in \Omega$, then they are transformed into $\overline{\mathbf{a}} = \mathbf{J}(\mathbf{x})\mathbf{a}$ and $\overline{\mathbf{b}} = \mathbf{J}(\mathbf{x})\mathbf{b}$ on the high-d embedded surface or volume $\overline{\Omega}$, where $\mathbf{J}(\mathbf{x})$ denotes the Jacobian matrix of mapping $\phi$ at $\mathbf{x}$. The dot product between $\overline{\mathbf{a}}$ and $\overline{\mathbf{b}}$ is given by the *pullback metric* of $\phi$, defined as: $\langle \overline{\mathbf{a}}, \overline{\mathbf{b}} \rangle = \mathbf{a}^T \mathbf{J}(\mathbf{x})^T \mathbf{J}(\mathbf{x}) \mathbf{b} = \mathbf{a}^T \mathbf{M}(\mathbf{x}) \mathbf{b}$.

*Importance of High-D Embedding.* (1) In a high-d space, more degrees of freedom are available to deform and embed the given surface or volume, so that better embedding quality is obtained. Experiments are shown in Sec. 6. (2) When using the high-d space, we can avoid self-intersections of the embedded surface or volume as discussed in Sec. 3.4.2, instead of embedding them in the original space. (3) By generating a high-d Euclidean embedding without self-intersections, we are able to drastically simplify several Riemannian geometric applications, such as computing anisotropic RVD and meshing on surface and volume with high-quality elements using only isotropic Euclidean computations. This is both more efficient and more accurate than the computations in metric space. Details are presented in Sec. 4.

### 3.2 High-D Embedding Transformation

In this work, we assume the surfaces are all represented as triangle meshes and the volumes are all represented as tetrahedron meshes. The two surfaces / volumes $\Omega$ (in the original space) and $\overline{\Omega}$ (in the high-d space) share the same number of vertices and the same connectivity between vertices. The only difference between them is their vertex coordinates and dimensions; thus, we assume that the two surfaces / volumes have the same indices of vertices and triangles / tetrahedrons. In the following subsection, we discuss the surface and volume cases, respectively.

*Surface:* For a triangle $j$ on $\Omega$ and $\overline{\Omega}$, let $\{\mathbf{v}_{j_1}, \mathbf{v}_{j_2}, \mathbf{v}_{j_3}\}$ and $\{\overline{\mathbf{v}}_{j_1}, \overline{\mathbf{v}}_{j_2}, \overline{\mathbf{v}}_{j_3}\}$ denote their vertices, respectively. A basis to the tangent space is given by their corresponding edge vectors, which can be represented as the following two matrices $\mathbf{W}_j = [\mathbf{v}_{j_2} - \mathbf{v}_{j_1}, \mathbf{v}_{j_3} - \mathbf{v}_{j_1}]$ and $\overline{\mathbf{W}}_j = [\overline{\mathbf{v}}_{j_2} - \overline{\mathbf{v}}_{j_1}, \overline{\mathbf{v}}_{j_3} - \overline{\mathbf{v}}_{j_1}]$.

*Volume:* For a tetrahedron $j$ on $\Omega$ and $\overline{\Omega}$, let $\{\mathbf{v}_{j_1}, \mathbf{v}_{j_2}, \mathbf{v}_{j_3}, \mathbf{v}_{j_4}\}$ and $\{\overline{\mathbf{v}}_{j_1}, \overline{\mathbf{v}}_{j_2}, \overline{\mathbf{v}}_{j_3}, \overline{\mathbf{v}}_{j_4}\}$ denote their vertices, respectively. Similar to the surface case, the corresponding edge vectors can be represented as the following two matrices $\mathbf{W}_j = [\mathbf{v}_{j_2} - \mathbf{v}_{j_1}, \mathbf{v}_{j_3} - \mathbf{v}_{j_1}, \mathbf{v}_{j_4} - \mathbf{v}_{j_1}]$ and $\overline{\mathbf{W}}_j = [\overline{\mathbf{v}}_{j_2} - \overline{\mathbf{v}}_{j_1}, \overline{\mathbf{v}}_{j_3} - \overline{\mathbf{v}}_{j_1}, \overline{\mathbf{v}}_{j_4} - \overline{\mathbf{v}}_{j_1}]$.

Their relationship can be represented as:

$$\overline{\mathbf{W}}_j = \mathbf{J}_j \mathbf{W}_j, \tag{3}$$

where $\mathbf{J}_j$ is the Jacobian transformation matrix for triangle or tetrahedron $j$, and $\mathbf{J}_j^T \mathbf{J}_j = \mathbf{M}_j$.

In what follows, we assume that the original metric is smooth over the surface or the volume (satisfying a Lipschitz condition [Freidlin 1968; Itô 1950]), and sampled as a constant symmetric matrix $\mathbf{M}_j$ attached to each mesh simplex $j$. Note that in practice, it may be specified in the input at each vertex. In this case, several techniques can be used to interpolate the metric and deduce a reasonable constant value on each simplex (see [Courty et al. 2006] for a discussion).

We now elaborate on the relationship between the Jacobian matrix of the mapping $\mathbf{J}_j$ and the metric $\mathbf{M}_j$, i.e., $\mathbf{J}_j^T \mathbf{J}_j = \mathbf{M}_j$. To better

understand $\mathbf{J}_j$, it is necessary to explore the relationship between the original triangle or tetrahedron $j$ on $\Omega$ and the embedded triangle or tetrahedron $j$ on $\overline{\Omega}$, $\mathbf{J}_j$ is an $\overline{m}\times m$ matrix, and it can be represented as the product of a rotation in the high-d embedding space, and a scaling and rotation in the original space. Based on Eq. (3), that is:

$$\overline{\mathbf{W}}_j = \overline{\mathbf{U}}_j \begin{bmatrix} \mathbf{S}_j\mathbf{R}_j \\ \mathbf{0} \end{bmatrix} \mathbf{W}_j, \tag{4}$$

where $\overline{\mathbf{U}}_j$ is an $\overline{m} \times \overline{m}$ unitary matrix (i.e., a rotation matrix in $\mathbb{R}^{\overline{m}}$); $\mathbf{0}$ is a $(\overline{m} - m) \times m$ block of zeros; $\mathbf{S}_j$ and $\mathbf{R}_j$ are the diagonal scaling matrix and rotation matrix extracted from SVD of the metric $\mathbf{M}_j$ as shown in Eq. (2).

Thus, the embedding transformation $\mathbf{J}_j$ is:

$$\mathbf{J}_j = \overline{\mathbf{U}}_j \begin{bmatrix} \mathbf{S}_j\mathbf{R}_j \\ \mathbf{0} \end{bmatrix}. \tag{5}$$

By denoting $\mathbf{Q}_j = \begin{bmatrix} \mathbf{S}_j\mathbf{R}_j \\ \mathbf{0} \end{bmatrix}$, we can simply represent $\mathbf{J}_j$ as:

$$\mathbf{J}_j = \overline{\mathbf{U}}_j\mathbf{Q}_j. \tag{6}$$

### 3.3 High-D Deformation Gradient

Intuitively, the transformation between the original surface or volume $\Omega$ and its high-d embedded one $\overline{\Omega}$ can be considered as the deformation of $\Omega$. It is represented by the field of the *deformation gradient* over the surface or volume as intuitively shown by Sumner and Popović [2004]. This field is defined per triangle face or tetrahedron volume. We extend their idea from 3D surface to our high-d embedding as well as volume cases. Given a manifold with no deformation (i.e., the original surface or volume $\Omega$) and its corresponding deformed manifold (i.e., the embedded surface or volume $\overline{\Omega}$), the deformation gradient $\mathbf{T}_j$ for triangle or tetrahedron $j$ ($\mathbf{T}_j$ is an $\overline{m}\times m$ matrix) can be defined by:

$$\mathbf{T}_j\mathbf{W}_j = \overline{\mathbf{W}}_j. \tag{7}$$

In order to obtain $\mathbf{T}_j$, we need to compute the inverse of $\mathbf{W}_j$. Since the surface and volume have different representations of edge vectors, there are two cases as follows:

*(1) Surface:* For surfaces, $\mathbf{W}_j$ is a 3×2 matrix, so we cannot directly compute its inverse matrix. With QR factorization of $\mathbf{W}_j$ [Golub and Loan 1996] (equivalent to the pseudoinverse):

$$\mathbf{W}_j = \mathbf{O}_j \begin{bmatrix} \mathbf{P}_j \\ \mathbf{0} \end{bmatrix} = \begin{bmatrix} \mathbf{O}_{j\alpha}\mathbf{O}_{j\beta} \end{bmatrix} \begin{bmatrix} \mathbf{P}_j \\ \mathbf{0} \end{bmatrix} = \mathbf{O}_{j\alpha}\mathbf{P}_j, \tag{8}$$

where $\mathbf{P}_j$ is a 2×2 upper triangular matrix, $\mathbf{O}_j$ is an $m\times m$ orthogonal matrix, $\mathbf{O}_{j\alpha}$ is an $m \times 2$ matrix, and $\mathbf{O}_{j\beta}$ is an $m \times (m-2)$ matrix, where $m = 3$. Then we have the minimum norm solution of the deformation gradient for triangle $j$:

$$\mathbf{T}_j = \overline{\mathbf{W}}_j\mathbf{P}_j^{-1}\mathbf{O}_{j\alpha}^T. \tag{9}$$

*(2) Volume:* For volumes, $\mathbf{W}_j$ is a 3×3 square matrix, so the inverse of it can be directly computed as:

$$\mathbf{T}_j = \overline{\mathbf{W}}_j\mathbf{W}_j^{-1}. \tag{10}$$

### 3.4 Embedding Optimization

From Eq. (6) and Eq. (9) (or Eq. (10)), we can clearly see that in essence, the high-d embedding transformation $\mathbf{J}_j$ and the high-d deformation gradient $\mathbf{T}_j$ are the same. For $\mathbf{J}_j$ in Eq. (6), $\mathbf{Q}_j$ is coming from the given metric $\mathbf{M}_j$ on the surface or volume, while $\overline{\mathbf{U}}_j$ is the high-d rotation matrix that is unknown. For $\mathbf{T}_j$ in Eq. (9), $\mathbf{P}_j$ and $\mathbf{O}_{j\alpha}$ are from the edge vector $\mathbf{W}_j$ of the original surface, or in Eq. (10), $\mathbf{W}_j^{-1}$ is from the edge vector $\mathbf{W}_j$ of the original volume, while $\overline{\mathbf{W}}_j$ is the edge vector of the high-d embedded triangle or tetrahedron that is unknown.

Thus, knowing either of these two matrices, i.e., $\mathbf{J}_j$ and $\mathbf{T}_j$, guides us in computing the other. We can formulate an expression to minimize the function $E_{em}$ defined as the difference between these two matrices:

$$E_{em}(\overline{\mathbf{v}}_1, \cdots, \overline{\mathbf{v}}_{n_v}) = \min \sum_{j=1}^{n_{ele}} \left\| \mathbf{T}_j - \mathbf{J}_j \right\|_F^2, \tag{11}$$

where $n_v$ is the number of vertices, $n_{ele}$ is the number of mesh elements, i.e., triangles or tetrahedrons, and $\{\overline{\mathbf{v}}_1, \cdots, \overline{\mathbf{v}}_{n_v}\}$ are the vertices of the high-d embedded surface or volume $\overline{\Omega}$. The matrix norm $||.||_F$ is the Frobenius norm.

