arXiv:2410.23109v2 [cs.CV] 31 Oct 2024

NASM: Neural Anisotropic Surface Meshing

HONGBO LI, Wayne State University, USA
HAIKUAN ZHU, Wayne State University, USA
SIKAI ZHONG, Wayne State University, USA
NINGNA WANG, The University of Texas at Dallas, USA
CHENG LIN, The University of Hong Kong, China
XIAOHU GUO, The University of Texas at Dallas, USA
SHIQING XIN, Shandong University, China
WENPING WANG, Texas A&M University, USA
JING HUA, Wayne State University, USA
ZICHUN ZHONG*, Wayne State University, USA

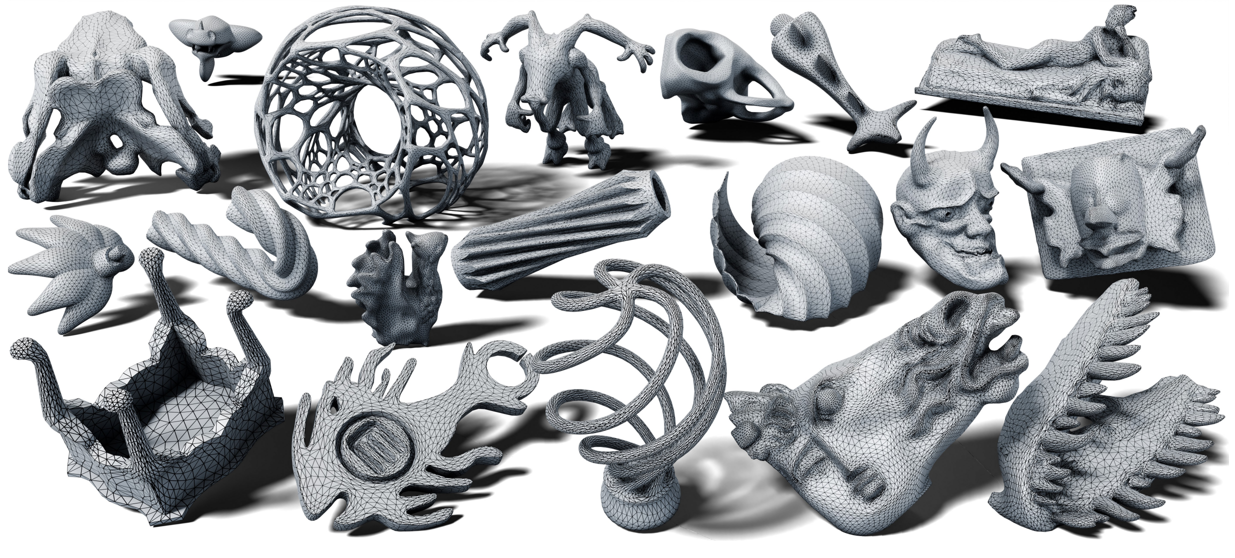


Fig. 1. A gallery of anisotropic surface meshes generated by our NASM method. These results are selected from our testing models in Thingi10K dataset, including complicated organic surfaces, and surfaces with sharp and weak features as well as varying anisotropic metrics.

This paper introduces a new learning-based method, NASM, for anisotropic surface meshing. Our key idea is to propose a graph neural network to embed an input mesh into a high-dimensional (high-d) Euclidean embedding space to preserve curvature-based anisotropic metric by using a dot product loss between high-d edge vectors. This can dramatically reduce the computational time and increase the scalability. Then, we propose a novel feature-sensitive remeshing on the generated high-d embedding to automatically capture sharp geometric features. We define a high-d normal metric, and then derive an automatic differentiation on a high-d centroidal Voronoi tessellation (CVT) optimization with the normal metric to simultaneously preserve geometric features and curvature anisotropy that exhibit in the original 3D shapes. To our knowledge, this is the first time that a deep learning framework and a large dataset are proposed to construct a high-d Euclidean embedding space for 3D anisotropic surface meshing. Experimental results are evaluated and compared with the state-of-the-art in anisotropic surface meshing on a large number of surface models from Thingi10K dataset as well as tested on extensive unseen 3D shapes from Multi-Garment Network dataset and FAUST human dataset.

*Corresponding author.

CCS Concepts: • **Mathematics of computing → Mesh generation**; • **Computing methodologies → Neural networks**.

Additional Key Words and Phrases: Anisotropic surface mesh, graph neural network, high-d Euclidean embedding, feature-sensitive meshing

ACM Reference Format:
Hongbo Li, Haikuan Zhu, Sikai Zhong, Ningna Wang, Cheng Lin, Xiaohu Guo, Shiqing Xin, Wenping Wang, Jing Hua, and Zichun Zhong. 2024. NASM: Neural Anisotropic Surface Meshing. In *SIGGRAPH Asia 2024 Conference Papers (SA Conference Papers '24), December 3–6, 2024, Tokyo, Japan.* ACM, New York, NY, USA, 21 pages. https://doi.org/10.1145/3680528.3687700

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SA Conference Papers '24, December 3–6, 2024, Tokyo, Japan
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-1131-2/24/12.
https://doi.org/10.1145/3680528.3687700

1 INTRODUCTION

In geometric modeling, physical simulation, and mechanical engineering fields, anisotropic meshes are crucial for performing better shape approximations [Simpson 1994] and achieving higher accuracy in numerical simulations [Alauzet and Loseille 2010; Narain et al. 2012]. Anisotropic surface meshes are triangulations with elements elongated along prescribed directions and stretchings. One of the fundamental geometric merits is that, with a given number of mesh elements (i.e., vertices or triangles), the L_2 optimal approximation to a smooth surface is achieved when the anisotropy of triangles conforms to the eigenvalues and eigenvectors of the curvature tensors [Heckbert and Garland 1999; Simpson 1994]. This improves the simulation's fidelity, stability, and efficiency. They are vital for high-fidelity structural analysis, such as in the study of turbine blades, aircraft wings, or biomedical implants. Due to the difficulty in anisotropic meshing, recently some researchers [Zhong et al. 2013, 2018] proposed a computational method to map the complicated anisotropic 3D space onto a higher dimensional Euclidean space, which can make the anisotropic mesh generation computation simpler. This research line is inspired by Nash Embedding Theory [Kuiper 1955; Nash 1954]. However, the major bottleneck of the high-dimensional (high-d) embedding method is time-consuming in the computation. Some other high-d embedding-based methods can only use specific embeddings, such as normal-based embedding, which cannot consider arbitrary input metrics [Dassi et al. 2014, 2015; Lévy and Bonneel 2013].

Besides that, most existing anisotropic meshing methods [Alliez et al. 2003; Boissonnat et al. 2015a; Bossen and Heckbert 1996; Du and Wang 2005; Fu et al. 2014; Valette et al. 2008; Zhong et al. 2013, 2018] need to have a given metric as input to control the element stretching ratio and orientation, which is tedious and unrobust. Furthermore, in order to handle geometric sharp or weak features, users need to identify these feature edges and corners in the input reference mesh at first, and then constrain the mesh vertex position and connectivity along the feature edges or fix the feature corners during optimization. [Lévy and Liu 2010] and [Xu et al. 2024] extend the CVT objective function by a metric term, which can automatically attract the site point onto the feature lines in 3D space so as to naturally recover the features. However, it can only handle with isotropic meshing. To our knowledge, there is no method which can generate anisotropic mesh to preserve both metric field as well as sharp / weak features. Moreover, all the existing methods for anisotropic meshing are model-based approaches, which are not scalable to generate a large number of anisotropic meshes from a variety of shape geometries and typologies.

In this work, we address the above-mentioned challenges in anisotropic mesh generation. It includes main twofold: (1) how to develop a learning-based method to efficiently and robustly compute a high-d embedding without providing a pre-computed metric field; (2) how to generate high-fidelity and high-quality anisotropic surface meshes with automatical feature preserving. The *main contributions* are as follows:

- Develop a scalable computational paradigm to generate high-quality anisotropic surface meshes from arbitrary input meshes only (no curvature metric is needed);
- Design an efficient GNN-based method with high-d dot product loss to embed an input mesh into a high-d Euclidean embedding space to preserve curvature-based anisotropic metric (a speedup of about $1,500\times$ times);
- Define a high-d normal metric CVT optimization with an automatic differentiation to compute the feature-sensitive anisotropic surface meshes (without any user's input on tagging features);
- Construct a large dataset for anisotropic surface meshing and processing (more than 800+ mesh models from Thingi10K, Multi-Garment Network, and FAUST datasets).

The overview pipeline of the proposed neural anisotropic surface meshing (NASM) approach is shown in Fig. 2.

2 RELATED WORKS

In this section, we review the related literature on anisotropic triangle meshing approaches and neural geometric learning on meshes.

2.1 Anisotropic Triangle Meshing

In terms of meshing a surface, anisotropic is related to optimal approximation of a discretization of the surface function with a given number of element count [Alliez et al. 2003]. Anisotropic triangular meshing has been extensively studied over decades, both on flat, 2D regions [Bossen and Heckbert 1996], and the Euclidean 3D surface [Shimada et al. 1997]. [Shapiro et al. 1996] introduces the Adaptive Smoothed Particle Hydrodynamics (ASPH) which uses inter-particle Gaussian kernels with an anisotropic metric tensor. [Zhong et al. 2013] further extends the formulation of the energy between particles to a higher embedding space which leads to enhanced flexibility and accuracy when confronted with either mild or significant variations in the metric.

2.1.1 Anisotropic Centroidal Voronoi Tessellation. [Du and Wang 2005] further generalizes the concept of CVT to the anisotropic centroidal Voronoi tessellation (ACVT) by integrating the given Riemannian metric into the calculation. However, the Riemannian metric needs to be constructed in each Lloyd [Lloyd 1982] iteration, which is not efficient. [Valette et al. 2008] provides a discrete approximation of ACVT to accelerate the computation speed with the cost of degraded mesh quality. [Zhong et al. 2014] computes the CVT on an 2D parametric domain where the metric surface can be conformally mapping to. [Lévy and Bonneel 2013] leverages the embedding theory [Nash 1954] and formulates a 6D space where the surface mesh can be embedded in, followed with the CVT isotropic remeshing within this extended dimensionality. The final results reveal the distinctive anisotropic characteristics, albeit not distributed across the entire surface. [Zhong et al. 2018] introduces a variational approach to compute the higher dimensional space through aligning the Jacobian of transformation and the gradient of deformation between the 3D space and the high-d space where the surface is embedded. This approach considers the metric space defined over the surface which leads the result with anisotropic properties regarding to the given metric. However, the computation is involved in solving a linear system w.r.t. the input mesh resolution, which demands a significant investment of time.

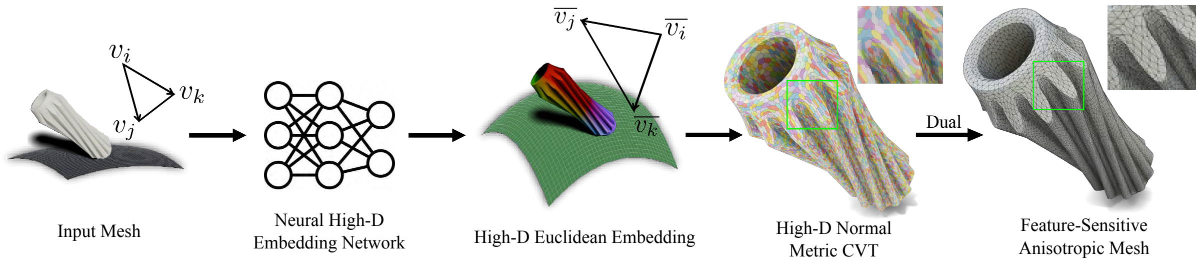


Fig. 2. The overview pipeline of the proposed neural anisotropic surface meshing (NASM) approach. Our method includes two main components: neural high-d Euclidean embedding and high-d normal metric CVT for feature-sensitive anisotropic meshing. The training data for neural high-d Euclidean embedding is generated based on SIFHDE^2 [Zhong et al. 2018]. More details are given in Section A of Supplemental Document.