By substituting Eq. (6) and Eq. (9) (or Eq. (10)) into Eq. (11), we can get the more detailed expressions for this objective function:

*Surface:*

$$E_{em}(\overline{\mathbf{v}}_1, \cdots, \overline{\mathbf{v}}_{n_v}) = \min \sum_{j=1}^{n_{ele}} \left\| \overline{\mathbf{W}}_j\mathbf{P}_j^{-1}\mathbf{O}_{j\alpha}^T - \overline{\mathbf{U}}_j\mathbf{Q}_j \right\|_F^2. \tag{12}$$

*Volume:*

$$E_{em}(\overline{\mathbf{v}}_1, \cdots, \overline{\mathbf{v}}_{n_v}) = \min \sum_{j=1}^{n_{ele}} \left\| \overline{\mathbf{W}}_j\mathbf{W}_j^{-1} - \overline{\mathbf{U}}_j\mathbf{Q}_j \right\|_F^2. \tag{13}$$

*3.4.1 Regularization Term.* It is well known that solutions to Nash isometric embedding problem often present wrinkles, known as corrugations, that can form fractal patterns (see [Borrelli et al. 2012] and references herein on convex integration). Thus, a direct minimization of the criterion in Eq. (12) or Eq. (13) may lead to an oscillating high-d embedding, with wrinkles (an example shown in Fig. 3), that may have an influence on the stability / robustness of the subsequent geometric operations: in the high-d embedding space, we would like to use the Euclidean distance to approximate the metric distance on the original surface or volume, in order to efficiently compute: (1) K-NN for particle optimization in the high-d space; and (2) high-d RVD and dual mesh based on uniformly optimized particles. Thus the embedded surface or volume has to be smooth enough, in order to ensure that the K-NN and RVD computed with Euclidean distances in the high-d space are sufficiently accurate.

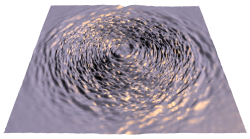

Fig. 3. A 3D embedding from a 2D domain with an anisotropic metric given in Fig. 4 (a) computed without the regularization.

Thus, to regularize the embedding, we add the following term:

$$E_{reg}(\overline{\mathbf{v}}_1, \cdots, \overline{\mathbf{v}}_{n_v}) = \sum_{i=1}^{n_{v-b}} \sum_{d=3,4}^{\overline{m}} \left( \frac{\sum_{k\in N(i)} (\overline{v}_k^d - \overline{v}_i^d)}{|N(i)|} \right)^2, \tag{14}$$

where $n_{v-b}$ is the total number of vertices excluding those on the open boundaries in surface case or boundary surface in volume case, $N(i)$ is the set of one-ring neighbors of vertex $i$, $|N(i)|$ is the size of set $N(i)$, and $\overline{v}_i^d$, $\overline{v}_k^d$ are the $d$-th dimensional coordinates of vertices $\overline{\mathbf{v}}_i$ and $\overline{\mathbf{v}}_k$ (why $d$ starts from 3 or 4 will be explained in the

following Sec. 3.4.2). The regularity term $E_{reg}$ is a summation of the square of graph Laplacian operations over every vertex in the embedding space except those vertices on the open boundaries in surface case or boundary surface in volume case.

Now our regularized objective function for the embedding optimization includes two terms: the similarity between two transformations and the regularity used to achieve smoothness of the embedding:

$$E_{total} = E_{em} + \mu E_{reg}, \tag{15}$$

where $\mu$ is a weighting factor to balance the similarity and regularity terms during optimization. In our experiments, $\mu$ is set to be 100. The experiments for analyzing the behaviors of different $\mu$ values with embedding results are given in Appendix B.

*3.4.2 Avoiding Intersections and Choosing Target Dimension.* According to Nash embedding theorem, using the mapping $\phi : \Omega \rightarrow \overline{\Omega}$ given by $\mathbf{v}^{1:m} \rightarrow (\mathbf{v}^{1:m}, \overline{\mathbf{v}}^{m+1}, \cdots, \overline{\mathbf{v}}^{\overline{m}})$, one obtains an embedding without self-intersections. In our case, to avoid self-intersections that may come from numerical approximations or from the regularization term, in addition we keep the original 3D coordinates, thus, in a certain sense, the additional $n$D coordinates that we compute act as "correcting terms" to solve for the metric. Clearly, if the original 3D surface is free of self-intersections, then it is also the case of our ($n$+3)D embedding. Compared with methods that avoid self-intersections a-posteriori in a 3D embedding [Panozzo et al. 2014], this is both simpler, and leaves more degrees of freedom to fit the user-specified metric. Generally speaking, the higher the dimension is, the smaller the embedding error is, since more degrees of freedom are provided to deform and embed the given surface or volume in such an embedding space; it is necessary to choose the dimension in a way that balances the computational efficiency and the embedding errors. Another advantage of the proposed high-d embedding is that we can automatically preserve the original 2D / 3D shape geometric features (such as sharp feature edges and corners) by using the strategy of keeping the original 2D / 3D coordinates, and only embedding and smoothing additionally higher dimensions ($d$ starts from 3 or 4 in the regularization term), i.e., "anisotropic metric is traded as additional dimensions". Our empirical results showed that 8D is a reasonable target dimension for embedding most general metric surfaces and volumes in small errors and without any self-intersection. More details and empirical validation will be given in Sec. 6.1.2.

*3.4.3 Numerical Solution Mechanism.* The optimization defined above is a non-linear problem with two unknown parameters (i.e., $\overline{\mathbf{W}}_j$ and $\overline{\mathbf{U}}_j$). We use an iterative method to obtain the optimal solution of $\overline{\mathbf{W}}_j$. It should be noted that the optimized vertex coordinates $\overline{\mathbf{v}}_1, \cdots, \overline{\mathbf{v}}_{n_v}$ of the embedded surface or volume are included in the edge vectors $\overline{\mathbf{W}}_j$.

Initially, for each vertex $i$, we set its high-d embedding coordinates as $\overline{\mathbf{v}}_i = (\mathbf{v}_i^{1:m}, \overline{v}_i^{m+1}, \cdots, \overline{v}_i^{\overline{m}})$, where the first $m$ dimensions are the same as the original surface or volume's coordinates, and use a small random perturbation to each higher dimensional coordinate (from dimension $m+1$ to dimension $\overline{m}$).

When the vertex coordinates $\{\overline{\mathbf{v}}_i | i = 1, \cdots, n_v\}$ are fixed (i.e., all $\overline{\mathbf{W}}_j$s are fixed), the optimal $\overline{\mathbf{U}}_j$ can be computed as the orthogonal factor of the following Polar decomposition:

*Surface:*

$$\overline{\mathbf{U}}_j = Polar(\overline{\mathbf{W}}_j \mathbf{P}_j^{-1} \mathbf{O}_{j\alpha}^T \mathbf{Q}_j^T). \tag{16}$$

*Volume:*

$$\overline{\mathbf{U}}_j = Polar(\overline{\mathbf{W}}_j \mathbf{W}_j^{-1} \mathbf{Q}_j^T). \tag{17}$$

The detailed derivations are given in the supplementary material (Appendix A).

When all $\overline{\mathbf{U}}_j$s are fixed, we can compute the optimal vertex coordinates $\{\overline{\mathbf{v}}_i | i = 1, \cdots, n_v\}$, where $\overline{\mathbf{v}}_i^{1:3} = \mathbf{v}_i^{1:3}$, by solving a linear system, since the objective function $E_{total}$ is a quadratic form of the vertex coordinates. This is similar to the method used by Sumner and Popović [2004].

Our goal is to ensure that the regularized objective function always decreases during optimization. The optimization strategy is similar to [Panozzo et al. 2014; Sorkine-Hornung and Alexa 2007]. In summary, as shown in the following Alg. 1, we can optimize the vertex coordinates $\overline{\mathbf{v}}_1, \cdots, \overline{\mathbf{v}}_{n_v}$ of the embedded surface or volume iteratively, until convergence by satisfying a specified stopping condition, e.g., the magnitude of the gradient is smaller than a threshold or the desired number of iterations is reached. In our experiments, we use the number of iterations as 50 for the stopping condition. One example of the evolution of the objective function during optimization is given in Fig. 4 (b), which can reach a satisfactory small embedding error. It is noted that we optimize Eq. (15) with a regularization smoothness term, instead of Eq. (12) and Eq. (13). Not only it forces the embedding to be $C^2$ (with the graph Laplacian regularization), but also it smoothens, thus it prevents the energy of the objective function from reaching zero.

**Data:** Original surface or volume $\Omega$ with user-specified metric $\mathbf{M}$ and target dimension $\overline{m}$
**Result:** Vertex coordinates $\{\overline{\mathbf{v}}_i | i = 1, \cdots, n_v\}$ of the embedded surface or volume $\overline{\Omega}$ in $\overline{m}$ space
Initialize vertex coordinates in $\overline{m}$ space;
**while** *stopping condition not satisfied* **do**
  **for** *each triangle or tetrahedron j* **do**
    Compute $\overline{\mathbf{U}}_j$ by fixing $\{\overline{\mathbf{v}}_i | i = 1, \cdots, n_v\}$ and using Eq. (16) or Eq. (17);
  **end**
  Compute $\{\overline{\mathbf{v}}_i | i = 1, \cdots, n_v\}$ by fixing all $\overline{\mathbf{U}}_j$s and fixing $\overline{\mathbf{v}}_1$, i.e., solve a linear system to minimize the regularized objective function $E_{total}$ of Eq. (15);
**end**

**Algorithm 1:** High-D Embedding Optimization.

## 4 ANISOTROPIC RVD AND MESHING

In order to demonstrate the importance and usefulness of the proposed SIFHDE$^2$, we investigate two applications, i.e., computing anisotropic RVD and anisotropic triangular / tetrahedral meshing.

Under the isotropic metric space, we can use the particle optimization technique (Sec. 4.1) to compute isotropic RVD and its dual Delaunay triangulation on the high-d embedded surface or volume (Sec. 4.2), since it is efficient to generate regular hexagonal patterns

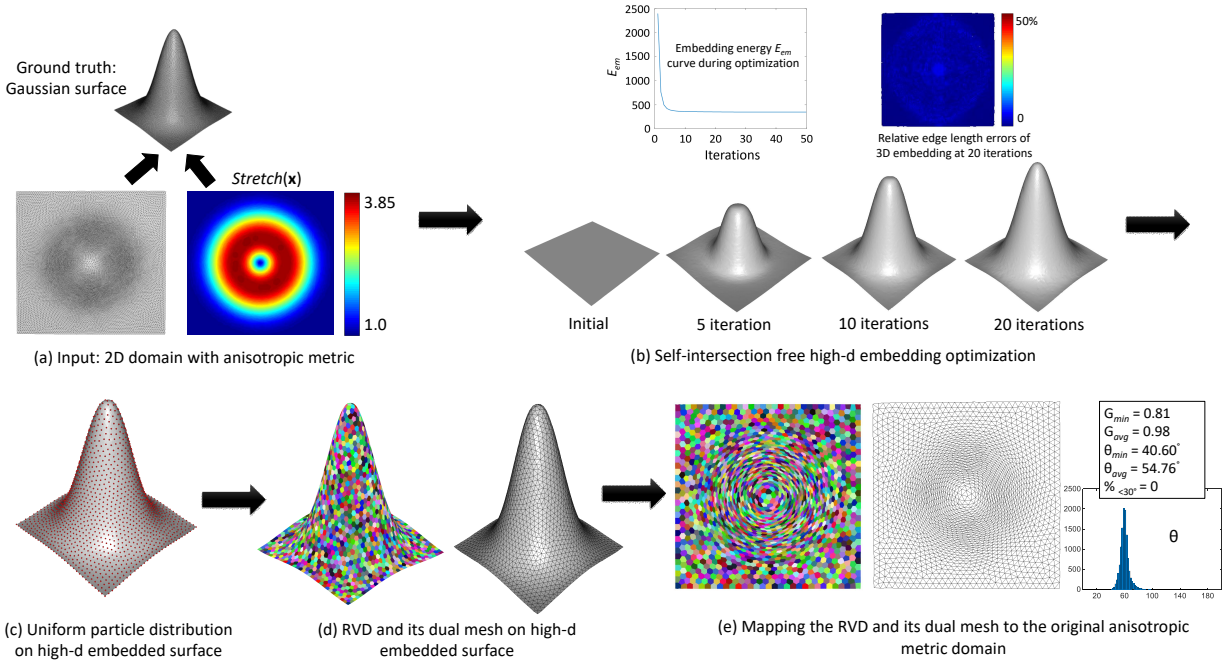


Fig. 4. Illustration of the proposed high-d embedding for anisotropic RVD and surface meshing. (a) A simple example of a 2D domain equipped with anisotropic metric, which has an ideal 3D embedding – the Gaussian surface as the ground truth. (b) Our embedding optimization iteratively deforms the original domain into the embedded surface (Sec. 3). After embedding we can (c) uniformly and isotropically sample particles (Sec. 4.1) and (d) compute the isotropic RVD and its dual mesh on the embedded surface (Sec. 4.2). (e) When they are mapped to the original 2D domain, the RVD and the triangular mesh will exhibit the desired anisotropy (Sec. 4.2).

of particles on 2-manifold and regular body-centered-cubic (BCC) lattices on 3-manifold, which are similar to the results of CVT [Du et al. 1999; Du and Wang 2005b; Liu et al. 2009]. Moreover, the particle-based method is easily formulated in high-d space. Finally the generated RVD and triangulation / tetrahedralization can be mapped from the high-d embedding space to the original space, on which RVD and triangulation / tetrahedralization will exhibit the desired anisotropic property.

## 4.1 High-D Particle Optimization with Efficient K-NN

In this section, we discuss the isotropic particle optimization on the explicitly-computed high-d embedded surface or volume, which is the main difference between ours and Zhong et al.'s method [2013]. The limitations of their methods are: (1) they made use of the concept of embedding space for their energy and force formulations, without computing such high-d embedding explicitly, which may lead to more approximation error; (2) during the particle optimization, they had to have a very large range for searching neighboring particles in the original metric space when the anisotropic stretching ratio is high, which is less efficient. Besides that, they only worked on surface case without considering volume case.

However, once the high-d embedding is computed as discussed in Sec. 3, we can directly define the inter-particle energy and force in such embedding space. The goal is to let particles reach isotropic uniform distribution at their equilibrium state, where the particle distribution on the original surface or volume will exhibit the desired anisotropy.

In our method, we formulate the isotropic inter-particle energy and force with explicit coordinates of the particles in the high-d embedding space, which is based on [Zhong et al. 2013]. The energy $\overline{E}^{pq}$ between particles $p$ and $q$ in the high-d embedding space is defined as: $\overline{E}^{pq} = e^{-\frac{\|\bar{\mathbf{x}}_p - \bar{\mathbf{x}}_q\|^2}{4\sigma^2}}$, where $\sigma$ is kernel width, and $\bar{\mathbf{x}}_p$ and $\bar{\mathbf{x}}_q$ are particle positions in the embedding space. Note that our goal is to let the particles uniformly and isotropically sampled on $\overline{\Omega}$, so $\sigma = 0.3\sqrt[d]{|\overline{\Omega}|/n}$, as suggested in [Zhong et al. 2013], where $d = 2$ in surface and $d = 3$ in volume, $|\overline{\Omega}|$ denotes the area or the volume of $\overline{\Omega}$ in the embedding space, $n$ is the number of particles. The force applied from particle $p$ to particle $q$ in the embedding space is defined as the gradient of the inter-particle energy. To sum up all inter-particle energies, the computational complexity of particle optimization is $O(cn_p)$, where $c$ is a constant, i.e., the average number of K-NN for each particle, and $n_p$ is total number of particles (i.e., the number of RVD cells / final mesh vertices). The k-d tree data structure is used to compute K-NN. Since $n_p$ is provided by user, $c$ is the key factor to affect the efficiency of the computation. With the computed high-d Euclidean embedded surface or volume, the particle positions are defined and optimized on such embedding, which helps us to determine the $c$ around 20∼30, i.e., nearest neighbors in Euclidean metric space within five standard deviations of the Gaussian energy kernel ($5\sigma$), where $\sigma$ is the kernel width. If without the high-d Euclidean embedding, hundreds or thousands of neighbors need to be checked and then the spurious neighbors are pruned when the stretching ratio in the given metric

field is high, such as $\geq 10$. The K-NN comparison experiments are given in Appendix D. Finally, we use the L-BFGS algorithm [Liu and Nocedal 1989] to optimize the particle positions constrained on the high-d embedded surface or inside the high-d embedded volume.

*Boundary and Sharp Feature Handling.* Particles on a surface with boundaries should be restricted to remain on it during the particle optimization. During L-BFGS optimization, the updated particle positions need to be projected to their nearest locations on the embedded surface, if they are out of the boundary or out of the surface. The particles in a volume with boundaries can be treated similarly. Besides that, the particle optimization formulation is constructed by using a pushing energy and force based on a Gaussian kernel, which can maximally and optimally push particles away within the boundary constraint so as to cover the entire domain (surface and volume), leading to automatically capture and reconstruct the boundaries without requiring the user's tagging as an extra input (such as CVT and LCT approaches need user's intervention). However, as for the sharp features (in a smooth metric field), users need to identify the feature edges and corners in the input reference mesh at first, and then constrain the movement of the particles along the feature edges or fix the feature corners during optimization. We will provide experiments on these cases in Sec. 6.

### 4.2 Restricted Voronoi Diagram and Mesh Generation

*High-D RVD and Meshing.* The input of high-d RVD computation are the triangle / tetrahedral mesh of the high-d embedded surface / volume $\overline{\Omega}$ and the sites (i.e., optimized particle positions in high-d space) of high-d Voronoi cells. The key task is to identify the high-d Voronoi cells that overlap each triangle / tetrahedron of the embedded surface / volume $\overline{\Omega}$ and compute their intersections. The RVD computation on surface is based on Yan et al.'s method [2009] and the volume version is based on Lévy and Bonneel's method [2012] with the exact geometric predicates from [Lévy 2016], and then extended in high-d space. All these computations are done under the Euclidean metric, which is easy and efficient, even in the high-d space. In particular, note that the geometric predicates in [Lévy 2016] compute barycentric coordinates, that depend on the intrinsic dimension (2 for surfaces, 3 for volumes) rather than ambient dimension (high-d). Without using the high-d Euclidean embedding, one would need to compute anisotropic RVD and its dual anisotropic mesh on the original surface with the given metric, requiring a high-resolution tessellation of the input surface / volume and additional costly computations, such as computation of anisotropic Voronoi cell boundaries, search for neighboring Voronoi sites in anisotropic metric space, etc. Once the RVD is obtained, we can easily compute its dual graph, i.e., Restricted Delaunay Triangulation (RDT).

*Anisotropic RVD and Mesh.* Using the barycentric coordinates of each output site or vertex, we can back-project the RVD and RDT from the high-d embedding space onto the original space, and generate the final anisotropic RVD and mesh. As we mentioned before, if the given Riemannian metrics are smooth, the proposed high-d embedding framework can compute a smooth and small-error embedding in higher dimensions. The algorithm robustly computes the RVD using filtered geometric predicates and symbolic perturbation to resolve degeneracies [Lévy 2016]. Note that in general, there is no guarantee that the dual mesh (i.e., the associated RDT) will not have inverted elements (negative Jacobian) when back-projected to 3D. Whenever such an inverted element is detected, our implementation inserts additional points using the provably terminating algorithm in [Rouxel-Labbé et al. 2016], that uses the same discretization of the metric as us (also used in [Lévy and Bonneel 2012]). In practice, we did not observe such inverted elements in our empirical experiments (the refinement step was never triggered).

## 5 EVALUATIONS

### 5.1 Embedding Quality

In order to evaluate the optimized high-d embedding of the surface or volume equipped with a given metric (in practice, the given metric is specified in the input mesh at each vertex), we consider the edge lengths of the input triangular surface or tetrahedral volume under the given metric, and use them as the ground truth to evaluate computed embedding results.

For each edge with vertices $\mathbf{v}_a$ and $\mathbf{v}_b$, we use the average metric of two vertices $\mathbf{Q}_{ab} = (\mathbf{Q}(\mathbf{v}_a) + \mathbf{Q}(\mathbf{v}_b))/2$ to approximate the metric for this edge. Then the length of each edge can be computed according to its own $\mathbf{Q}_{ab}$ and we use it as the ground truth. The *relative edge length error* is the percentage of the absolute difference between the ground truth and the edge length of computed embedding with respect to the ground truth. In our experiments, we only use relative edge length error to measure the computed embedding, because it is invariant to scales. The criteria for evaluating the embedding quality are: $L^{rel}_{max}$ and $L^{rel}_{avg}$, i.e., the maximal and average values of relative edge length errors of all embedded triangles / tetrahedrons. The color-coded diagram of the relative edge length errors is rendered.

### 5.2 Anisotropic Mesh Quality

Since RVD is not straightforward to be evaluated as compared with its dual triangular and tetrahedral meshes, in the experiments, we will focus on the mesh quality for evaluations.

*Surface:* In the anisotropic triangular meshing, for each triangle $\triangle_{abc}$ in the final mesh, we use its approximated metric $\mathbf{Q}(\triangle_{abc}) = (\mathbf{Q}(\mathbf{x}_a) + \mathbf{Q}(\mathbf{x}_b) + \mathbf{Q}(\mathbf{x}_c))/3$ to affine-transform it from the original anisotropic space into the Euclidean space. After that, we employ the following isotropic triangular criteria, as suggested by Frey and Borouchaki [1997] and used in Zhong et al. [2013] and Fu et al. [2014]'s recent work, to evaluate the quality of generated anisotropic triangular mesh: The quality of a triangle is measured by $G = 2\sqrt{3}\frac{S}{ph}$, where $S$ is the triangle area, $p$ is its half-perimeter, and $h$ is the length of its longest edge. $G_{min}, G_{avg}$ are the minimal and average qualities of all triangles. $\theta_{min}, \theta_{avg}$ are the smallest and average angles of the minimal angles of all triangles. $\%_{<30^\circ}$ is the percentage of triangles with their minimal angles smaller than $30^\circ$. The angle histogram is also provided. $\theta_{avg}$ should be $60^\circ$ if it is a regular triangle. $G$ is between 0 and 1, where 0 denotes a skinny triangle and 1 denotes a regular triangle.

*Volume:* In the anisotropic tetrahedral mesh, for each tetrahedron $tet_{abcd}$, its approximated metric is $\mathbf{Q}(tet_{abcd}) = (\mathbf{Q}(\mathbf{x}_a) + \mathbf{Q}(\mathbf{x}_b) + \mathbf{Q}(\mathbf{x}_c) + \mathbf{Q}(\mathbf{x}_d))/4$. Then we use the corresponding $\mathbf{Q}(tet_{abcd})$ to affine-transform the tetrahedron in the Euclidean space, and then

employ the following isotropic tetrahedral mesh quality measurements [Dardenne et al. 2009; Yan et al. 2013] to evaluate the quality of generated anisotropic tetrahedral mesh: The quality of a tetrahedron is measured by $G = \frac{12\sqrt[3]{9V^2}}{\sum l_{i,j}^2}$, where $V$ is the volume of the tetrahedron, and $l_{i,j}$ is the length of the edge which connects vertices $v_i$ and $v_j$. $G_{min}, G_{avg}$ are the minimal and average qualities of all tetrahedrons. $\theta_{min}, \theta_{avg}$ are the smallest and average angles of the minimal dihedral angles of all tetrahedrons. $\%_{<15^\circ}$ is the percentage of tetrahedrons with their dihedral angles smaller than $15^\circ$, which are considered as slivers. The angle histogram is also provided to show the distribution of minimal dihedral angles for all tetrahedrons. $\theta_{avg}$ should be $\approx 70.53^\circ$ if it is a regular tetrahedron. $G$ is between 0 and 1, where 0 denotes a sliver and 1 denotes a regular tetrahedron.

## 6 RESULTS

We implement our algorithms by using Microsoft Visual C++ 2013. The embedding and mesh quality evaluations are implemented with Matlab R2015a. For the hardware platform, the experiments are run on a desktop computer with Intel(R) Core(TM) i7-6700 CPU with 3.40GHz, 32GB DDR4 RAM.

*Riemannian Metric Design.* In our experiments, users firstly design a smooth scaling field $\mathbf{S}(\mathbf{x})$ and a rotation field $\mathbf{R}(\mathbf{x})$ that is smooth in regions other than some singularities, and then compose them to $\mathbf{M}(\mathbf{x}) = \mathbf{R}(\mathbf{x})^T\mathbf{S}(\mathbf{x})^2\mathbf{R}(\mathbf{x})$, which is similar to the approach mentioned in [Du and Wang 2005a; Yamakawa and Shimada 2000]. Since this article focuses on both 2- and 3-manifolds, the metric design has some slight differences as follows.