2.1.2 *Anisotropic Delaunay Triangulation.*

Many works have extended the Delaunay triangulation to the anisotropic case, both through the refinement-based approaches [Dobrzynski and Frey 2008; Frey and Borouchaki 1999] and the variational approaches [Chen et al. 2007; Chen and Xu 2004]. [Boissonnat et al. 2015a,b] leverages a Delaunay refinement scheme that progressively inserts vertices to reach the final anisotropic meshes. [Rouxel-Labbé et al. 2016] implements an algorithm for the computation of the discrete approximations to Riemannian Voronoi diagrams on 2-manifolds employing numerical methods for calculating the geodesic distances. [Fu et al. 2014] presents a Locally Convex Triangulation (LCT) method to integrate anisotropic into optimal Delaunay triangulation, which constructs convex functions that locally match the predefined Riemaninain metric. [Budninskiy et al. 2016] introduces a variational method to lift the points onto a convex function, and minimize the error between the lifted function and the constructed mesh. The utilization of the convex function instead of commonly used paraboloid integrates the anisotropy in final result. However, their limitation is only applicable to a small class of anisotropies that can be represented by convex functions.

2.1.3 *Mesh Processing on High Dimensional Space.*

As every smooth Riemannian manifold can be isometrically embedded into some Euclidean space [Nash 1954], several mesh processing tasks related to Riemannian manifold have been studied in 3D or higher dimensions. [Panozzo et al. 2014] computes a 3D embedding via surface deformation to obtain anisotropic meshing results. [Dassi et al. 2014] follows the 6D configurations the same as in [Lévy and Bonneel 2013] and proposed local mesh operations to better preserve important geometric features on CAD models. [Dassi et al. 2015] configures the extended dimensions with a smooth function or the solution of a partial differential equation, whereas they consider only the case in the 2D space. [Zhong et al. 2018] recently proposes to directly calculate the high dimension counterpart and use the coordinates of original dimension as the first three dimensions to achieve a self-intersection free high-d embedding mesh.

2.2 Neural Geometric Learning on Meshes

2.2.1 *Non-Graph-Based Neural Network.*

Recent advances in deep learning facilitate the use of learnable components in 3D modeling applications. One way of applying learnable methods is to treat 3D geometry models using regular data structures, whereupon the established learning paradigms in the 2D image domain can be seamlessly extended, providing a straightforward mean of processing and analyzing. For instance, geometry models can be described with 3D grids of values and 3D Convolutional Neural Network (CNN) can be applied [Chen et al. 2021; Mescheder et al. 2019; Wang et al. 2017, 2018a]. The other way is to directly handle the irregular inherence of the geometry structure. An earlier work [Ivrissimtzis et al. 2004] proposes a learning algorithm for surface reconstruction by simulating an incrementally expanding neural network which learns a point cloud through a learning process. Recently, PointNet [Qi et al. 2017a] and PointNet++ [Qi et al. 2017b] stand as pioneering methods in this direction. Nevertheless, lack of explicit connectivity information acquires intensive computational demands and limited representational capacity for the point structure. Surface mesh, as a conventional widely adopted structure, has been used for neural geometric learning and manifest favorable prospects. Recent surveys [Bronstein et al. 2017; Xiao et al. 2020] can be referred.

2.2.2 *Graph-Based Neural Network.*

A common representation for irregular data is the graph structure. As mesh data being a special type of graph structure, several works have advanced to leverage graph-based neural networks (GNNs) [Hamilton et al. 2018; Kipf and Welling 2017] and their spectral variants [Bruna et al. 2014; Defferrard et al. 2016; Kostrikov et al. 2018] on classic discrete mesh problem. [Sharp et al. 2022; Smirnov and Solomon 2021] extend the Laplacian operator to apply learning on the graph. [Hu et al. 2022] introduces a uniform downsampling scheme into the learning pipeline. [Pfaff et al. 2020] leverages the GNN on simulation tasks. [Hanocka et al. 2019] constructs an edge-based graph and defines convolution on it. [Wang et al. 2018b] generates 3D shape from 2D by updating a coarse mesh through GNN. [Potamias et al. 2022] leverages GNN to predict the connectivity in mesh simplification. The closest application to our work is [Pang et al. 2023], which exploits the neural network to map the meshes to high-d embedding and calculate the geodesic on the surface. However, their method is not designed for learning Riemannian metric and their loss function is not effective on anisotropic mesh generation. We conduct experiments to show that the loss functions as they proposed, e.g., L1 or L2 loss, present inferior quality in our meshing results.

3 NEURAL HIGH-D EUCLIDEAN EMBEDDING

As illustrated in Fig. 2, our method takes a triangle surface mesh as input and computes the mapping that isometrically embeds the mesh vertices to a high dimensional (high-d) Euclidean space. The outputs are the vertex-wise coordinates of this high-d space. In the following subsections, we discuss the main technical components of our neural high-d Euclidean embedding method in detail: high-d Euclidean embedding loss and the network architecture. Due to the page limit, data generation for embedding and meshing is given in Section A of Supplemental Document.

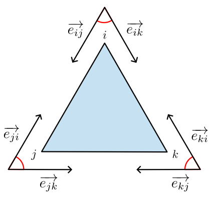
3.1 High-D Euclidean Embedding Loss

Anisotropy represents how distances and angles are distorted, which can be measured by the dot product in geometry. For a metric defined over the surface domain Ω, $M(.) \in \mathbb{R}^3$, at a given point $\mathbf{x} \in \Omega$, the dot product between two vectors $\mathbf{a}$ and $\mathbf{b}$ that starting from $\mathbf{x}$ is denoted by $\langle \mathbf{a}, \mathbf{b} \rangle_{M(\mathbf{x})}$, which is defined over the tangent space of the surface: $\langle \mathbf{a}, \mathbf{b} \rangle_{M(\mathbf{x})} = \mathbf{a}^t \mathbf{M}(\mathbf{x}) \mathbf{b}$.

Our goal is to construct a high-d space $\mathbb{R}^{\overline{m}}$, in which the surface can be embedded with Euclidean metric. Inspired by [Nash 1956], considering the high-d corresponding vectors $\overline{\mathbf{a}}$ and $\overline{\mathbf{b}}$, the dot product between $\overline{\mathbf{a}}$ and $\overline{\mathbf{b}}$ should introduce the same meaning of the dot product between $\mathbf{a}$ and $\mathbf{b}$ under the given metric, namely:

$$\langle \overline{\mathbf{a}}, \overline{\mathbf{b}} \rangle = \langle \mathbf{a}, \mathbf{b} \rangle_{M(\mathbf{x})}. \tag{1}$$

Equation (1) can be proved by the pullback metric. Considering the smooth mapping ϕ between $\mathbb{R}^3$ and $\mathbb{R}^{\overline{m}}$ at the point $\mathbf{x}$, the transformation between the vector in $\mathbb{R}^3$ and its correspondence in $\mathbb{R}^{\overline{m}}$ can be defined by the Jacobian matrix $\mathbf{J}(\mathbf{x})$ of ϕ, in essence, $\overline{\mathbf{a}} = \mathbf{J}(\mathbf{x})\mathbf{a}$ and $\overline{\mathbf{b}} = \mathbf{J}(\mathbf{x})\mathbf{b}$. Therefore, $\langle \overline{\mathbf{a}}, \overline{\mathbf{b}} \rangle = \mathbf{a}^t \mathbf{J}(\mathbf{x})^t \mathbf{J}(\mathbf{x}) \mathbf{b} = \langle \mathbf{a}, \mathbf{b} \rangle_{M(\mathbf{x})}$, where $\mathbf{M}(\mathbf{x}) = \mathbf{J}(\mathbf{x})^t \mathbf{J}(\mathbf{x})$.


$\overrightarrow{e_{ij}}$
$\overrightarrow{e_{ik}}$
i
$\overrightarrow{e_{ji}}$
$\overrightarrow{e_{ki}}$
j
k
$\overrightarrow{e_{jk}}$
$\overrightarrow{e_{kj}}$

Our designed loss for neural Euclidean embedding leverages the advantage that the dot product between the vertices in $\mathbb{R}^{\overline{m}}$ has already contained their metrics in $\mathbb{R}^3$. Additionally, considering the discrete nature of mesh, a metric can be interpreted as influencing a piecewise linear region and distorted the dot product defined within the region. We can view those regions as the simplices that constitute the entire manifold surface. Therefore, we define our loss on the dot product between two edge vectors defined on an internal face angle via Mean Squared Error:

$$\begin{aligned}\mathcal{L}_{dot} = \frac{1}{3|\mathbb{F}|} \sum_{\mathcal{F} \in \mathbb{F}} & \left| \langle \xi(\mathbf{e}_{ij}), \xi(\mathbf{e}_{ik}) \rangle - \langle \overline{\mathbf{e}}_{ij}, \overline{\mathbf{e}}_{ik} \rangle \right|^2 \\ & + \left| \langle \xi(\mathbf{e}_{jk}), \xi(\mathbf{e}_{ji}) \rangle - \langle \overline{\mathbf{e}}_{jk}, \overline{\mathbf{e}}_{ji} \rangle \right|^2 \\ & + \left| \langle \xi(\mathbf{e}_{ki}), \xi(\mathbf{e}_{kj}) \rangle - \langle \overline{\mathbf{e}}_{ki}, \overline{\mathbf{e}}_{kj} \rangle \right|^2,\end{aligned} \tag{2}$$

where $\xi(\mathbf{e}_{ij}) = f(\mathbf{v}_j) - f(\mathbf{v}_i)$, $f(\mathbf{v}_i)$ and $f(\mathbf{v}_j)$ are predicted high-d edge vector and vertex coordinates, which are learned from the input 3D coordinates; $\overline{\mathbf{e}}_{ij} = \overline{\mathbf{v}}_j - \overline{\mathbf{v}}_i$, $\overline{\mathbf{v}}_i$ and $\overline{\mathbf{v}}_j$ are the ground truth high-d edge vector and vertex coordinates. The ground truth data is geneated based on SIFHDE^2 [Zhong et al. 2018] (refer Section A of Supplemental Document for details). Similar definitions are made for other edge vectors and vertices. So, each triangle face $\mathcal{F}$ has three dot product loss terms.

The dot product loss can well align the metric distortion of the learned Euclidean embedding surface with the ground truth, but neural network tends to overfit the local distortion. To solve this problem, we add a Laplacian loss term as regularization:

$$\mathcal{L}_{lap} = \sum_{i \in \mathcal{V}} \left\| \frac{\sum_{j \in N(i)} (f(\mathbf{v}_i) - f(\mathbf{v}_j))}{|N(i)|} - \frac{\sum_{j \in N(i)} (\overline{\mathbf{v}}_i - \overline{\mathbf{v}}_j)}{|N(i)|} \right\|_2^2, \tag{3}$$

where $N(i)$ is the set of one-ring neighbors of vertex i.

Thus, our total loss is defined as the weighted sum of two losses:

$$\mathcal{L} = \mathcal{L}_{dot} + w_{lap} \mathcal{L}_{lap}, \tag{4}$$

where $w_{lap} = 0.1$ is based on extensive experiments. The analysis of w_{lap} is discussed in Section F.2 of Supplemental Document.