For the anisotropic meshing in 2D domains, we use the following $2 \times 2$ metric tensor:

$$\mathbf{M} = [\mathbf{v}_1, \mathbf{v}_2] diag({s_1}^2, {s_2}^2)[\mathbf{v}_1, \mathbf{v}_2]^T, \tag{18}$$

where $\mathbf{v}_1$ and $\mathbf{v}_2$ are two orthogonal principal directions, composing a rotation field. $s_1$ and $s_2$ are two stretching factors along principal directions.

For the anisotropic meshing on 3D surfaces, we use the following $3 \times 3$ metric tensor:

$$\mathbf{M} = [\mathbf{v}_{min}, \mathbf{v}_{max}, \mathbf{n}] diag(1, (\frac{s_2}{s_1})^2, 0)[\mathbf{v}_{min}, \mathbf{v}_{max}, \mathbf{n}]^T, \tag{19}$$

where $\mathbf{v}_{min}$ and $\mathbf{v}_{max}$ are the directions of the principal curvatures, $\mathbf{n}$ is the unit surface normal. $s_1$ and $s_2$ are two user-specified stretching factors along principal curvature directions. Since the surface models are computed by curvature-based metric tensor fields, we use the metric of Eq. (19) with $s_1 = \sqrt{K_{min}}$ and $s_2 = \sqrt{K_{max}}$, where $K_{min}$ and $K_{max}$ are the principal curvatures. We set small thresholds to preserve both $K_{min}$ and $K_{max}$ not vanishing. As suggested by Alliez et al. [2003], Laplacian smoothing is applied to both the stretching factors and directions, to ensure smoothness of the input metric field on the surface. Then, $\frac{s_2}{s_1}$ is defined as stretching ratio ($\geq 1$) in the surface metric field $\mathbf{M}$.

For the anisotropic meshing in 3D volumes, we use the following $3 \times 3$ metric tensor:

$$\mathbf{M} = [\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3] diag({s_1}^2, {s_2}^2, {s_3}^2)[\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3]^T, \tag{20}$$

where $\mathbf{v}_1$, $\mathbf{v}_2$, and $\mathbf{v}_3$ are three orthogonal principal directions. $s_1$, $s_2$, and $s_3$ are three stretching factors along principal directions.

It is noted that since our embedding computation is based on the strategy of keeping the original 2D / 3D coordinates, and embedding additionally higher dimensions, the Euclidean distance in the high-d space will be at least longer than or equal to what was given in the original mesh. In order to overcome this obstruction, in the smaller metric cases, we need to pre-process the input metrics for existing benchmarks to compute embeddings and generate anisotropic meshes. For instance, if any magnitude of $s_1$ and $s_2$ (in a 2D tensor) or $s_1$, $s_2$, and $s_3$ (in a 3D tensor) is less than one, we will multiply the target Riemannian metric $\mathbf{M}$ by a suitable global constant (scaling), which can guarantee that the smallest metric is larger than or equal to one, as well as preserve the shapes and size ratios of the cells in the RVD, and those of triangles / tetrahedrons in the anisotropic mesh. We will give examples in this case in later Sec. 6.

### 6.1 SIFHDE$^2$ Computation

*6.1.1 Importance of Higher Dimensions.* In order to demonstrate the importance of the high-d embedding, instead of embedding computed in the original space, we use two examples in detail to explain its advantages.

One example is a 2D domain with a Riemannian metric field. Fig. 4 shows a simple 3D self-intersection free embedding computed by our method for a 2D domain equipped with an anisotropic metric, which can "perfectly" embed the original 2D domain into a Gaussian surface in 3D space. Since we have both the original domain with the metric and its ground truth of the embedded surface (the target dimension is known, i.e., 3D), we can explicitly evaluate our high-d embedding optimization framework quantitatively and visually.

The pipeline of the proposed work is given in Fig. 4, i.e., to compute the self-intersection free high-d embedding, isotropic particle optimization on such embedding, high-d isotropic RVD and dual meshing, final anisotropic RVD and mesh. The anisotropic metric field is $\mathbf{M}(\mathbf{x}) = \mathbf{R}(\mathbf{x})^T diag\{Stretch(\mathbf{x})^2, 1\}\mathbf{R}(\mathbf{x})$, where the stretching field $Stretch(\mathbf{x}) \in [1, 3.85]$ with the rotation field $\mathbf{R}(\mathbf{x})$. Our proposed embedding optimization framework converges superbly fast, i.e., 20 iterations in this 3D embedding. The embedding quality is quite satisfactory, i.e., $L_{avg}^{rel} = 0.92\%$ and $L_{max}^{rel} = 10.58\%$. Besides the high quality, there is no self-intersection, due to the strategy we discuss in Sec. 3.4.2, i.e., fixing the original 2D coordinates and deforming the new added 3rd coordinates in the embedding computation. Visually, the shape of the embedding result at 20 iterations (Fig. 4 b) looks very similar to the shape of the ground truth Gaussian surface (Fig. 4 a). By using this 3D embedding, we can compute the uniform distribution of particles (Fig. 4 c) for RVD and dual meshing on it (Fig. 4 d), and finally generate high-quality anisotropic RVD and triangular mesh with 2000 output samples in Fig. 4 (e).

Note that without using an extra higher dimension (i.e., the 3rd coordinate), the output embedding will have self-intersections, since the input metric controls the deformations of each triangle in the original 2D space, which will very easily cause self-intersections, especially when the metric field has large stretching ratios and directions. Regularizing the embedding to avoid self-intersections is not a good strategy, since it would dramatically increase the error of the embedding.

Another example is a 3D Cyclide surface with a Riemannian metric field, which is designed according to the surface curvature. The

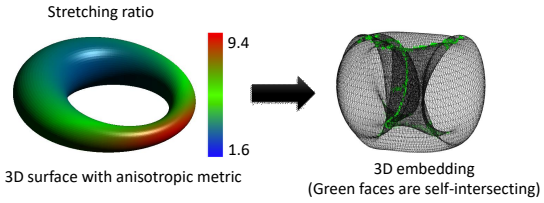


Fig. 5. A 3D embedding result [Panozzo et al. 2014] of a Cyclide surface. There are 1146 self-intersecting faces out of total 21,600 faces as shown in green color.

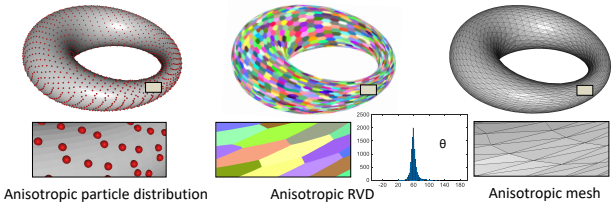


Fig. 6. The anisotropic particle distribution, RVD, and its dual anisotropic mesh of the Cyclide surface by using the 8D embedding.

stretching ratio is $\frac{s_2}{s_1} \in [1.6, 9.4]$. Without a high-d embedding, i.e., using a 3D embedding as suggested by Panozzo et al. [2014], in Fig. 5, though the embedding errors are relatively small, i.e., $L^{rel}_{avg} = 2.36\%$ and $L^{rel}_{max} = 30.62\%$ (still larger than our 8D embedding errors as shown in Tab. 2), there are many self-intersecting faces in the computed 3D embedding (i.e., 1146 self-intersecting faces in this Cyclide model), which are harmful for the particle optimization, RVD and dual mesh computations. The major fundamental difference between their framework and ours is that they deform the embedding directly in the original 3D coordinates, which may lead to self-intersections; however, we deform the embedding in high-d space with fixing the first three coordinates to avoid self-intersections (as discussed in Sec. 3.4.2).

Based on the self-intersection free high-d embedding, we can compute particle distribution, RVD, and its dual mesh; and finally map them back to the original 3D space to generate the anisotropic RVD and its dual anisotropic mesh. The results of the Cyclide model are shown in Fig. 6.

Moreover, we also investigate some other surface models and Tab. 1 shows that self-intersecting faces do widely exist in the 3D embedding (i.e., Panozzo et al.'s method [2014]), so that it needed an extra step to parameterize the 3D embedding for their final quad meshing. Besides that, the parameterization methods have limitations to deal with complicated topologies as pointed out by [Fu et al. 2014; Zhong et al. 2014]. However, our high-d embedding method does not have any constraint on model topologies. In order to further compare with Panozzo et al.'s 3D embedding method [2014], we provide the analysis of the embedding accuracy in Sec. 6.2.

Table 1. Statistics (i.e., numbers and percentages) of self-intersecting faces for embeddings in 3D and high-d spaces on different surfaces.

| Model | Cyclide1 | Cyclide2 | Kitten | Gargo | Upright | Nefertiti |
|---|---|---|---|---|---|---|
| 3D | 1146<br>5.31% | 1751<br>3.38% | 1001<br>2.50% | 2405<br>2.40% | 3584<br>2.38% | 1385<br>5.61% |
| High-D | 0 | 0 | 0 | 0 | 0 | 0 |

*6.1.2 Choosing the Dimension of the Embedding.* We elaborate on how to choose the dimension of the high-d target for computing the SIFHDE$^2$ with small relative edge length errors. In general, we do not know beforehand the required dimension for the high-d embedding. From Nash theorem, it is known that for constructing a $C^1$ isometric embedding of a $n$D metric, $2n$ dimensions suffice (Nash-Kuiper). For higher smoothness ($C^k, k \geq 2$), the bound on embedding dimension becomes $n(3n + 11)/2$ (Nash-Moser). With our regularization term, we constrain the embedding to be $C^2$ and thus avoid the oscillations of $C^1$ embeddings, but the Nash-Moser theoretical bound becomes 17D for surfaces and 30D for volumes. We now evaluate the required embedding dimension in practice for surfaces and volumes and show empirically that in both cases 8D suffices to accurately account for the input metric:

*Surface Case:* We conduct different experiments with surfaces in varying dimensions. The associated metric deviation errors of a Cyclide surface are plotted in Fig. 7. It is noted that in our case, when using embedding dimensions larger than 8D, the relative edge length errors are quite stable and small (Cyclide surface: $L^{rel}_{avg} = 1.83\%$ and $L^{rel}_{max} = 25.81\%$).

We also investigate some other surface models to conclude that 8D space is a good target dimension for computing SIFHDE$^2$ with smaller relative edge length errors, e.g., Kitten surface: $L^{rel}_{avg} = 3.82\%$ and $L^{rel}_{max} = 139.21\%$. More experiments with varying dimensions for different surface models are given in Appendix C.

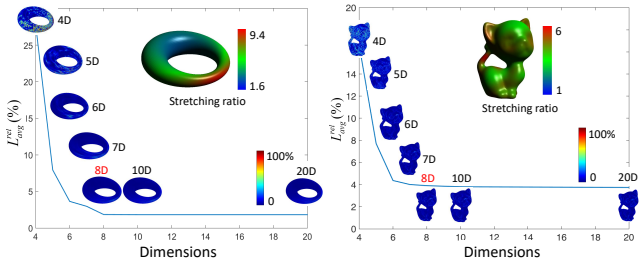


Fig. 7. The relative edge length errors of high-d embeddings (i.e., $4D \sim 20D$) of the Cyclide and Kitten surfaces.

*Volume Case:* As for the volume models, we also investigate the ideal target dimension for computing the self-intersection free embedding with smaller relative edge length errors. Similar to surface case, volume models (as shown in Fig. 8) also conclude that 8D is a good target dimension for computing SIFHDE$^2$ with smaller relative edge length errors, e.g., Cube volume: $L^{rel}_{avg} = 1.65\%$ and $L^{rel}_{max} = 17.54\%$, Sphere volume: $L^{rel}_{avg} = 3.21\%$ and $L^{rel}_{max} = 65.01\%$. As demonstrated in the surface case, we also observe that beyond 8D, the relative edge length errors remain stable and small. More experiments with varying dimensions for different volume models are given in Appendix C.

## 6.2 Comparison with 3D Embedding

Tab. 2 shows the comparison of embedding accuracy between Panozzo et al.'s 3D embedding method [2014] and our proposed high-d embedding method on different surface models. It demonstrates that the 8D embeddings have smaller errors, mainly because of more degrees of freedom, besides the advantages of avoiding self-intersection as shown in Tab. 1.

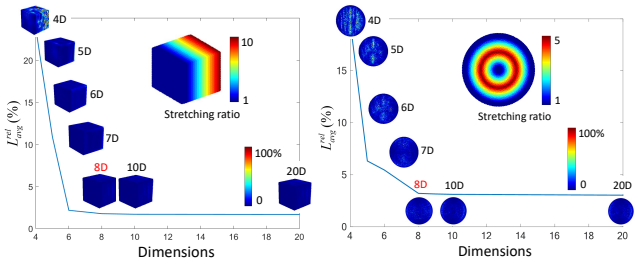


Fig. 8. The relative edge length errors of high-d embeddings (i.e., $4D \sim 20D$) of the Cube and Sphere (shown its cross section) volumes.

Table 2. Statistics of average relative edge length errors in 3D embedding [Panozzo et al. 2014] and the proposed 8D embedding on different surface models.

| Model | Cyclide1 | Cyclide2 | Kitten | Gargo | Upright | Nefertiti |
|---|---|---|---|---|---|---|
| $3D$ | 2.36% | 2.41% | 3.86% | 8.35% | 1.85% | 3.19% |
| $8D$ | 1.83% | 2.18% | 3.82% | 6.72% | 1.52% | 2.98% |

## 6.3 3D Surface RVD and Meshing

*Our Results.* Fig. 1 (2D domain) shows the anisotropic RVD and meshing results of a 2D square domain with a highly-varying metric field, which is defined by the Hessian of the non-convex analytic functions. Actually, a 2D domain can be considered as a 3D flat surface. Fig. 1 (3D surface), Fig. 6, and Fig. 9 show the anisotropic 3D surface RVD and meshing results of Kitten, Cyclide and Gargo models with metrics designed by surfaces' curvature tensors.

*Sharp Feature and Boundary Models.* Fig. 10 is a surface meshing result on Upright CAD model with smooth metrics but sharp geometric features. The computed vertices and triangle edges can well preserve the sharp features (e.g., edges) on the model. However, if the input metric has sudden discontinuities, that is beyond the scope of this paper. Fig. 11 shows surface RVD and meshing results on Nefertiti model with boundaries.

The statistics and timings for our 8D embedding computation and surface meshing are shown in Tab. 3. It is noted that our embedding computation in 8D space is quite efficient, and most surface models provided in this article need only dozens of seconds. There are some more surface RVD and meshing results with different sizes given in Appendix E. Totally, we have tested 16 surface models with corresponding curvature tensors by using the proposed 8D embedding for RVD and meshing computation.

*Comparison with Other Embedding Meshings.* Fig. 12 shows the comparison with 2D conformal embedding [Zhong et al. 2014] and Lévy and Bonneel's 6D embedding [2012], and our method, on a Kitten surface model with 5000 output samples. Let us first compare the 2D conformal embedding method with our embedding method, since both of them are considering the curvature tensors as input. The 2D conformal embedding method produces poor mesh quality results as shown in Tab. 4, since it computes the embedded surface into 2D space, which has very limited degrees of freedom. Our high-d embedding method, i.e., 8D, obtains better mesh quality result since it better matches the input curvature-based metric field. The 6D embedding method has some more degrees of freedom compared with 2D embedding, but it applies a 6D metric in terms of vertex positions

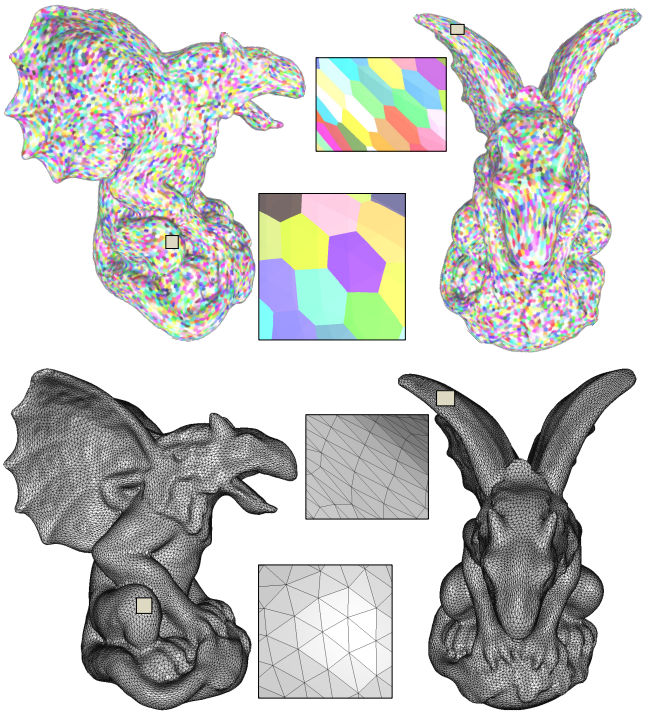

Fig. 9. The anisotropic 3D surface RVD and mesh with 25,000 vertices on Gargo model (stretching ratio $\frac{s_2}{s_1} \in [1, 7]$) generated by our 8D embedding method. The result has both isotropic and anisotropic Voronoi cells and triangles according to the input surface curvature tensor field.

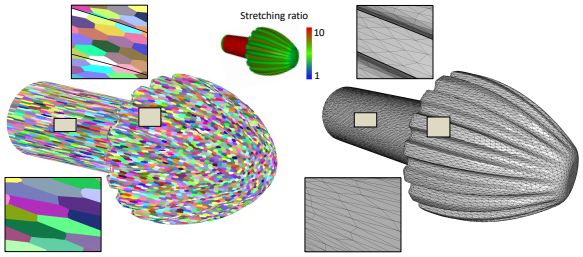


Fig. 10. The anisotropic 3D surface RVD and mesh on Upright model (stretching ratio $\frac{s_2}{s_1} \in [1, 10]$) with sharp geometric features generated by our 8D embedding method.

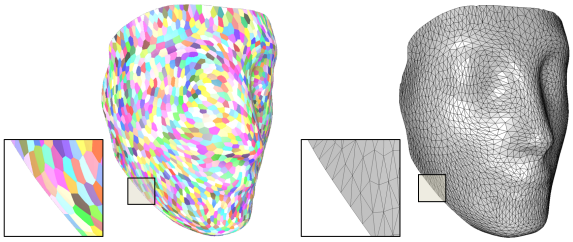

Fig. 11. The anisotropic 3D surface RVD and mesh on Nefertiti model (stretching ratio $\frac{s_2}{s_1} \in [1, 6]$) with boundaries generated by our 8D embedding method.

Table 3. Statistics and timings for embedding computation and surface meshing.

| Model | Input #Vert. | Stretch | $L^{rel}_{avg}$ | $L^{rel}_{max}$ | Embed. Time | Output #Vert. | $G_{min}$ | $G_{avg}$ | $\theta_{min}$ | $\theta_{avg}$ | $\%_{<30°}$ | Mesh Time |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 2D Domain (Fig. 1) | 63, 001 | [1, 40] | 4.04% | 76.79% | 64.69 s | 4000 | 0.34 | 0.89 | 18.11° | 51.51° | 0.19% | 27.15 s |
| Cyclide1 (Fig. 6) | 10, 800 | [1.6, 9.4] | 1.83% | 25.81% | 9.14 s | 2000 | 0.57 | 0.92 | 28.88° | 54.47° | 0.05% | 6.22 s |
| Cyclide2 (Fig. 13) | 25, 920 | [2, 29] | 2.18% | 19.39% | 23.63 s | 8000 | 0.63 | 0.92 | 29.07° | 53.70° | 0.009% | 42.83 s |
| Kitten (Fig. 1) | 20, 000 | [1, 6] | 3.82% | 139.21% | 17.56 s | 2000 | 0.33 | 0.85 | 20.69° | 48.60° | 0.34% | 7.87 s |
| Nefertiti | 12, 478 | [1, 6] | 2.98% | 46.33% | 9.20 s | 2000 | 0.32 | 0.86 | 20.14° | 48.09° | 0.49% | 7.05 s |
| Upright | 75, 196 | [1, 10] | 1.52% | 28.35% | 115.57 s | 10, 000 | 0.46 | 0.88 | 25.44° | 51.12° | 0.04% | 62.16 s |
| Gargo | 50, 032 | [1, 7] | 6.72% | 151.55% | 50.76 s | 25, 000 | 0.32 | 0.87 | 20.02° | 49.94° | 0.45% | 91.19 s |

Note: Embed. Time: timing for embedding computation with 50 iterations. Mesh Time: timing for both particle optimization with 50 iterations and RVD computation.

and normals, without considering the curvature tensors, which does not adapt to the geometry defined by an input Riemannian metric field. In order to compare these three embedding meshing methods, we use Hausdorff distance to measure the faithfulness accuracy of these three results to the original input surface. The Hausdorff distance of our result is 0.004964% of the bounding-box's diagonal length, while Hausdorff distances of 2D conformal embedding and 6D embedding results are 0.013415% and 0.005110%, respectively. It is noted that the meshing fidelity of 2D embedding is worse than both those of 6D and our 8D embeddings, though it has a reasonable mesh quality to match the designed metric. Essentially, when using the dimension reduction, such as mapping a 3D shape into a 2D domain, it may lose some 3D shape fidelity properties.

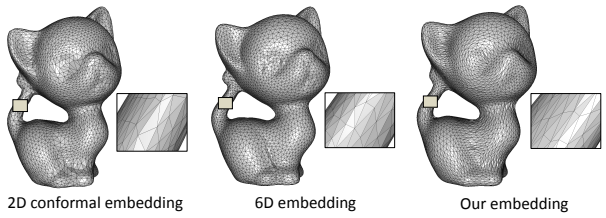


Fig. 12. Comparison on anisotropic meshing results with 2D conformal embedding method [Zhong et al. 2014], Lévy and Bonneel's 6D embedding [2012], and our method on the Kitten surface model.

Table 4. Comparison of mesh quality on different embedding meshings.

| Method | $G_{min}$ | $G_{avg}$ | $\theta_{min}$ | $\theta_{avg}$ | $\%_{<30°}$ |
|---|---|---|---|---|---|
| 2D embedding | 0.08 | 0.75 | 5.02° | 41.91° | 8.24% |
| Our embedding | 0.32 | 0.91 | 19.04° | 50.13° | 0.86% |

*Comparison with Other Anisotropic Surface Meshings.* In this subsection, we show further comparative analysis and experiments with other state-of-the-art anisotropic meshing approaches. Fig. 13 compares our method with both particle-based method [Zhong et al. 2013] and LCT method [Fu et al. 2014] on a Cyclide model with the large stretching ratio $\frac{s_2}{s_1} \in [2, 29]$. Our method improves the mesh quality significantly as shown in Fig. 13 compared with the original particle-based method [Zhong et al. 2013], since we have computed the particle optimization directly on the high-d embedding. Compared with LCT method (one of the optimal high-quality anisotropic meshing methods), our method increases the minimal angle from 26.94° to 29.07° and $\%_{<30°} = 0.03\%$ versus $\%_{<30°} = 0.009\%$, meanwhile we have similar quality of average values. Fig. 14 visualizes the final mesh quality errors (i.e., 1-$G$, where $G$ is the quality of triangle in Sec. 5.2) of the above three methods. We can see that our errors are more smoothly distributed, whereas particle-based method and LCT method have more poor quality triangles.

There are some more comparison results given in Appendix F due to the page limit.