3.2 High-D Euclidean Embedding Network

In order to take advantage of the connectivity of mesh, we utilize the graph neural network (GNN) to learn the high-d embedding. Specifically, given an surface mesh $\Omega \in \mathbb{R}^3$, its vertices $\mathcal{V}$ and edges $\mathcal{E}$ naturally define an undirected graph. We introduce the details about our network design in this section.

3.2.1 *Graph Convolution.*

The main purpose of graph convolution layer is to learn how to aggregate feature information from a node's local neighborhood and how to update its own feature. We employ the updating scheme based on [Hamilton et al. 2018] for our high-d embedding task. The convolution layer in our network follows:

$$f_i^{k+1} = \mathcal{W}_0^{k+1} \Big(f_i^k, \max_{j \in N(i)} \mathcal{W}_1^{k+1} \mathcal{A}(f_i^k, f_j^k) \Big), \tag{5}$$

where $\mathcal{W}_0$ and $\mathcal{W}_1$ are learnable parameters. f^{k+1} and f^k are the feature vectors on vertex i before and after the convolution. $\mathcal{N}(i)$ is the set of one-ring neighbors of vertex i. $\mathcal{A}$ is a differentiable function to process feature information through the edge between vertex i and j.

In pursuit of tailoring the convolution layer for learning high-d embedding, we want the learned extended coordinates to compensate the distortion / deformation followed by the metric. Therefore, we explicitly incorporate the direction and distance between each vertex and its neighbors into the aggregated feature information. $\mathcal{A}$ is accordingly defined as:

$$\mathcal{A}\Big(f_i^k, f_j^k\Big) = \Big[f_j^k, f_j^k - f_i^k, \|f_j^k - f_i^k\|_2 \Big]. \tag{6}$$

3.2.2 *Network Architecture.*

We construct a Graph U-Net [Gao and Ji 2019] with our tailored message passing paradigm. The network takes a 3D surface mesh as input, and embeds each vertex into the high-d space. The output keeps the same mesh connectivity as input. Fig. 3 illustrates the architecture of the proposed high-d Euclidean embedding network. We build our residual blocks by two layers of graph convolution with skip connection followed by a batch normalization [Ioffe and Szegedy 2015] and a Leaky ReLU activation [Maas et al. 2013]. Each down-sampling block is constituted by a residual block and one layer of Top-K pooling [Gao and Ji 2019]. Each up-sampling block is constituted by a residual block and one layer of interpolation. We employ five down-sampling blocks and five up-sampling layers. The network input is composed with six channels for each vertices, including vertex coordinates and normals. The output has five channels, and we concatenate

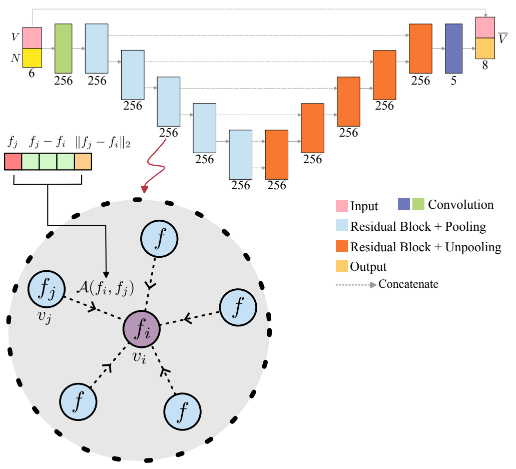


Fig. 3. The architecture of the proposed high-d Euclidean embedding network. Each residual block combines two graph convolution layers with skip-connection. Each convolution layer is followed by a normalization layer and an activation layer except the output layer. The numbers represent the feature dimensions of each network layer. The graph convolution computation and neighboring feature aggregation are illustrated in detail.

the original 3D coordinates in front of the output to form a self-intersection free 8D embedding coordinates as in [Zhong et al. 2018].

3.2.3 Training Data Augmentation. The diversity in shape meshes within our training set introduces biases in various aspects, particularly in the stretching direction and the distribution of stretching ratio across different parts of the mesh. To mitigate these issues and ensure a more robust and generalized model performance, we augment the training data by rotation (by $\pi/2$ around three Euclidean axes), and mirroring (according to xy, xz, and yz planes). These two augmentation types reorient the stretching direction and redistribute the stretching ratio according to vertex coordinates which ensure the anisotropy property is represented more uniformly.

4 FEATURE-SENSITIVE ANISOTROPIC MESHING

Remeshing 3D surfaces with features is a challenging problem, not to mention anisotropic remeshing. Existing anisotropic remeshing approaches [Fu et al. 2014; Zhong et al. 2018] let the user specify which edges correspond to features, and use constrained optimization to sample them properly. This is tedious in practical for 3D surface shapes with different features. In this work, after mapping the surface $\Omega \in \mathbb{R}^3$ into the neural high-d Euclidean space, the follow-up task is to isotropically remesh this high-d embedded surface $\overline{\Omega} \in \mathbb{R}^{\overline{m}}$. Inspired by [Lévy and Liu 2010], we re-design the normal metric from 3D to high-d space, and then formulate a new optimization energy function for high-d CVT ($d = 8$ in our case) with quadratic normal metric to automatically preserve features and anisotropy that exhibit in the original 3D shapes.

4.1 Normal Metric in High-D

The original normal metric $\mathbf{M}^{\mathcal{T}}$ associated with facet $\mathcal{T}$ in 3D is defined as follows [Lévy and Liu 2010]: $\mathbf{M}^{\mathcal{T}} = (s-1)\begin{pmatrix} \mathbf{N}_x^{\mathcal{T}}[\mathbf{N}^{\mathcal{T}}]^t \\ \mathbf{N}_y^{\mathcal{T}}[\mathbf{N}^{\mathcal{T}}]^t \\ \mathbf{N}_z^{\mathcal{T}}[\mathbf{N}^{\mathcal{T}}]^t \end{pmatrix} +$ $\mathbf{I}_{3\times 3}$, where $\mathbf{N}^{\mathcal{T}}$ denotes the unit normal of facet $\mathcal{T}$ in 3D and s is a factor to emphasize the normal metric ($s = 7$ in the experiments).

In our high-d embedded surface, since we only focus on the features that are identified in the original 3D surface, we can extend $\mathbf{M}^{\mathcal{T}}$ by padding 1s on the diagonal of $\mathbf{M}^{\mathcal{T}}$ and 0s otherwise to define the normal metric $\overline{\mathbf{M}}^{\mathcal{T}}$ in the high-d space $\mathbb{R}^{\overline{m}}$:

$$\overline{\mathbf{M}}^{\mathcal{T}} = \begin{pmatrix} \mathbf{M}^{\mathcal{T}} & 0 \\ 0 & \mathbf{I}_{hd} \end{pmatrix}, \tag{7}$$

where $\mathbf{I}_{hd}$ is an $(\overline{m}-3)\times(\overline{m}-3)$ identity matrix. $\overline{\mathbf{M}}^{\mathcal{T}}$ is an orthogonal matrix, which does not affect the Euclidean distance calculated from the high-d space. In this way, the proposed high-d quadratic normal metric can penalize the remeshing vertices that are far away from the tangent plane of the high-d embedding. On a feature edge, the combined effects of the normal metrics of both facets incident to a feature edge tend to attract remeshing vertices onto such edge.

4.2 High-D Normal Metric CVT

Inspired [Lévy and Liu 2010] (in 3D), we define the combinatorial structure and algebraic structure of high-d normal metric CVT. To compute high-d CVT is to minimize the energy function E_{hd}. We use gradient-based optimization method which needs to evaluate E_{hd} and its gradient ∇E_{hd}. For each iteration, the high-d embedding surface is first decomposed into a set of simplicial triangle facets through a differentiable clipping algorithm (in Section B of Supplemental Document). By doing so, the expression of E_{hd} can be simply evaluated. After having the combinatorial representation, the value of E_{hd} is obtained by the closed form and ∇E_{hd} can be obtained by applying chain rule and reverse-mode differentiation (in Section C of Supplemental Document).

The objective function of feature-sensitive high-d CVT is defined by adding a high-d normal metric $\overline{\mathbf{M}}^{\mathcal{T}}$ into CVT energy in high-d:

$$E_{hd}(\overline{\mathbf{X}}) = \sum_i \int_{\overline{\Omega}_i \cap \mathcal{S}} \left\| \overline{\mathbf{M}}^{\mathcal{T}}[\overline{\mathbf{y}} - \overline{\mathbf{x}}_i] \right\|_2^2 d\overline{\mathbf{y}}, \tag{8}$$

where $\overline{\mathbf{x}}_i \in \overline{\mathbf{X}}$ are the Voronoi cell sites. For RVD on the high-d embedded surface mesh, Voronoi cells are discretized to a set of facet triangles, denoted as $\mathcal{T}(\overline{\mathbf{x}}_0, \mathbf{C}_1, \mathbf{C}_2, \mathbf{C}_3)$. $\mathbf{C}_1, \mathbf{C}_2, \mathbf{C}_3$ are three different configurations of the vertices on these facets triangles, which need to be treated differently in order to have the accurate gradient. Details are given in Section B of Supplemental Document. So $\overline{\Omega}$ is the sum of the area of all the facets. A high-d CVT is a stable and critical point of E_{hd} during the optimization.

As stated in [Lévy and Liu 2010], given the gradient of standard CVT energy E, $\nabla E = 2m_i(\mathbf{x}_i - \mathbf{g}_i)$, one cannot simply replace the centroid $\mathbf{g}_i$ and mass m_i with their anisotropic counterparts to get the gradient of high-d CVT energy E_{hd}, since the anisotropy varies between two adjacent cells. The closed form for gradient of high-d

CVT with normal metric over an integration simplex $\mathcal{T}$ is:

$$\frac{dE^{\mathcal{T}}_{hd}(\overline{\mathbf{x}}_0, \mathbf{C}_1, \mathbf{C}_2, \mathbf{C}_3)}{d\overline{\mathbf{X}}} = \frac{dE^{\mathcal{T}}_{hd}}{d\overline{\mathbf{x}}_0} + \sum_{i=1,2,3} \frac{dE^{\mathcal{T}}_{hd}}{d\mathbf{C}_i}\frac{d\mathbf{C}_i}{d\overline{\mathbf{X}}}, \tag{9}$$

where $\overline{\mathbf{X}}$ is the site point of the Voronoi cell. $\mathcal{T}(\mathbf{C}_1, \mathbf{C}_2, \mathbf{C}_3)$ is one of the facet triangles that compose the restricted Voronoi cell in the high-d embedding space.

4.3 Auto Differentiation for High-D Normal Metric CVT

Automatic differentiation is a computational technique that leverages the chain rule of calculus. Instead of computing gradients from an explicit formula, automatic differentiation computes gradient starting from the function value, and tracing backwards along how the function value has been computed; and leverages the chain rule to compute the gradient.

From Equation (15), the calculation of gradient can be separated to two parts: $\frac{dE^{\mathcal{T}}_{hd}}{d\mathbf{C}_i}$ and $\frac{d\mathbf{C}_i}{d\overline{\mathbf{X}}}$, and the total gradient can be assembled from these two expressions. In this subsection, we introduce how to calculate $E^{\mathcal{T}}_{hd}$ and $\mathbf{C}_i$ in the forward pass in order to get the correct gradient $\frac{dE^{\mathcal{T}}_{hd}}{d\mathbf{C}_i}$ and $\frac{d\mathbf{C}_i}{d\overline{\mathbf{X}}}$ in the reverse pass.