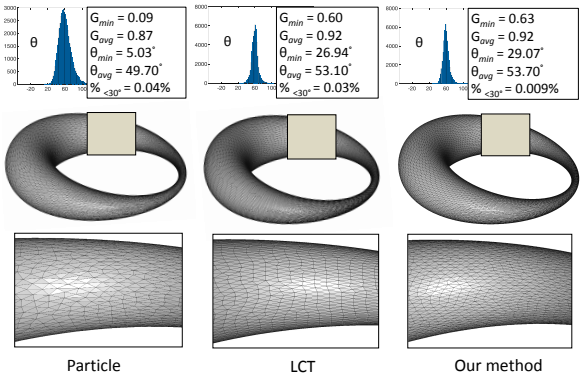


Fig. 13. Comparison on anisotropic surface meshing results (8000 output samples) with particle-based method [Zhong et al. 2013], LCT method [Fu et al. 2014], and our method on the Cyclide surface model with the stretching ratio $\frac{s_2}{s_1} \in [2, 29]$.

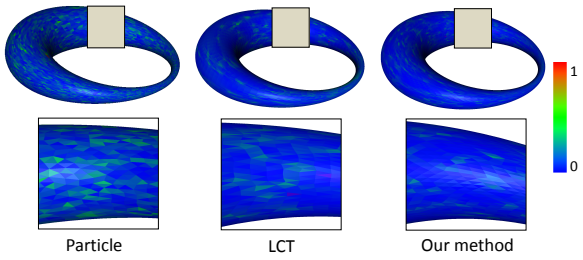


Fig. 14. Comparison on final mesh quality errors (i.e., 1-$G$) with particle-based method [Zhong et al. 2013], LCT method [Fu et al. 2014], and our method on the Cyclide surface model with the stretching ratio $\frac{s_2}{s_1} \in [2, 29]$.

## 6.4 3D Volume RVD and Meshing

*Volumetric RVD and Meshing.* Firstly, the anisotropic metric fields in 3D volume models are designed by different analytic functions in our experiments. The statistics and timings for our 8D embedding computation and volume meshing are shown in Tab. 5. Fig. 15 shows the anisotropic 3D volume RVD and meshing results of a Cube model with domain $[0, 1]^3$ of two different Riemannian metric fields $\mathbf{M}(\mathbf{x}) = diag\{Stretch(\mathbf{x})^2, 1, 1\}$. The first one is designed by a linear function with stretching ratios $Stretch(\mathbf{x}) = 10x$, where $Stretch(\mathbf{x}) \in [1, 10]$, and the second one is designed by a highly nonlinear function with large stretching ratios $Stretch(\mathbf{x}) = 0.2(0.0025 + 0.2(1 - e^{-|x-0.5|}))^{-0.8}, \in [1, 25]$.

A unit Sphere model with a cylindrical anisotropic metric field is in Fig. 1. The Riemannian metric is designed as $\mathbf{M}(\mathbf{x}) = \mathbf{R}(\mathbf{x})^T diag$

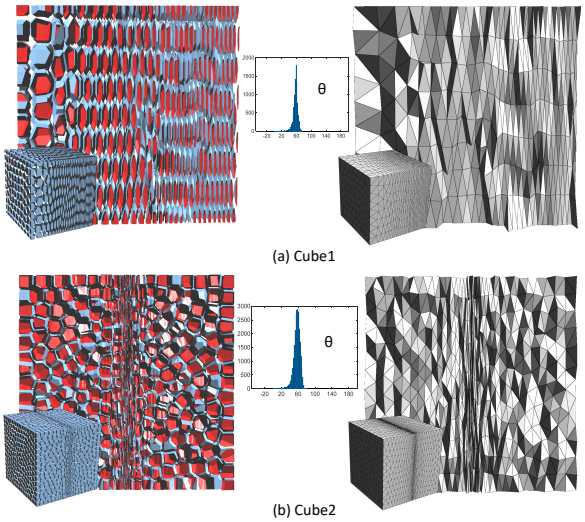


Fig. 15. The anisotropic 3D volume RVD and meshes on two Cube models (with different Riemannian metric fields) generated by our 8D embedding method. Both the surface and interior of the volume RVD and tetrahedral meshes are shown. In order to better visualize the 3D RVD results, we shrink each Voronoi cell (i.e., polyhedron) a little bit. The blue color represents the outside of each 3D Voronoi cell, and the red color represents the inside of each 3D Voronoi cell.

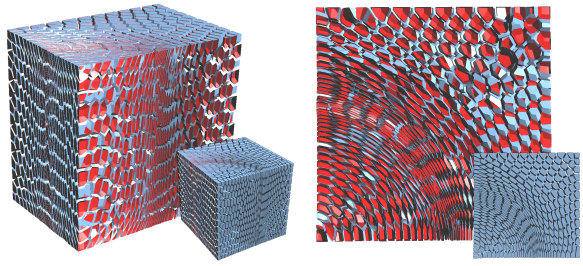

Fig. 16. The anisotropic 3D volume RVD on the Cube model with the large cylindrical anisotropic metric field generated by our 8D embedding method. Both the surface and interior of the volume RVD are shown.

$\{Stretch(\mathbf{x})^2, 1, 1\}\mathbf{R}(\mathbf{x})$, where $Stretch(\mathbf{x}) \in [1, 5]$, and $\mathbf{R}(\mathbf{x})$'s columns are $(x/\sqrt{x^2+y^2}, y/\sqrt{x^2+y^2}, 0)^T$, $(-y/\sqrt{x^2+y^2}, x/\sqrt{x^2+y^2}, 0)^T$, and $(0, 0, 1)^T$.

Fig. 16 shows one volumetric RVD result on a Cube model with domain $[1, 11]^3$. The targeted cylindrical Riemannian metric field is specified via a highly nonlinear analytic function as $\mathbf{M}(\mathbf{x}) = \mathbf{R}(\mathbf{x})^T diag\{Stretch(\mathbf{x})^2, 1, 1\}\mathbf{R}(\mathbf{x})$, where $Stretch(\mathbf{x}) = 2(0.1 + 2(1 - e^{-0.01|x^2+y^2-49|}))^{-1}$, and $\mathbf{R}(\mathbf{x})$ is a cylindrical rotation field as defined previously. The stretching ratio is $Stretch(\mathbf{x}) \in [1, 20]$. By using our method, we can generate highly curved 3D Voronoi cells in regions where the metric varies quickly, while in low stretching regions, the regular 3D Voronoi cells are obtained. Its dual tetrahedral mesh is given in the following.

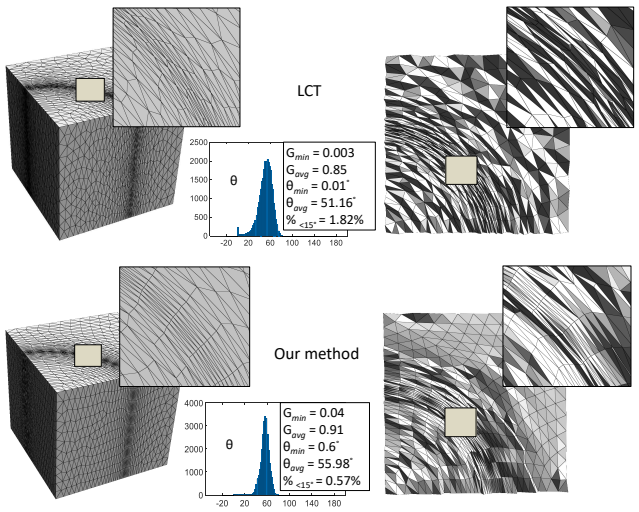


Fig. 17. Comparison on anisotropic volume meshing results with LCT method [Fu et al. 2014] and our method on the Cube volume model with the cylindrical Riemannian metric field of stretching ratio $Stretch(\mathbf{x}) \in [1, 20]$ (without sliver elimination process).

Fig. 17 shows the meshing comparison with our method and LCT method [Fu et al. 2014] on a Cube model with the previous cylindrical Riemannian metric field. By using almost the same number of vertices, i.e., 5637 vertices in LCT method and 5600 vertices in our method, we obtain better mesh quality results in average and minimal value measurements as shown in Fig. 17. Visually, our result achieves better mesh stretching ratios and directions in both surface and interior of the volume. The reason why we have poor results in minimal mesh quality and smallest angle is that we do not apply any sliver elimination strategy. Since the sliver elimination is a post-processing in both methods, it is more straightforward to compare the performance of two algorithms without doing sliver elimination. As demonstrated, LCT method has 517 slivers (i.e., tetrahedrons with the dihedral angle $< 15°$), and we have a smaller number of slivers, i.e., 164. It is noted that the LCT result in Fig. 17 is before applying sliver elimination, which was provided by the authors of [Fu et al. 2014].

Fig. 18 shows one volumetric RVD result on a unit Sphere model with a smaller number of samples, i.e., 1000. The Riemannian metric is defined as a cosine wave anisotropy: $\mathbf{M}(\mathbf{x}) = \mathbf{R}(\mathbf{x})^T diag\{Stretch(\mathbf{x})^2, 1, 1\}\mathbf{R}(\mathbf{x})$, where $Stretch(\mathbf{x}) = 10$, and $\mathbf{R}(\mathbf{x})$'s columns are $(2cos(4x), 1, 0)^T$, $(1, -2cos(4x), 0)^T$, and $(0, 0, 1)^T$. A closeup shows clearly how the 3D Voronoi cells are extremely curved. Both the surface and interior of the volumetric RVD are illustrated. This example is to demonstrate the advantage of using our proposed SIFHDE$^2$ as a tool to compute arbitrary anisotropic Voronoi diagrams. Without using SIFHDE$^2$, such as Budninskiy et al. [2016]'s work by using a convex function to represent the anisotropy, it is very difficult to obtain our result.

*Volumetric Tensor Field Visualization.* Lastly, we apply our 3D volumetric RVD method on some real engineering and CFD tensor field datasets. The eigenvalues in Eq. (20) (i.e., $s_1$, $s_2$, and $s_3$), after computing SVD of input Riemannian metric $\mathbf{M}$, can be arbitrary in these

Table 5. Statistics and timings for embedding computation and volume meshing.

| Model | Input #Vert. | Stretch | $L^{rel}_{avg}$ | $L^{rel}_{max}$ | Embed. Time | Output #Vert. | $G_{min}$ | $G_{avg}$ | $\theta_{min}$ | $\theta_{avg}$ | $\%_{<15^\circ}$ | Mesh Time |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Cube1 (Fig. 15 a) | 9261 | [1, 10] | 1.65% | 17.54% | 138.84 s | 2000 | 0.09 | 0.92 | $1.66^\circ$ | $57.24^\circ$ | 0.29% | 29.71 s |
| Cube2 (Fig. 15 b) | 44, 541 | [1, 25] | 1.30% | 48.39% | 567.56 s | 5000 | 0.10 | 0.91 | $1.68^\circ$ | $56.76^\circ$ | 0.42% | 83.73 s |
| Cube3 (Fig. 17) | 44, 541 | [1, 20] | 2.12% | 71.32% | 562.28 s | 5600 | 0.04 | 0.91 | $0.6^\circ$ | $55.98^\circ$ | 0.57% | 91.32 s |
| Sphere (Fig. 1) | 50, 000 | [1, 5] | 3.21% | 65.01% | 587.61 s | 5000 | 0.08 | 0.88 | $1.34^\circ$ | $54.02^\circ$ | 1.24% | 77.67 s |

Note: Embed. Time: timing for embedding computation with 50 iterations. Mesh Time: timing for both particle optimization with 50 iterations and RVD computation. The small values are in $G_{min}$ and $\theta_{min}$, since we do not apply any sliver elimination strategy (a post-processing) in our meshing results.