For $E^{\mathcal{T}}_{hd}$, it can be discretized onto the triangle facets that compose the restricted Voronoi cell, and the expression for this discretization can be shortly expressed as:

$$E^{\mathcal{T}}_{hd} = |\mathcal{T}|F^{\mathcal{T}}_{hd}, \tag{10}$$

where $|\mathcal{T}|$ denotes the area of current triangle facet, referring Section A in [Lévy and Liu 2010]'s Appendix for details.

It is noted that our computation is more complicated and challenging than [Lévy and Bonneel 2013], which also uses Heron's formula to calculate the area of triangle in 6D, but without having the metric in their CVT energy function. So, their 6D CVT optimization does not require to compute the gradient of Heron's formula. Conversely, for our high-d normal metric CVT, $|\mathcal{T}|$ becomes dependent regards to E^{T}_{hd}:

$$\frac{dE^{\mathcal{T}}_{hd}}{d\mathbf{C}_i} = (\frac{dF^{\mathcal{T}}_{hd}}{d\mathbf{U}_i}|\mathcal{T}| + \frac{d|\mathcal{T}|}{d\mathbf{U}_i}F^{\mathcal{T}}_{hd})\overline{\mathbf{M}}^{\mathcal{T}}, \tag{11}$$

where $\mathbf{U}_i$ is defined in Equation (3) of Supplemental Document. The detailed computations of automatic differentiation on $\frac{dE^{\mathcal{T}}_{hd}}{d\mathbf{C}_i}$ and the derivative of $\mathbf{C}_i$ are provided in Sections D.1 and D.2 of Supplemental Document, respectively.

4.4 Restricted Voronoi Diagram and Mesh Generation

Following the optimization of the CVT in high-d space, the barycentric coordinates of each site point can be utilized to back-project the RVD from the high-d embedding space onto the original three-dimensional space. This process results in the generation of the final anisotropic RVD and dual mesh. We leverage the advantage of Geogram [Lévy 2015], which computes the RVD using filtered geometric predicates and symbolic perturbation to resolve degeneracies [Lévy 2016]. In general, there is a possibility that when generating a mesh using dual mesh of RVD, i.e., the associated RDT, inverted elements may occur upon back-projection to 3D space. Our implementation addresses this issue by inserting additional points using a provably terminating algorithm [Rouxel-Labbé et al. 2016] whenever such an inverted element is detected.

5 EXPERIMENTAL RESULTS

5.1 Datasets

In this work, the evaluations and applications mainly focus on better and faster approximating shapes by generating the anisotropic surface meshes on a large scale.

5.1.1 Benchmark. In all of our experiments, we train our network on the selected models from Thingi10k dataset [Zhou et al. 2016]. We select 240 meshes (after applying the data augmentation strategy proposed in Section 3.2, there are 2,400 meshes in our experiments) for training and 280 meshes for testing. Meshes are selected to ensure that our neural network can effectively learn curvature-related information. In the dataset, we only exclude surface models predominantly composed of planar or zero mean curvature regions where there are no anisotropic / curvature properties at all. Majority of meshes in our current training dataset contain planar regions. This is already sufficient for our NASM to learn how to effectively apply isotropic distribution in high-d spaces to planar regions as demonstrated in our results. We use the surface meshes that are generated by TetWild [Hu et al. 2018] and normalize them to [-1, 1] to make meshes with evenly distributed vertices and valid faces for building the ground truth data and evaluation use.

5.1.2 Generalization. To validate the generalization ability, we further evaluate our method on two large-scale unseen datasets with different types of meshes. There are 154 garment mesh models selected from Multi-Garment Net (MGN) [Bhatnagar et al. 2019], a dataset of clothes with open boundary, including 96 pants and 58 tops. Another dataset is 3D human FAUST [Bogo et al. 2014] with 200 human scan models of 10 different subjects in 30 different poses. We test these two datasets by using the same network parameters trained on Thingi10k training set without any fine-tuning.

5.2 Implementation Details

For curvature metric generation, we first use Libigl [Jacobson et al. 2018] to calculate the curvatures and principal directions. The curvature metric is designed as follows: $\mathbf{M} = [\mathbf{v}_{min}, \mathbf{v}_{max}, \mathbf{n}]diag(1, (\frac{s_2}{s_1})^2, 1)[\mathbf{v}_{min}, \mathbf{v}_{max}, \mathbf{n}]^t$, 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 stretching factors along principal directions. For surface anisotorpic task, $s_1 = \sqrt{|K_{min}|}$ and $s_2 = \sqrt{|K_{max}|}$ where K_{min} and K_{max} are the principal curvatures. We set small thresholds to preserve K_{min} and K_{max} not vanishing. To obtain smooth stretching factors, we compose $\frac{s_2}{s_1}$ with the principal curvatures calculated by Libigl and then apply weighted average over the one-ring neighborhood.

For ground truth high-d embedding generation, we implement the process in Section A of Supplemental Document by using Eigen [Guennebaud et al. 2010] and leverage MUMPS [Amestoy et al. 2001] sparse solver to accelerate the computation on AMD processor.

For the neural network development, we use PyTorch Geometric [Fey and Lenssen 2019] to build the network and employ AdamW [Loshchilov and Hutter 2019] optimizer with a learning rate 0.01 for training. We train our network for 600 epochs with batch size 4, and the learning rate is halved every 100 epochs.

For the high-d normal metric CVT development, we use Geogram [Lévy 2015] for restricted Voronoi diagram (RVD) calculation, Stan Math Library [Carpenter et al. 2015] for auto differentiation gradient computation, and L-BFGS [Liu and Nocedal 1989] for CVT optimization. *The source code of our framework and data will be publicly released after acceptance.*

5.3 Evaluation Metrics

Besides the qualitative visualization measurement, the quantitative evaluation for anisotropic mesh result includes: surface accuracy, mesh quality, and computational time.

5.3.1 Surface Accuracy. For the evaluation on surface mesh accuracy, we use Chamfer Distance (CD), F-Score (F1), normal consistency (NC), and Hausdorff Distance (HD). To evaluate the ability of preserving sharp features, following PoNQ [Maruani et al. 2024], we use Edge Chamfer Distance (ECD) and Edge F-score (EF1).

5.3.2 Mesh Quality. To measure the anisotropic mesh quality, for each triangle $\triangle_{abc}$ in the final mesh, we use its approximated metric $\mathbf{Q}(\triangle_{abc}) = (\mathbf{Q}(\mathbf{x}_a) + \mathbf{Q}(\mathbf{x}_b) + \mathbf{Q}(\mathbf{x}_c))/3$, where $\mathbf{Q}(\cdot) = \sqrt{\mathbf{M}(\cdot)}$, to affine-transform it from the original anisotropic space into the Euclidean space. After that, we employ the isotropic triangular criteria [Frey and Borouchaki 1999], to evaluate the quality of generated anisotropic triangular mesh, as suggested by [Fu et al. 2014; Zhong et al. 2013]. 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_{avg} is the average qualities of all triangles.

5.3.3 Timing. One of our main advantages is the efficiency of computing the high-d embedding. In Tab. 1, we report the inference time T_{em} of the neural high-d embedding and computational time of mesh T_{me} on high-d normal metric CVT or high-d CVT, for the Thingi10k testing set. Furthermore, we report the inference time of our method with respect to the number of input mesh vertices for all three testing datasets in Section F.1 of Supplemental Document. The timings of our method and SIFHDE^2 [Zhong et al. 2018] are collected from 3.8 GHz AMD Ryzen 3960x processor and an NVIDIA GeForce RTX 3090 GPU with 24GB GDDR6X.

5.4 Results on Surfaces without Sharp Features

We first evaluate our NASM method on the testing set with 280 mesh models selected from Thingi10k as shown in Fig. 4, and compare the result to SIFHDE^2 [Zhong et al. 2018]. The original version of SIFHDE^2 heavily relies on the smoothness of input metric field, and we conduct the comparison experiments on our improved version (in Section A of Supplemental Document), for both the surface accuracy and anisotropic mesh quality measurement. The improved version of SIFHDE^2 takes surface mesh and its corresponding metric field as input. We use the same process to generate high-d embedding for our dataset preparation (in Section 5.1). In their paper [Zhong et al. 2018], they used high-d version of particle-based method [Zhong et al. 2013]. However, the definition for inter-particle energy and force are defined on Euclidean distance, not on the surface manifold. For this reason, we leverage the high-d CVT from the meshing process, which uses RVD during the optimization. It can better to approximate the geodesic distance. The quantitative results between our method and SIFHDE^2 are reported in Tab. 1. Our method outperforms SIFHDE^2 in both surface accuracy, anisotropic mesh quality, as well as computational time with about 1,500× speedup. Fig. 5 shows the qualitative comparison with SIFHDE^2 method. Our method can better represent the local geometry changes, such as the zoom-in regions for selected models from Thingi10k dataset.

Table 1. Quantitative comparison with our NASM, NASM w/o high-d normal metric CVT, and SIFHDE^2 method [Zhong et al. 2018] on 80 models selected from Thingi10k dataset, including surfaces without and with sharp and weak features. All the evaluation metrics are average values of all models from the dataset. The best results are highlighted in bold per different $\#V_{out}$. Note: $\#V_{in}$ and $\#V_{out}$ are the average numbers of vertices of all input and output meshes, 'Stretch' is the average anisotropic stretching ratios of all models, CD (×10^5), HD (×10^2), ECD (×10^2), T_{em} (s), T_{me} (s).

Method	$\#V_{in}$	$\#V_{out}$	Stretch	CD ↓	F1 ↑	NC ↑	HD ↓	ECD ↓	EF1 ↑	T_{em} ↓	G_{avg} ↑	T_{me} ↓
NASM	5,982	5,982	12.736	**0.709**	**0.978**	**0.993**	**0.725**	**0.066**	**0.897**	**0.029**	0.745	14.022
	5,982	3,590	12.736	**0.720**	**0.978**	**0.991**	**0.779**	**0.086**	**0.850**	**0.029**	0.748	7.366
	5,982	1,202	12.736	**0.882**	**0.967**	**0.984**	**1.127**	**0.148**	**0.676**	**0.029**	0.744	3.499
	5,982	608	12.736	**1.538**	**0.928**	**0.974**	**1.774**	**0.207**	**0.501**	**0.029**	0.732	**2.092**
NASM w/o NM CVT (w/ CVT)	5,982	5,982	12.736	0.779	0.972	0.989	0.882	0.137	0.687	**0.029**	**0.758**	3.780
	5,982	3,590	12.736	0.875	0.963	0.987	1.055	0.155	0.571	**0.029**	**0.764**	**3.354**
	5,982	1,202	12.736	1.684	0.905	0.975	1.613	0.215	0.290	**0.029**	**0.776**	3.127
	5,982	608	12.736	3.692	0.779	0.962	2.352	0.267	0.170	**0.029**	**0.779**	2.922
SIFHDE^2	5,982	5,982	12.736	0.808	0.969	0.988	0.949	0.146	0.612	49.25	0.729	**3.718**
	5,982	3,590	12.736	0.928	0.959	0.985	1.145	0.166	0.487	49.25	0.732	3.441
	5,982	1,202	12.736	1.878	0.893	0.975	1.848	0.222	0.249	49.25	0.734	**3.005**
	5,982	608	12.736	4.144	0.766	0.963	2.674	0.270	0.157	49.25	0.730	2.935

Table 2. Quantitative comparison with our NASM and LCT method [Fu et al. 2014] on Fertility and Rocker Arm models. The best results are highlighted in bold. Note: CD (×10^5) and HD (×10^2). Note: *Due to lacking their curvature metrics, LCT's G_{avg} values are copied from their original paper.

Model	Method	$\#V_{out}$	CD ↓	F1 ↑	NC↑	HD ↓	G_{avg} ↑
Rocker Arm	LCT	5,550	0.625	0.995	0.994	0.597	**0.86***
	NASM	5,547	**0.600**	**0.996**	**0.996**	**0.576**	0.722
Fertility	LCT	12,480	0.584	**0.996**	**0.996**	0.603	**0.89***
	NASM	12,475	**0.578**	**0.996**	**0.996**	**0.596**	0.712

We show further comparative analysis and experiments with another state-of-the-art anisotropic surface meshing approach, Locally Convex Triangulation (LCT) [Fu et al. 2014] on Fertility and Rocker Arm models. From Fig. 6, in highlighted regions, it is clear to see that our method can capture the local curvatures more accurately. However, the stretching ratios of the mesh elements in LCT result are either too big or too small on some regions, since their method highly depends on the specific input curvature metric. The curvature metric computation is not stable and variant due to the different mesh discretizations. In Tab. 2, it shows that our method has better surface accuracy on CD, F1, NC, HD metrics. LCT is a bit better than ours on mesh quality G_{avg}. Due to lacking their curvature metrics, LCT's G_{avg} values are copied from the original paper.

5.5 Results on Surfaces with Sharp Features

Another advantage of our NASM method is to automatically keep sharp features from the input mesh without any user's intervention as shown in Fig. 4. To show the robustness of our ability of feature preserving, we conduct the experiments on different level of resolutions for output meshes and compare the results with SIFHDE^2. We use 100%, 60%, 20%, and 10% of vertex count from input meshes and make the quantitative report in Tab. 1 (a full version is included in Section E.1 of Supplemental Document). Our NASM outperforms

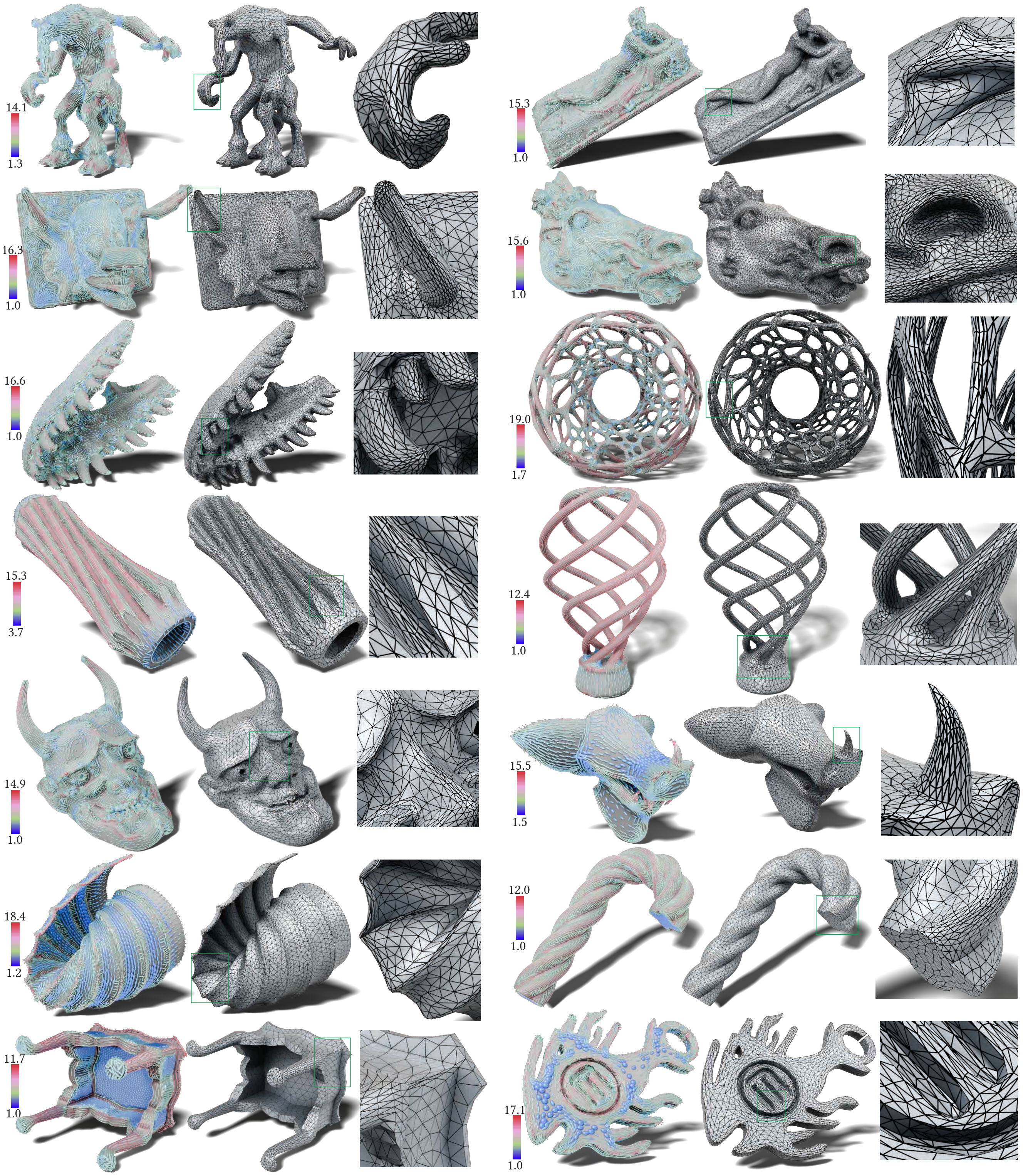


Fig. 4. Our anisotropic surface meshing results on smooth surfaces (top three rows) and surfaces with sharp or weak features (bottom four rows). (left to right: curvature tensors with corresponding stretching ratios denoted in colors, anisotropic meshing, and a zoom-in illustration). The files' IDs are provided from Thingi10k dataset. Left column: 133077, 76778, 46461, 741525, 81589, 87688, 51015; Right column: 72870, 68380, 61258, 39086, 40992, 107910, 75989.

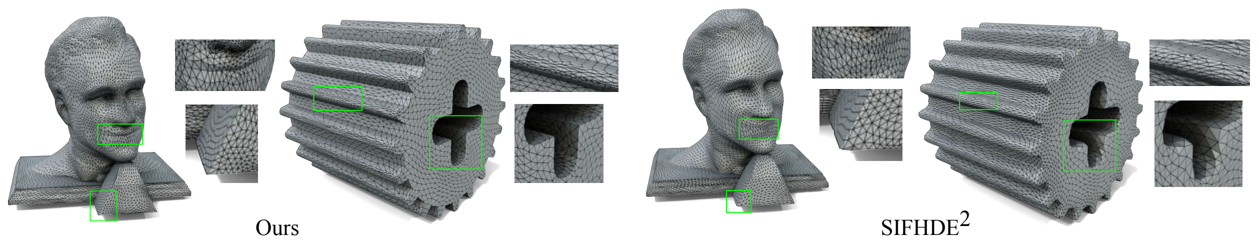


Fig. 5. Comparison between our method and SIFHDE^2 [Zhong et al. 2018] on models from Thingi10k dataset with the same number of vertices.

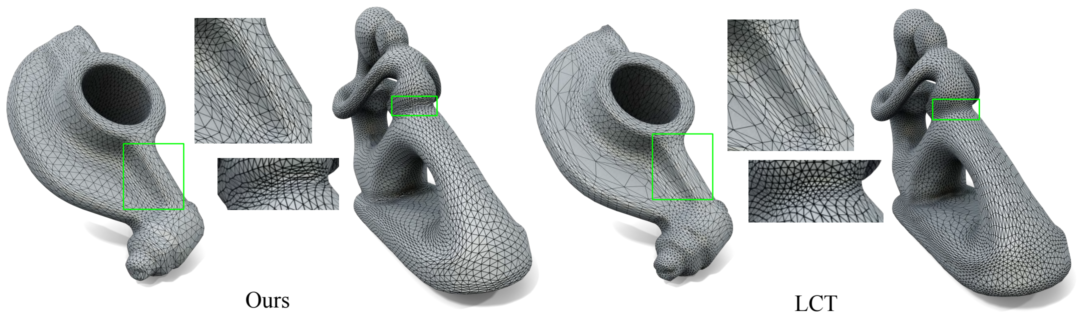


Fig. 6. Comparison between our method and LCT method [Fu et al. 2014] on Rocker Arm and Fertility models with the same number of vertices.

SIFHDE^2 at all levels of resolution. It obtains better surface accuracy and anisotropic quality than SIFHDE^2. Fig. 7 shows one example on different resolutions for anisotropic meshes. This can be applied in the curvature-induced mesh simplifications.

Furthermore, we do an ablation study on normal metric CVT for our NASM by using a general CVT to generate the anisotropic mesh. The result is reported in Tab. 1. We observe an increasing of anisotropic mesh quality when using CVT for meshing. Based on our analysis, it is actually a trade-off between anisotropy mesh quality and surface accuracy (feature preserving). Normal metric CVT can push the vertices towards sharp feature, which may distort the anisotropic metric direction and stretching ratio. So that the mesh quality evaluation may become lower.

The qualitative comparison with NASM method, NASM without high-d normal metric CVT (with CVT), and SIFHDE^2 method is provided in Fig. 2 of Supplemental Document. Fig. 8 shows anisotropic RVD results on some complicated surfaces from the Thingi10k dataset, which are computed by our high-d normal metric CVT optimization. Some additional RVD results are shown in Fig. 2 of Supplemental Document due to the page limit.

5.6 Results on Unseen Datasets

To further demonstrate the robustness and extensibility of our method, we train our neural high-d embedding on Thingi10k dataset, and directly test it on 154 models (including 96 pants and 58 tops) from MGN dataset without fine-tuning the network parameters. It is noted that MGN models are quite different from the training mesh models. Fig. 9 shows that our results can well capture open boundaries, detailed anisotropies and features around cloth wrinkles and folds from pants models. More visualization results and quantitative evaluations are provided in Section E.2 of Supplemental Document.

We also test our method on part of FAUST dataset with 200 human scan models of 10 different subjects in 30 different poses without fine-tuning the network parameters. Each scan is a high-resolution and non-watertight triangular mesh. We use QEM [Garland and Heckbert 1997] to reduce the mesh resolution to feed into the neural network. Both qualitative and quantitative evaluations are provided in Section E.3 of Supplemental Document. Our results show promising results to capture geometric anisotropies and features around hands, arms, legs, and thin clothes wrinkles on 3D human models.

Table 3. Ablation study on the losses of our dot product, L2, Cos, and without data augmentation on 80 models from Thingi10k dataset. All the evaluation metrics are average values from the dataset. The best results are highlighted in bold. Note: CD ($\times 10^5$), HD ($\times 10^2$), ECD ($\times 10^2$).