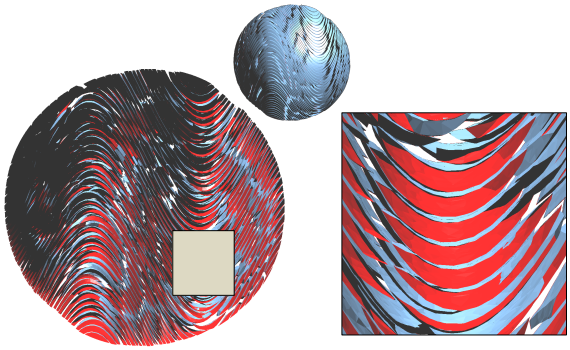

Fig. 18. The 3D anisotropic volumetric RVD on the Sphere model (with a cosine wave anisotropy) generated by our 8D embedding method. Both the surface and interior of the volume RVD are shown.

real datasets, e.g., sometimes the absolute values are smaller than one. In this case, as mentioned before, a global constant should be multiplied to $\mathbf{M}$ (we need to multiply a constant value, 5, in both models, since the minimal eigenvalues are 0.2). Fig. 19 shows the visualization of the 3D stress tensor field resulted from the simulation of a brake lever (data downloaded from www.tensorvis.org) and Fig. 20 shows the 3D tensor field visualization of a CFD data. To our knowledge, this is the first time to use 3D anisotropic Voronoi diagram to visualize the 3D tensor field. The main advantage of using 3D volumetric RVD to visualize the tensor field is that Voronoi cells can be freely distorted with respect to the local tensor, which can achieve a continuous visualization of the given tensor field.

There are some more volumetric RVD and meshing results with different sizes given in Appendix E. Through the above experiments, it is noted that our proposed SIFHDE$^2$ and high-d RVD are effective computational algorithms to generate the high-quality anisotropic 3D Voronoi diagrams and tetrahedral meshes with arbitrary smooth Riemannian metric fields.

## 6.5 Special Riemannian Metric Cases

In this subsection, we further investigate the convergence speed and accuracy of the proposed embedding computation on some special 2D and 3D Riemannian metric fields with small metric tensors (especially some stretching factors / eigenvalues are less than one) buried in large metric tensors all around.

*A 2D Metric.* Fig. 21 shows an input 2D metric domain having concentric rings with smaller metrics (in small regions shown in the blue color) and larger metrics (in large regions). Since the input stretching factors ($\in [0.3, 3]$) in the major direction have values less than one, we need to multiply the input metric by a global constant value, such as 3.34. Our embedding framework can efficiently converge within 50 iterations with smaller relative edge length errors, e.g., $L^{rel}_{avg} = 1.55\%$ and $L^{rel}_{max} = 28.39\%$. Based on

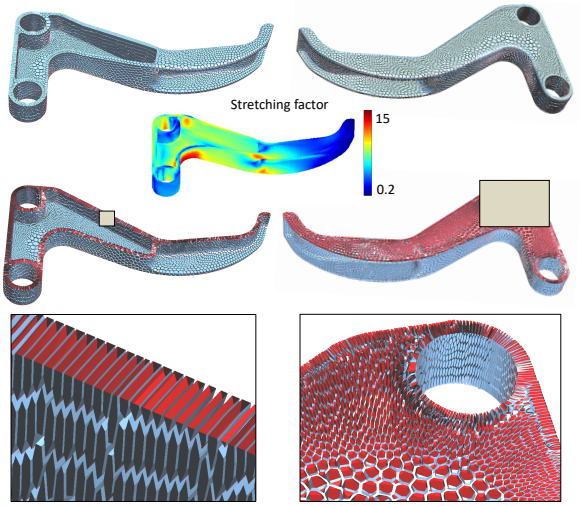


Fig. 19. The visualization of the stress tensor field by using 3D anisotropic volumetric RVD (with 10,000 samples) on the Brake Lever model (filled with air outer part) with the stretching factor in the major direction $s_1(\mathbf{x}) \in [0.2, 15]$ generated by our 8D embedding method.

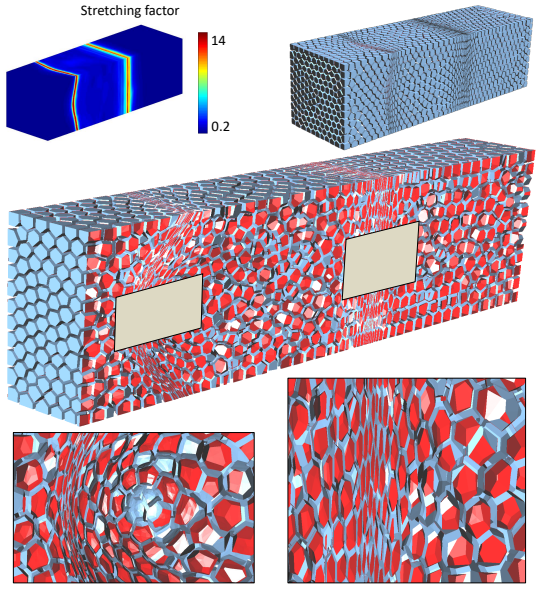


Fig. 20. The visualization of the tensor field by using 3D anisotropic volumetric RVD (with 10,000 samples) on the CFD model (courtesy of Adrien Loseille) with the stretching factor in the major direction $s_1(\mathbf{x}) \in [0.2, 14]$ generated by our 8D embedding method.

the computed embedding, we can generate the high-quality anisotropic mesh with 5000 vertices (i.e., $G_{min} = 0.29$, $G_{avg} = 0.91$, $\theta_{min} = 17.01°$, $\theta_{avg} = 53.23°$, and $\%_{<30°} = 0.4\%$).

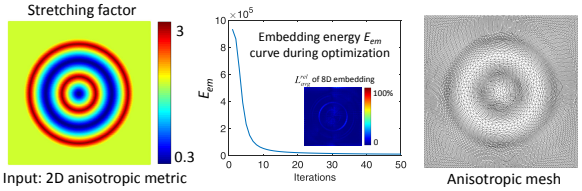


Fig. 21. Illustration of the embedding energy ($E_{em}$) curve during optimization of 50 iterations, 8D embedding errors, and anisotropic meshing result of an input 2D metric domain having concentric rings with smaller metrics (in small regions shown in the blue color) and larger metrics (in other regions).

*A 3D Metric.* Fig. 22 shows a CFD model with a complicated boundary and topology. It has a 3D metric field including small metric tensors buried in large metric tensors as shown in the closeup regions, and all other regions in small metric tensors (i.e., eigenvalues are less than one). With the stretching factors (eigenvalues) in the input metric $\in [0.2, 30]$, we need to multiply a constant value, 5, in this model. The embedding energy curve during optimization and 3D tensor field visualization result are given in Fig. 22. Our embedding framework can efficiently converge within 50 iterations with smaller relative edge length errors, e.g., $L^{rel}_{avg} = 2.13\%$ and $L^{rel}_{max} = 79.38\%$. Based on the computed embedding, we can generate the 3D anisotropic volumetric RVD result with 20,000 samples, which does well preserve the shapes and size ratios of Voronoi cells.

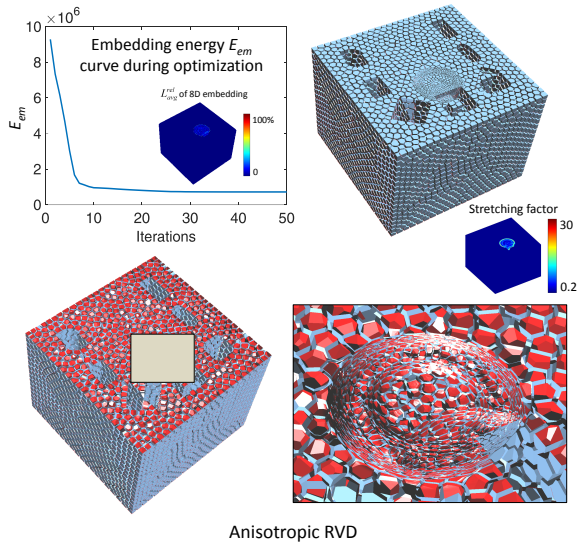


Fig. 22. Illustration of the embedding energy ($E_{em}$) curve during optimization of 50 iterations, 8D embedding errors, and 3D anisotropic volumetric RVD result (with 20,000 samples) of an input CFD model (courtesy of Adrien Loseille) with small metric tensors buried in large metric tensors (as shown in the closeup regions), and all other regions in small metric tensors.

## 7 DISCUSSION AND FUTURE WORK

To the best of our knowledge, this is the first article in the literature for computing the self-intersection free Euclidean embedding in arbitrary dimensions and using it in Voronoi diagram, surfaces and volume meshing equipped with Riemannian metrics. In particular, this is the first practical algorithm for computing volumetric anisotropic Voronoi diagrams. While there is a direct application to anisotropic meshing, we feel that there are many unexplored applications as limitations and future work: (1) We have not yet explored the cases where the input metric has sudden discontinuities, in such case, finding a practical embedding may be challenging. (2) In this work, we would like to emphasize that all the costs of computations in a high-d space are justified since we can obtain high-quality anisotropic RVD and meshes as compared with previous methods. In the future, we will improve the computational speed by using GPU-based parallel algorithm and implementation. (3) In order to generate the sliver-free tetrahedral meshes, we will apply some sliver suppressing strategies, such as perturbations [Tournois et al. 2009], sliver elimination [Fu et al. 2014] or gradient-based shape matching [Ni et al. 2017]. (4) Since the optimization strategy used in the embedding computation is not a global approach, the energy is possibly trapped in a local minimum when minimized at a certain number of iterations. In this case, some global optimization strategies can be applied, such as to perturb the high-d vertex positions to jump out of some local minima and keep searching for a rather-global one, e.g., Monte Carlo minimization, etc. Furthermore, in the optimization framework, some relaxation techniques for metric smoothness to handle singularities will be considered and studied. (5) Since in this paper, we only show the potential applications in 3D tensor field visualization, they are not yet the best results, and better visualization techniques combined with 3D AVD will be further investigated in our future work. (6) Besides anisotropic RVD and meshing, we will explore other important and interesting applications by using the proposed SIFHDE$^2$, such as simulations in medical imaging and computer animation. Especially, we will investigate more practical tasks where the absolute self-intersection free is a requirement that should be strictly enforced.

## ACKNOWLEDGMENTS

We would like to thank the reviewers for their valuable comments. We are grateful to Joshua A. Levine for the early discussions of this work, Adrien Loseille for sharing 3D CFD tensor datasets, TensorVis.org for providing 3D engineering tensor datasets, and the authors of the comparison work for sharing their results. This work was partially supported by the National Science Foundation under Grant Numbers ACI-1657364, CNS-16472000, IIS-1149737, the Wayne State University Subaward 4207299A of CNS-1821962, the Wayne State University Startup Grant, and the Natural Science Foundation of China under Grant Number 61572021.

## REFERENCES

F. Alauzet and A. Loseille. 2010. High-Order Sonic Boom Modeling Based on Adaptive Methods. *J. Comput. Phys.* 229, 3 (2010), 561–593.
F. Alauzet and A. Lozeille. 2016. A Decade of Progress on Anisotropic Mesh Adaptation for Computational Fluid Dynamics. *Computer-Aided Design* 72 (2016), 13–39.
P. Alliez, D. Cohen-Steiner, O. Devillers, B. Lévy, and M. Desbrun. 2003. Anisotropic Polygonal Remeshing. *ACM Trans. Graph.* 22, 3 (2003), 485–493.
P. Alliez, D. Cohen-Steiner, M. Yvinec, and M. Desbrun. 2005. Variational Tetrahedral Meshing. *ACM Trans. Graph.* 24, 3 (2005), 617–625.
J-D. Boissonnat, K. Shi, J. Tournois, and M. Yvinec. 2015a. Anisotropic Delaunay Meshes of Surfaces. *ACM Trans. Graph.* 34, 2 (2015), 14.

J-D. Boissonnat, C. Wormser, and M. Yvinec. 2008a. Anisotropic Diagrams: Labelle Shewchuk Approach Revisited. *Theoretical Computer Science* 408, 2-3 (2008), 163–173.

J-D. Boissonnat, C. Wormser, and M. Yvinec. 2008b. Locally Uniform Anisotropic Meshing. In *Proceedings of the 24th annual symposium on Computational geometry*. 270–277.

J-D. Boissonnat, C. Wormser, and M. Yvinec. 2015b. Anisotropic Delaunay Mesh Generation. *SIAM J. Comput.* 44, 2 (2015), 467–512.

H. Borouchaki, P. George, F. Hecht, P. Laug, and E. Saltel. 1997b. Delaunay Mesh Generation Governed by Metric Specifications. Part I. Algorithms. *Finite Elements in Analysis and Design* 25, 1–2 (1997), 61–83.