Loss / Method	$\#V_{in}$	$\#V_{out}\downarrow$	CD $\downarrow$	F1 $\uparrow$	NC$\uparrow$	HD $\downarrow$	ECD $\downarrow$	EF1 $\uparrow$	$G_{avg}\uparrow$
Dot prod	5,982	5,982	**0.709**	**0.978**	**0.993**	0.725	**0.066**	**0.897**	**0.745**
L2	5,982	39,543	7.97×10^6	0.951	0.977	578.65	0.267	0.765	0.625
Cos	5,982	6,143	0.725	0.977	0.991	**0.703**	0.092	0.813	0.654
w/o aug	5,982	6,143	0.738	0.976	0.991	**0.703**	0.077	0.862	0.656

5.7 Ablation Study

We provide quantitative and qualitative ablation studies in Tab. 3 and Fig. 9 of Supplemental Document on different loss functions, and data augmentation. The ablation study is performed on the Thingi10k dataset. Compared with L2 loss and Cosine loss, our proposed dot product loss is most effective on the anisotropic surface meshing. Our loss outperforms other losses on all the surface accuracy and mesh quality metrics. Visually, it is noted that the dot product loss can better recover the curvature metrics as well as better capture the geometric features.

6 CONCLUSION

In this article, we develop a novel scalable anisotropic surface mesh generation method. To our knowledge, this is the first deep learning-based method capable of generating a large number of high-quality

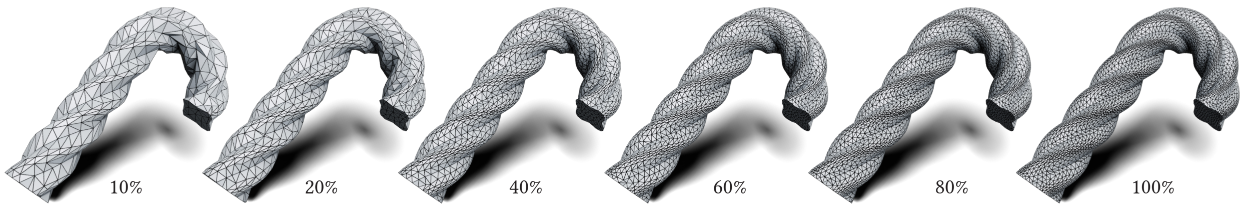


Fig. 7. Anisotropic surface meshing results of our NASM method on different number of output vertices (from 10% to 100% of the user specified vertex numbers, i.e., 555, 1103, 2207, 3311, 4415, 5518). It is clear to see that even 10% of vertices, our method can still well capture both the curvature metrics and features.

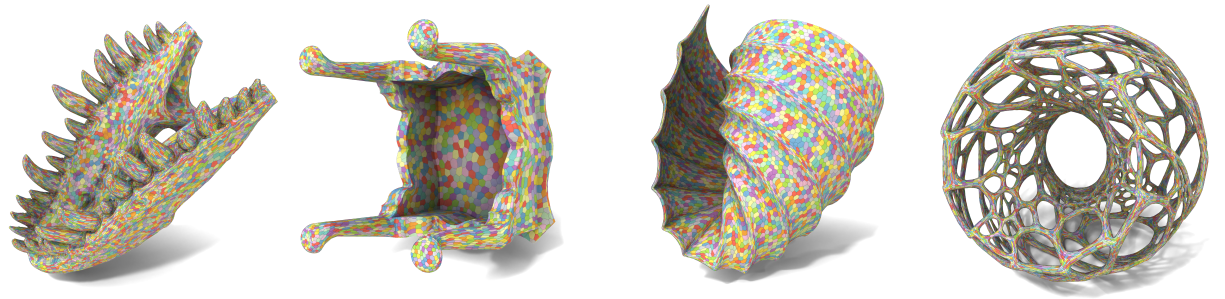


Fig. 8. Anisotropic RVD results on some complicated surfaces from the Thingi10k dataset, which are computed by our high-d normal metric CVT optimization. The files' IDs are provided from Thingi10k dataset from left to right: 46461, 51015, 87688, 61258.

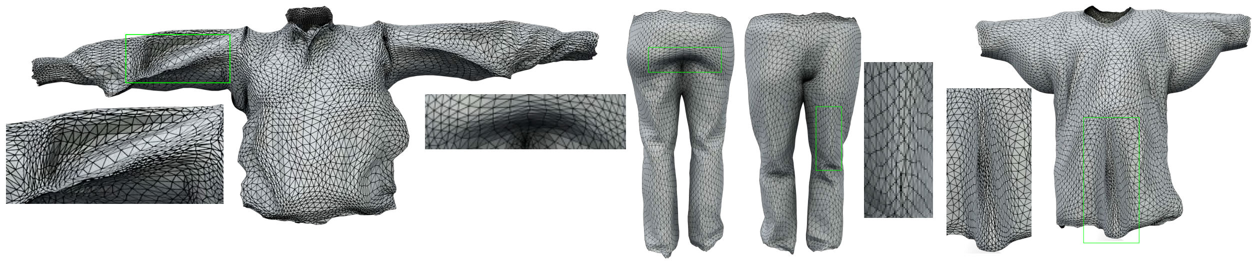


Fig. 9. Anisotropic surface meshing results of our NASM method on an unseen testing dataset, e.g., MGN dataset. The examples of our results can well capture the open boundaries and detailed cloth wrinkles and folds from tops and pants models.

and high-fidelity anisotropic surface meshes. There are several advantages of this method compared with traditional methods: (1) there is no curvature metric needed as input for our mesh generation. It can relieve the issues coming from the curvature estimation; (2) the high-d embedding computational time has been dramatically reduced to real time with the help of the designed learning-based method; (3) the developed high-d normal metric CVT formulation can generate feature-sensitive anisotropic meshes, which well capture both sharp and weak features; (4) this method is robust to generate a large number of anisotropic surface meshes from complicated geometric shapes.

7 LIMITATIONS AND FUTURE WORK

For some CAD-like models with very sparse input vertices and flat planes only, NASM may fail, since the high-quality embeddings are very challenging to be computed / predicted in such cases, which have been shown in Section F.3 of Supplemental Document. Another limitation is about the generalization to other anisotropic metrics fields. This requires the preparation of such training datasets for those particular applications. In the future, we will extend the current framework to deal with large-scale 3D scene mesh generation. It is also promising to explore how to effectively integrate the high-d normal metric CVT computation into the neural network computing framework. Since curvature-induced anisotropic meshes are essential for enhancing accuracy, efficiency, and stability in simulations involving complex geometries across various fields, we will apply our method in fluid dynamics, computer animation, and medical simulations, etc.

ACKNOWLEDGMENTS

The authors would like to thank the anonymous reviewers for their valuable comments and suggestions. Hongbo Li, Haikuan Zhu, Sikai Zhong, Jing Hua, and Zichun Zhong were partially supported by National Science Foundation (OAC-1845962, OAC-1910469, and OAC-2311245). Ningna Wang and Xiaohu Guo were partially supported by National Science Foundation (OAC-2007661). Shiqing Xin was partially supported by National Key R&D Program of China (2021YFB1715900).

REFERENCES

Frédéric Alauzet and Adrien Loseille. 2010. High-Order Sonic Boom Modeling Based on Adaptive Methods. *J. Comput. Phys.* 229, 3 (2010), 561–593.

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

Patrick R Amestoy, Iain S Duff, Jean-Yves L'Excellent, and Jacko Koster. 2001. A fully asynchronous multifrontal solver using distributed dynamic scheduling. *SIAM J. Matrix Anal. Appl.* 23, 1 (2001), 15–41.

Bharat Lal Bhatnagar, Garvita Tiwari, Christian Theobalt, and Gerard Pons-Moll. 2019. Multi-Garment Net: Learning to Dress 3D People from Images. In *IEEE International Conference on Computer Vision.*

Federica Bogo, Javier Romero, Matthew Loper, and Michael J Black. 2014. FAUST: Dataset and Evaluation for 3D Mesh Registration. In *Proceedings of the IEEE conference on computer vision and pattern recognition.* 3794–3801.

Jean-Daniel Boissonnat, Kan-Le Shi, Jane Tournois, and Mariette Yvinec. 2015a. Anisotropic Delaunay Meshes of Surfaces. *ACM Transactions on Graphics* 34, 2 (2015).

Jean-Daniel Boissonnat, Camille Wormser, and Mariette Yvinec. 2015b. Anisotropic Delaunay Mesh Generation. *SIAM J. Comput.* 44, 2 (2015), 467–512.

Frank J Bossen and Paul S Heckbert. 1996. A Pliant Method for Anisotropic Mesh Generation. 63 (1996), 76.

Michael M. Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. 2017. Geometric Deep Learning: Going beyond Euclidean data. *IEEE Signal Processing Magazine* 34, 4 (July 2017), 18–42.

Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann Lecun. 2014. Spectral Networks and Locally Connected Networks on Graphs. (2014).

Max Budninskiy, Beibei Liu, Fernando De Goes, Yiying Tong, Pierre Alliez, and Mathieu Desbrun. 2016. Optimal Voronoi tessellations with Hessian-based anisotropy. *ACM Transactions on Graphics* 35, 6 (2016), 1–12.

Bob Carpenter, Matthew D. Hoffman, Marcus Brubaker, Daniel Lee, Peter Li, and Michael Betancourt. 2015. The Stan Math Library: Reverse-Mode Automatic Differentiation in C++. arXiv:1509.07164

Long Chen, Pengtao Sun, and Jinchao Xu. 2007. Optimal Anisotropic Meshes for Minimizing Interpolation Errors in Lp-norm. *Math. Comp.* 76, 257 (2007), 179–204.

Long Chen and Jin-chao Xu. 2004. Optimal Delaunay Triangulations. *Journal of Computational Mathematics* (2004), 299–308.

Zhiqin Chen, Andrea Tagliasacchi, and Hao Zhang. 2021. Learning Mesh Representations via Binary Space Partitioning Tree Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2021), 1–1.

Franco Dassi, Andrea Mola, and Hang Si. 2014. Curvature-Adapted Remeshing of CAD Surfaces. *Procedia Engineering* 82 (2014), 253–265.

Franco Dassi, Hang Si, Simona Perotto, and Timo Streckenbach. 2015. Anisotropic Finite Element Mesh Adaptation via Higher Dimensional Embedding. *Procedia Engineering* 124 (2015), 265–277.

Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering. (2016).

Cécile Dobrzynski and Pascal Frey. 2008. Anisotropic Delaunay Mesh Adaptation for Unsteady Simulations. In *Proceedings of the 17th International Meshing Roundtable.* 177–194.

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

Matthias Fey and Jan E. Lenssen. 2019. Fast Graph Representation Learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds.*

Pascal J Frey and Houman Borouchaki. 1999. Surface Mesh Quality Evaluation. *International journal for numerical methods in engineering* 45, 1 (1999), 101–118.

Xiao-Ming Fu, Yang Liu, John Snyder, and Baining Guo. 2014. Anisotropic Simplicial Meshing Using Local Convex Functions. *ACM Transactions on Graphics* 33, 6 (2014), 1–11.

Hongyang Gao and Shuiwang Ji. 2019. Graph u-nets. In *international conference on machine learning.* PMLR, 2083–2092.

Michael Garland and Paul S Heckbert. 1997. Surface Simplification Using Quadric Error Metrics. In *Proceedings of the Annual Conference on Computer Graphics and Interactive Techniques.* 209–216.

Gaël Guennebaud, Benoît Jacob, et al. 2010. Eigen v3. http://eigen.tuxfamily.org.

William L. Hamilton, Rex Ying, and Jure Leskovec. 2018. Inductive Representation Learning on Large Graphs.

Rana Hanocka, Amir Hertz, Noa Fish, Raja Giryes, Shachar Fleishman, and Daniel Cohen-Or. 2019. MeshCNN: A Network with an Edge. *ACM Transactions on Graphics* 38, 4 (2019), 90:1–90:12.

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

Shi-Min Hu, Zheng-Ning Liu, Meng-Hao Guo, Junxiong Cai, Jiahui Huang, Tai-Jiang Mu, and Ralph R. Martin. 2022. Subdivision-based Mesh Convolution Networks. *ACM Transactions on Graphics* (2022).

Yixin Hu, Qingnan Zhou, Xifeng Gao, Alec Jacobson, Denis Zorin, and Daniele Panozzo. 2018. Tetrahedral Meshing in the Wild. *ACM Transactions on Graphics* 37, 4 (2018), 60:1–60:14.

Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning.* 448–456.

Ioannis Ivrissimtzis, Won-Ki Jeong, Seungyong Lee, Yunjin Lee, and Hans-Peter Seidel. 2004. Neural Meshes: Surface Reconstruction with a Learning Algorithm. (2004).

Alec Jacobson, Daniele Panozzo, et al. 2018. libigl: A simple C++ geometry processing library. https://libigl.github.io/.

Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks.

Ilya Kostrikov, Zhongshi Jiang, Daniele Panozzo, Denis Zorin, and Burna Joan. 2018. Surface Networks. (2018).

Nicolaas H Kuiper. 1955. On C^1-isometric embeddings I. In *Proc. Nederl. Akad. Wetensch. Ser. A.* 545–556.

Bruno Lévy. 2015. Geogram. *GitHub Repository. URL: https://github.com/BrunoLevy/geogram* (2015).

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

Bruno Lévy and Nicolas Bonneel. 2013. Variational Anisotropic Surface Meshing with Voronoi Parallel Linear Enumeration. In *Proceedings of the 21st International Meshing Roundtable.* 349–366.

Bruno Lévy and Yang Liu. 2010. Lp Centroidal Voronoi Tessellation and Its Applications. *ACM Transactions on Graphics* 29, 4 (2010).

Dong C Liu and Jorge Nocedal. 1989. On the limited memory BFGS method for large scale optimization. *Mathematical programming* 45, 1 (1989), 503–528.

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

Ilya Loshchilov and Frank Hutter. 2019. Decoupled Weight Decay Regularization. In *International Conference on Learning Representations.*

Andrew L Maas, Awni Y Hannun, Andrew Y Ng, et al. 2013. Rectifier Nonlinearities Improve Neural Network Acoustic Models. In *Proc. icml*, Vol. 30. 3.

Nissim Maruani, Maks Ovsjanikov, Pierre Alliez, and Mathieu Desbrun. 2024. PoNQ: a Neural QEM-based Mesh Representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition.* 3647–3657.

Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. 2019. Occupancy Networks: Learning 3D Reconstruction in Function Space. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition.* 4460–4470.

Rahul Narain, Armin Samii, and James F O'brien. 2012. Adaptive Anisotropic Remeshing for Cloth Simulation. *ACM Transactions on Graphics* 31, 6 (2012), 147:1–147:10.

John Nash. 1954. C^1-Isometric Embeddings. *Annals of Mathematics* 60, 3 (1954), 383–396.

John Nash. 1956. The Imbedding Problem for Riemannian Manifolds. *Annals of mathematics* 63, 1 (1956), 20–63.

Bo Pang, Zhongtian Zheng, Guoping Wang, and Peng-Shuai Wang. 2023. Learning the Geodesic Embedding with Graph Neural Networks. *ACM Transactions on Graphics* 42, 6 (Dec. 2023), 1–12.

Daniele Panozzo, Enrico Puppo, Marco Tarini, and Olga Sorkine-Hornung. 2014. Frame Fields: Anisotropic and Non-Orthogonal Cross Fields. *ACM Transactions on Graphics* 33, 4 (2014), 134:1–134:11.

Tobias Pfaff, Meire Fortunato, Alvaro Sanchez-Gonzalez, and Peter W. Battaglia. 2020. Learning Mesh-Based Simulation with Graph Networks. *CoRR* abs/2010.03409 (2020).

Rolandos Alexandros Potamias, Stylianos Ploumpis, and Stefanos Zafeiriou. 2022. Neural Mesh Simplification. (2022), 18562–18571.

Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. 2017a. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition.* 652–660.

Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. 2017b. PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space. In *Advances in Neural Information Processing Systems*, I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Eds.), Vol. 30. Curran Associates, Inc.

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

Paul R Shapiro, Hugo Martel, Jens V Villumsen, and J Michael Owen. 1996. Adaptive Smoothed Particle Hydrodynamics, with Application to Cosmology: Methodology. *Astrophysical Journal Supplement v. 103, p. 269* 103 (1996), 269.

Nicholas Sharp, Souhaib Attaiki, Keenan Crane, and Maks Ovsjanikov. 2022. DiffusionNet: Discretization Agnostic Learning on Surfaces. *ACM Transactions on Graphics* (2022).

Kenji Shimada, Atsushi Yamada, Takayuki Itoh, et al. 1997. Anisotropic Triangular Meshing of Parametric Surfaces via Close Packing of Ellipsoidal Bubbles. In *6th International Meshing Roundtable*, Vol. 375. 390.

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

Dmitriy Smirnov and Justin Solomon. 2021. HodgeNet: Learning Spectral Geometry on Triangle Meshes. *ACM Transactions on Graphics*, Article 166 (jul 2021), 11 pages.

Robert W Sumner and Jovan Popović. 2004. Deformation Transfer for Triangle Meshes. *ACM Transactions on Graphics* 23, 3 (2004), 399–405.

Ivan E. Sutherland and Gary W. Hodgman. 1974. Reentrant polygon clipping. *Commun. ACM* 17, 1 (jan 1974), 32–42. https://doi.org/10.1145/360767.360802

Sébastien Valette, Jean Marc Chassery, and Rémy Prost. 2008. Generic Remeshing of 3D Triangular Meshes with Metric-Dependent Discrete Voronoi Diagrams. *IEEE Transactions on Visualization and Computer Graphics* 14, 2 (2008), 369–381.

Nanyang Wang, Yinda Zhang, Zhuwen Li, Yanwei Fu, Wei Liu, and Yu-Gang Jiang. 2018b. Pixel2Mesh: Generating 3D Mesh Models from Single RGB Images. *CoRR* abs/1804.01654 (2018).

Peng-Shuai Wang, Yang Liu, Yu-Xiao Guo, Chun-Yu Sun, and Xin Tong. 2017. O-CNN: Octree-based convolutional neural networks for 3D shape analysis. *ACM Transactions On Graphics* 36, 4 (2017), 1–11.

Peng-Shuai Wang, Chun-Yu Sun, Yang Liu, and Xin Tong. 2018a. Adaptive O-CNN: A patch-based deep representation of 3D shapes. *ACM Transactions on Graphics* 37, 6 (2018), 1–11.

Yun-Peng Xiao, Yu-Kun Lai, Fang-Lue Zhang, Chunpeng Li, and Lin Gao. 2020. A Survey on Deep Geometry Learning: from a Representation Perspective. *Computational Visual Media* 6 (2020), 113–133.

Rui Xu, Longdu Liu, Ningna Wang, Shuangmin Chen, Shiqing Xin, Xiaohu Guo, Zichun Zhong, Taku Komura, Wenping Wang, and Changhe Tu. 2024. CWF: Consolidating Weak Features in High-quality Mesh Simplification. *ACM Transactions on Graphics* 43, 4 (2024), 1–14.

Dong-Ming Yan, Bruno Lévy, Yang Liu, Feng Sun, and Wenping Wang. 2009. Isotropic Remeshing with Fast and Exact Computation of Restricted Voronoi Diagram. *Computer Graphics Forum* 28, 5 (2009), 1445–1454.

Zichun Zhong, Xiaohu Guo, Wenping Wang, Bruno Lévy, Feng Sun, Yang Liu, and Weihua Mao. 2013. Particle-Based Anisotropic Surface Meshing. *ACM Transactions on Graphics* 32, 4 (2013).

Zichun Zhong, Liang Shuai, Miao Jin, and Xiaohu Guo. 2014. Anisotropic Surface Meshing with Conformal Embedding. *Graphical models* 76, 5 (2014), 468–483.

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 Transactions on Graphics* 37, 4 (2018).

Qingnan Zhou, Eitan Grinspun, Denis Zorin, and Alec Jacobson. 2016. Mesh Arrangements for Solid Geometry. *ACM Transactions on Graphics* 35, 4 (2016).

NASM: Neural Anisotropic Surface Meshing: Supplemental Document

A DATA GENERATION FOR EMBEDDING AND MESHING

High-D Euclidean Embedding Computation. For an arbitrary metric field $\mathbf{M}$ defined on a 3D surface domain $\Omega \subset \mathbb{R}^3$, there exists a high-d Euclidean embedded surface $\overline{\Omega} \subset \mathbb{R}^{\overline{m}}$ [Zhong et al. 2018; ?], where the mapping $\Omega \longrightarrow \overline{\Omega}$ can be considered as a high-d transformation. Our computation process depends on the faithful high-d coordinates of the training meshes. To obtain that, we first follow the core idea of SIFHDE^2 [Zhong et al. 2018] for the surface mesh case, which constructs the corresponding simplices in the target high-d space by deforming the tangent basis of each triangle in the mesh. The main advantages of this high-d embedding method are that we can automatically preserve the original 3D shape geometric features (such as sharp feature edges and corners) as well as avoid embedded self-intersections by using the strategy of keeping the original 3D coordinates, and only embedding additionally higher dimensions, i.e., "anisotropic metric is traded as additional dimensions".

For a triangle $\mathcal{F} \in \mathbb{R}^3$, let $\{\mathbf{v}_i, \mathbf{v}_j, \mathbf{v}_k\}$ denote its vertices. The basis of its tangent space $\mathbf{W}_{\mathcal{F}}$ can be given by its edge vectors, $\mathbf{W}_{\mathcal{F}} = [\mathbf{v}_j - \mathbf{v}_i, \mathbf{v}_k - \mathbf{v}_i]$. The corresponding simplex in high-d space is $\mathcal{F} \in \mathbb{R}^{\overline{m}}$ where $\overline{m} \geq 3$ (as suggested in [Zhong et al. 2018], $\overline{m} = 8$), and the basis of tangent space for $\mathcal{F}$ can be denoted by $\overline{\mathbf{W}}_{\mathcal{F}} = [\overline{\mathbf{v}}_j - \overline{\mathbf{v}}_i, \overline{\mathbf{v}}_k - \overline{\mathbf{v}}_i]$. Their relation can be represented as:

$$\overline{\mathbf{W}}_{\mathcal{F}} = \mathbf{J}_{\mathcal{F}} \mathbf{W}_{\mathcal{F}}, \tag{12}$$

where $\mathbf{J}_{\mathcal{F}}$ is the Jacobian transformation matrix of triangle $\mathcal{F}$, and $\mathbf{J}^t_{\mathcal{F}} \mathbf{J}_{\mathcal{F}} = \mathbf{M}_{\mathcal{F}}$.

However, this process could produce incorrect anisotropic direction on some part of the mesh surface, which makes the whole mesh object *unstable* for training. The main reason is because $\mathbf{W}_{\mathcal{F}}$ is not a square matrix (3×2 as formed above), which cannot fully determine the deformation since the direction perpendicular to the triangle is not established as shown in Fig. 10. To address this issue, inspired by [Sumner and Popović 2004], we construct a fourth vertex hypothetically on each triangle $\mathcal{F}$ in the direction perpendicular to the triangle as $\mathbf{v}_l = \frac{(\mathbf{v}_j - \mathbf{v}_i) \times (\mathbf{v}_k - \mathbf{v}_i)}{\sqrt{||(\mathbf{v}_j - \mathbf{v}_i) \times (\mathbf{v}_k - \mathbf{v}_i)||}}$, and result in a new basis of tangent plane $\mathbf{W}'_{\mathcal{F}} = [\mathbf{v}_j - \mathbf{v}_i, \mathbf{v}_k - \mathbf{v}_i, \mathbf{v}_l - \mathbf{v}_i]$. As for the high-d corresponding basis, we can discard the new vertex from the deformation result and keep the same notation $\overline{\mathbf{W}}_{\mathcal{F}}$ as above, since we only consider the surface case. Through the above computations, the ground truth data for training our neural high-d Euclidean embedding is generated.