H. Borouchaki, P. George, and B. Mohammadi. 1997a. Delaunay Mesh Generation Governed by Metric Specifications. Part II. Applications. *Finite Elements in Analysis and Design* 25, 1–2 (1997), 85–109.

V. Borrelli, S. Jabrane, F. Lazarus, and B. Thibert. 2012. Flat Tori in Three-dimensional Space and Convex Integration. *Proceedings of the National Academy of Sciences of the United States of America* 109, 19 (2012).

F. Bossen and P. Heckbert. 1996. A Pliant Method for Anisotropic Mesh Generation. In *5th International Meshing Roundtable*. 63–76.

M. Bronstein, A.Bronstein, R. Kimmel, and I. Yavneh. 2000. Multigrid Multidimensional Scaling. *Numerical Linear Algebra with Applications* 0 (2000), 1–6.

M. Bronstein, A. Bronstein, R. Kimmel, and I. Yavneh. 2005. A Multigrid Approach For Multi-Dimensional Scaling. In *Proceedings of the Copper Mountain Conference on Multigrid Methods*.

M. Budninskiy, B. Liu, F. de Goes, Y. Tong, P. Alliez, and M. Desbrun. 2016. Optimal Voronoi Tessellations with Hessian-based Anisotropy. *ACM Trans. Graph.* 35, 6 (2016), 242:1–242:12.

G. Cañas and S. Gortler. 2006. Surface Remeshing in Arbitrary Codimensions. *Visual Computer* 22, 9 (2006), 885–895.

L. Chen, P. Sun, and J. Xu. 2007. Optimal Anisotropic Meshes for Minimizing Interpolation Errors in $L^p$-Norm. *Math. Comp.* 76 (2007), 179–204.

L. Chen and J. Xu. 2004. Optimal Delaunay Triangulations. *Journal of Computational Mathematics* 22 (2004), 299–308.

Z. Chen, W. Wang, B. Lévy, L. Liu, and F. Sun. 2014. Revisiting Optimal Delaunay Triangulation for 3D Graded Mesh Generation. *SIAM Journal Scientific Computing* (2014).

H-L. Cheng, T. Dey, H. Edelsbrunner, and J. Sullivan. 2001. Dynamic Skin Triangulation. *ACM-SIAM symposium on Discrete algorithms* 25 (2001), 525–568.

S. Cheng, T. Dey, and E. Ramos. 2006. Anisotropic Surface Meshing. In *Proceedings of ACM-SIAM Symposium on Discrete Algorithms*. 202–211.

F. Courty, D. Leservoisier, P-L. George, and A. Dervieux. 2006. Continuous Metric and Mesh Adaptation. *Applied Numerical Mathematics* 55 (2006), 117–145.

J. Dardenne, S. Valette, N. Siauve, N. Burais, and R. Prost. 2009. Variational Tetrahedral Mesh Generation from Discrete Volume Data. *The Visual Computer* 25, 5 (2009), 401–410.

F. Dassi, A. Mola, and H. Si. 2014. Curvature-Adapted Remeshing of CAD Surfaces. In *23rd International Meshing Roundtable*. 253–265.

F. Dassi, H. Si, S. Perotto, and T. Streckenbach. 2015. Anisotropic Finite Element Mesh Adaptation via Higher Dimensional Embedding. In *24th International Meshing Roundtable*. 265–277.

C. Dobrzynski and P. Frey. 2008. Anisotropic Delaunay Mesh Adaptation for Unsteady Simulations. In *17th International Meshing Roundtable*. 177–194.

Q. Du, V. Faber, and M. Gunzburger. 1999. Centroidal Voronoi Tessellations: Applications and Algorithms. *SIAM Rev.* 41, 4 (1999), 637–676.

Q. Du and D. Wang. 2005a. Anisotropic Centroidal Voronoi Tessellations and Their Applications. *SIAM Journal on Scientific Computing* 26, 3 (2005), 737–761.

Q. Du and D. Wang. 2005b. The Optimal Centroidal Voronoi Tessellations and the Gersho's Conjecture in the Three-dimensional Space. *Computers & Mathematics with Applications* 49, 9 (2005), 1355–1373.

M. Freidlin. 1968. On the Factorization of Non-Negative Definite Matrices. *Theory of Probability and Its Applications* 13, 2 (1968), 354–356.

P. Frey and F. Alauzet. 2005. Anisotropic Mesh Adaptation for CFD Computations. *Computer Methods in Applied Mechanics and Engineering* 194, 48-49 (2005), 5068–5082.

P. Frey and H. Borouchaki. 1997. Surface Mesh Evaluation. In *6th International Meshing Roundtable*. 363–373.

X. Fu, Y. Liu, J. Snyder, and B. Guo. 2014. Anisotropic Simplicial Meshing using Local Convex Functions. *ACM Trans. Graph.* 33, 6 (2014), 182:1–182:11.

G. Golub and C. Loan. 1996. *Matrix Computations (3rd Ed.)*. Johns Hopkins University Press, Baltimore, Maryland.

M. Gromov. 2017. Geometric, Algebraic, and Analytic Descendants of Nash Isometric Embedding Theorems. *Bull. Amer. Math. Soc.* 54, 2 (2017), 173–245.

M. Gromov and V. Rokhlin. 1970. Embeddings and Immersions in Riemannian Geometry. *Russian Mathematical Surveys* 25, 5 (1970), 1–57.

Q. Han and J-X. Hong. 2006. *Isometric Embedding of Riemannian Manifolds in Euclidean Spaces*. Vol. 13. American Mathematical Society.

P. Heckbert and M. Garland. 1999. Optimal Triangulation and Quadric-Based Surface Simplification. *Computational Geometry* 14, 1–3 (1999), 49–65.

J-X. Hong. 1993. Realization in $\mathbb{R}^3$ of Complete Riemannian Manifolds with Negative Curvature. *Communications in Analysis and Geometry* 1, 4 (1993), 487–514.

K. Itô. 1950. Stochastic Differential Equations in a Differentiable Manifold. *Nagoya Mathematical Journal* 1 (1950), 35–47.

B. Klingner. 2008. *Tetrahedral Mesh Improvement*. Ph.D. Thesis, Dep. of Elec. Eng. and Comp. Sciences U. of California at Berkeley.

B. Klingner and J. Shewchuk. 2007. Agressive Tetrahedral Mesh Improvement. In *Proceedings of 16th International Meshing Roundtable*. 3–23.

D. Kovacs, A. Myles, and D. Zorin. 2010. Anisotropic Quadrangulation. In *Proceedings of the 14th ACM Symposium on Solid and Physical Modeling (SPM '10)*. 137–146.

N. Kuiper. 1955. On $C^1$-isometric Embeddings I. In *Proceedings of the Koninklijke Nederlandse Akademie van Wetenschappen*. 545–556.

B. Lévy. 2016. Robustness and Efficiency of Geometric Programs: The Predicate Construction Kit (PCK). *Computer-Aided Design* 72 (2016), 3–12.

B. Lévy and N. Bonneel. 2012. Variational Anisotropic Surface Meshing with Voronoi Parallel Linear Enumeration. In *21st International Meshing Roundtable*. 349–366.

D. Liu and J. Nocedal. 1989. On the Limited Memory BFGS Method for Large Scale Optimization. *Mathematical Programming* 45, 3 (1989), 503–528.

Y. Liu, W. Wang, B. Lévy, F. Sun, D. Yan, L. Lu, and C. Yang. 2009. On Centroidal Voronoi Tessellation – Energy Smoothness and Fast Computation. *ACM Trans. Graph.* 28, 4 (2009), 101:1–101:17.

S. Lloyd. 1982. Least Squares Quantization in PCM. *IEEE Transactions on Information Theory* 28, 2 (1982), 129–137.

A. Loseille and R. Löhner. 2016. Anisotropic Mesh Generation for High-fidelity Simulations in CFD. *INRIA* (2016). preprint.

E. Marchandise, C. Geuzaine, and J.F. Remacle. 2013. Cardiovascular and Lung Mesh Generation Based on Centerlines. *International Journal for Numerical Methods in Biomedical Engineering* 29, 6 (2013), 665–682.

R. Narain, A. Samii, and J. F. O'Brien. 2012. Adaptive Anisotropic Remeshing for Cloth Simulation. *ACM Trans. Graph.* 31, 6 (2012), 147:1–147:10.

J. Nash. 1954. $C^1$-isometric embeddings. *Annals of Mathematics* 60, 3 (1954), 383–396.

S. Ni, Z. Zhong, Y. Liu, W. Wang, Z. Chen, and X. Guo. 2017. Sliver-suppressing Tetrahedral Mesh Optimization with Gradient-based Shape Matching Energy. *Computer Aided Geometric Design* 52 (2017), 247–261.

D. Panozzo, E. Puppo, M. Tarini, and O. Sorkine-Hornung. 2014. Frame Fields: Anisotropic and Non-Orthogonal Cross Fields. *ACM Trans. Graph.* 33, 4 (2014), 134:1–134:11.

M. Rouxel-Labbé, M. Wintraecken, and J-D. Boissonnat. 2016. Discretized Riemannian Delaunay Triangulations. *Procedia Engineering* 163 (2016), 97–109.

E. Sauvage, J. Remacle, and E. Marchandise. 2014. Metric Field Construction for Anisotropic Mesh Adaptation with Application to Blood Flow Simulations. *International Journal for Numerical Methods in Biomedical Engineering* 30, 11 (2014), 1326–1346.

K. Shimada and D. Gossard. 1995. Bubble Mesh: Automated Triangular Meshing of Non-manifold Geometry by Sphere Packing. In *Proceedings of the 3rd ACM Symposium on Solid Modeling and Applications*. 409–419.

K. Shimada, A. Yamada, and T. Itoh. 1997. Anisotropic Triangular Meshing of Parametric Surfaces via Close Packing of Ellipsoidal Bubbles. In *6th International Meshing Roundtable*. 375–390.

R. Simpson. 1994. Anisotropic Mesh Transformations and Optimal Error Control. *Applied Numerical Mathematics* 14, 1–3 (1994), 183–198.

O. Sorkine-Hornung and M. Alexa. 2007. As-rigid-as-possible Surface Modeling. In *Proceedings of the fifth Eurographics symposium on Geometry processing*. 109–116.

R. Sumner and J. Popović. 2004. Deformation Transfer for Triangle Meshes. *ACM Trans. Graph.* 23, 3 (2004), 399–405.

J. Tournois, R. Srinivasan, and P. Alliez. 2009. Perturbing Slivers in 3D Delaunay Meshes. *Proceedings of the 18th international meshing roundtable* (2009), 157–173.

S. Valette, J. Chassery, and R. Prost. 2008. Generic Remeshing of 3D Triangular Meshes with Metric-Dependent Discrete Voronoi Diagrams. *IEEE Trans. Vis. Comput. Graph.* 14, 2 (2008), 369–381.

N. Verma. 2012. Distance Preserving Embeddings for General n-Dimensional Manifolds. *Journal of Machine Learning Research volume* 14 (2012), 2415–2448.

S. Yamakawa and K. Shimada. 2000. High Quality Anisotropic Tetrahedral Mesh Generation via Packing Ellipsoidal Bubbles. In *9th International Meshing Roundtable*. 263–273.

D. Yan, B. Lévy, Y. Liu, F. Sun, and W. Wang. 2009. Isotropic Remeshing with Fast and Exact Computation of Restricted Voronoi Diagram. *Computer Graphics Forum* 28, 5 (2009), 1445–1454.

D. Yan, W. Wang, B. Lévy, and Y. Liu. 2013. Efficient Computation of Clipped Voronoi Diagram for Mesh Generation. *Computer-Aided Design* 45, 4 (2013), 843–852.

Z. Zhong, X. Guo, W. Wang, B. Lévy, F. Sun, Y. Liu, and W. Mao. 2013. Particle-Based Anisotropic Surface Meshing. *ACM Trans. Graph.* 32, 4 (2013), 99:1–99:14.

Z. Zhong, L. Shuai, M. Jin, and X. Guo. 2014. Anisotropic Surface Meshing with Conformal Embedding. *Graphical Models* 76, 5 (2014), 468–483.