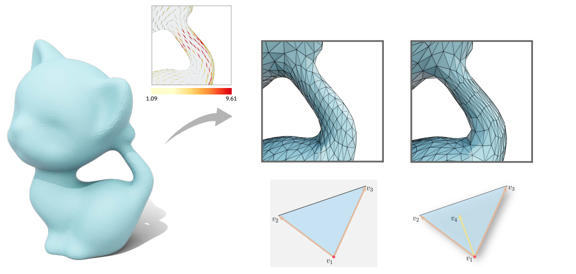


Fig. 10. Illustration of adding a fourth vertex on the direction perpendicular to a triangle.

Anisotropic Surface Mesh Generation. After obtaining the high-d embedded coordinates $\overline{\mathbf{W}}_{\mathcal{F}}$, we can efficiently compute high-d isotropic restricted Voronoi diagrams (RVD) of object $\overline{\Omega}$ using the particle optimization technique [Zhong et al. 2018] or centroidal Voronoi tessellation (CVT) method [Lévy and Bonneel 2013] in high-d space. Finally, the generated RVD and its dual restricted Delaunay triangulation can be mapped from the high-d embedding space to the original 3D space, where RVD and dual mesh can exhibit the desired anisotropy. In practice, we only need to get rid of the additional dimensions to get the back-mapped results since the embedding computation is based on the strategy of keeping the original 3D coordinates. Now, we can conduct the comparison experiments between the improved version of SIFHDE^2 and our NASM w.r.t. the surface accuracy and anisotropic mesh quality measurement.

B COMBINATORIAL STRUCTURE OF HIGH-D NORMAL METRIC CVT

We consider a domain $\overline{\Omega} \in \mathbb{R}^{\overline{m}}$, where $\overline{\Omega}$ is a surface embedded in $\mathbb{R}^{\overline{m}}$. The combinatorial structure of E_{hd} is determined by the high-d restricted Voronoi diagram (RVD), i.e., the intersection between the high-d Voronoi cells and the high-d embedded surface $\overline{\mathcal{S}}$. The RVD is first computed by an exact algorithm based on [Yan et al. 2009]. Then, each restricted Voronoi cell $\overline{\Omega}_{\mathbf{x}_0} \cap \overline{\mathcal{S}}$ is decomposed into a set of triangles called facets. The vertices of these facet triangles can have three different configurations and need to be treated differently in order to have the correct gradient when doing reverse-mode differentiation as follows:

- C$_1$: $\overline{\mathbf{V}}$ is a vertex of the original high-d embedded surface $\overline{\mathcal{S}}$;
- C$_2$: $\overline{\mathbf{V}}$ is the intersection between an edge of the original high-d embedded surface $\overline{\mathcal{S}}$ and a bisector of two Voronoi cell sites $\overline{\mathbf{x}}_i$, $\overline{\mathbf{x}}_j$;
- C$_3$: $\overline{\mathbf{V}}$ is the intersection between a triangle of the original high-d embedded surface $\overline{\mathcal{S}}$ and two sets of bisectors $\overline{\mathbf{x}}_i$, $\overline{\mathbf{x}}_j$ and $\overline{\mathbf{x}}_i$, $\overline{\mathbf{x}}_k$.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SA Conference Papers '24, December 3–6, 2024, Tokyo, Japan
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-1131-2/24/12.
https://doi.org/10.1145/3680528.3687700

C ALGEBRAIC STRUCTURE OF HIGH-D NORMAL METRIC CVT

The objective function of feature-sensitive high-d CVT is defined by adding a high-d normal metric $\overline{\mathbf{M}}^{\mathcal{T}}$ into the CVT energy in high-d:

$$E_{hd}(\overline{\mathbf{X}}) = \sum_i \int_{\overline{\Omega}_i \cap \mathcal{S}} \left\| \overline{\mathbf{M}}^{\mathcal{T}} [\overline{\mathbf{y}} - \overline{\mathbf{x}}_i] \right\|_2^2 d\overline{\mathbf{y}}, \tag{13}$$

$$= \frac{|\mathcal{T}|}{\binom{n+p}{n}} \sum_{\alpha+\beta+\gamma=p} \overline{\mathbf{U}_1^{\alpha} * \mathbf{U}_2^{\beta} * \mathbf{U}_3^{\gamma}}, \tag{14}$$

$$\text{where}: \begin{cases} \mathbf{U}_i & = \overline{\mathbf{M}}^{\mathcal{T}} (\mathbf{C}_i - \overline{\mathbf{x}}_0) \\ \overline{\mathbf{V}}_1 * \overline{\mathbf{V}}_2 & = [\overline{x}_1\overline{x}_2, \overline{y}_1\overline{y}_2, \overline{z}_1\overline{z}_2]^t \\ \overline{\mathbf{V}}^{\alpha} & = \overline{\mathbf{V}} * \overline{\mathbf{V}} * \cdots * \overline{\mathbf{V}} \ (\alpha \text{ times}) \\ \overline{\mathbf{V}} & = \overline{x} + \overline{y} + \overline{z} \end{cases}.$$

For RVD on the high-d embedded surface mesh, Voronoi cells are discretized to a set of facet triangles, denoted as $\mathcal{T}(\overline{\mathbf{x}}_0, \mathbf{C}_1, \mathbf{C}_2, \mathbf{C}_3)$. So the $\overline{\Omega}$ is the sum of the area of all the facets. A high-d CVT is a stable critical point of E_{hd} during the optimization.

As stated in [Lévy and Liu 2010], given the gradient of a standard CVT energy E, $\nabla E = 2m_i(\mathbf{x}_i - \mathbf{g}_i)$, one cannot simply replace the centroid $\mathbf{g}_i$ and mass m_i with their anisotropic counterparts to get the gradient of high-d CVT energy E_{hd}, since the anisotropy varies between two adjacent cells. The closed form for gradient of high-d CVT with normal metric over an integration simplex $\mathcal{T}$ is:

$$\frac{dE_{hd}^{\mathcal{T}}(\overline{\mathbf{x}}_0, \mathbf{C}_1, \mathbf{C}_2, \mathbf{C}_3)}{d\overline{\mathbf{X}}} = \frac{dE_{hd}^{\mathcal{T}}}{d\overline{\mathbf{x}}_0} + \sum_{i=1,2,3} \frac{dE_{hd}^{\mathcal{T}}}{d\mathbf{C}_i} \frac{d\mathbf{C}_i}{d\overline{\mathbf{X}}}, \tag{15}$$

$$\text{where}: \frac{dE_{hd}^{\mathcal{T}}}{d\overline{\mathbf{x}}_0} = -\frac{dE_{hd}^{\mathcal{T}}}{d\mathbf{C}_1} - \frac{dE_{hd}^{\mathcal{T}}}{d\mathbf{C}_2} - \frac{dE_{hd}^{\mathcal{T}}}{d\mathbf{C}_3}, \tag{16}$$

where $\overline{\mathbf{X}}$ is the site point of the Voronoi cell. $\mathcal{T}(\mathbf{C}_1, \mathbf{C}_2, \mathbf{C}_3)$ is one of the facet triangles that compose the restricted Voronoi cell in the high-d embedding space.

D AUTO DIFFERENTIATION FOR HIGH-D NORMAL METRIC CVT

Automatic differentiation is a computational technique that leverages the chain rule of calculus. Instead of computing gradients from explicit formula, automatic differentiation computes gradient starting from the function value, and tracing backwards along how the function value has been computed; and leverages the chain rule to compute the gradient.

From Equation (15), the calculation of gradient can be separated to two parts: $\frac{dE_{hd}^{\mathcal{T}}}{d\mathbf{C}_i}$ and $\frac{d\mathbf{C}_i}{d\overline{\mathbf{X}}}$, and the total gradient can be assembled from these two expressions. In this subsection, we introduce how to calculate the $E_{hd}^{\mathcal{T}}$ and $\mathbf{C}_i$ in the forward pass in order to get the correct gradient $\frac{dE_{hd}^{\mathcal{T}}}{d\mathbf{C}_i}$ and $\frac{d\mathbf{C}_i}{d\overline{\mathbf{X}}}$ in the reverse pass.

For $E_{hd}^{\mathcal{T}}$, it can be discretized onto triangle facets that compose the restricted Voronoi cell, and the expression for this discretization can be shortly expressed as:

$$E_{hd}^{\mathcal{T}} = |\mathcal{T}| F_{hd}^{\mathcal{T}}, \tag{17}$$

$$\text{where}: F_{hd}^{\mathcal{T}} = \sum_{\alpha+\beta+\gamma=p} \overline{\mathbf{U}_1^{\alpha} * \mathbf{U}_2^{\beta} * \mathbf{U}_3^{\gamma}},$$

where $|\mathcal{T}|$ denotes the area of current triangle facet, see Appendix A in [Lévy and Liu 2010] for details.

D.1 Derivative of $E_{hd}^{\mathcal{T}}$

In [Lévy and Liu 2010], they used $|\mathcal{T}| = \frac{1}{2}\|N\|$ to calculate the area of $|\mathcal{T}|$, where $\|N\|$ is the length of cross product between two edges of $|\mathcal{T}|$. It is fine for 3D case, and they also derive the explicit gradient expression for $d|\mathcal{T}|$. However, this way of calculating triangle area cannot be extended to dimension higher than 3. As we targeting on higher dimension (e.g., $d = 8$), we leverage Heron's formula for the area of a triangle in $\mathbb{R}^d$:

$$|\mathcal{T}| = \sqrt{s(s-a)(s-b)(s-c)}, \tag{18}$$

where $s = a + b + c$ and a, b, c denote the length of three edges under the normal metric $\overline{\mathbf{M}}^{\mathcal{T}}$.

It is noted that our computation is more complicated and challenging than [Lévy and Bonneel 2013], which also uses Heron's formula to calculate the area of triangle in 6D, but without having the metric in their CVT energy function. So, their 6D CVT optimization does not require to compute the gradient of Heron's formula. Conversely, for our high-d normal metric CVT, $|\mathcal{T}|$ becomes dependent regards to E_{hd}^{T}:

$$\frac{dE_{hd}^{\mathcal{T}}}{d\mathbf{C}_i} = \left(\frac{dF_{hd}^{\mathcal{T}}}{d\mathbf{U}_i}|\mathcal{T}| + \frac{d|\mathcal{T}|}{d\mathbf{U}_i} F_{hd}^{\mathcal{T}}\right)\overline{\mathbf{M}}^{\mathcal{T}}, \tag{19}$$

where $\mathbf{U}_i$ is defined in Equation (13).

To use automatic differentiation on $\frac{dE_{hd}^{\mathcal{T}}}{d\mathbf{C}_i}$, we first set $\mathbf{U}_1, \mathbf{U}_2, \mathbf{U}_3$ as dependent variables, and $F_{hd}^{\mathcal{T}}$ can be calculated from Equation (13). Then $\frac{dF_{hd}^{\mathcal{T}}}{d\mathbf{U}_i}$ can be derived by backpropagation. For automatic differentiation on $\frac{d|\mathcal{T}|}{d\mathbf{U}_i}$, we can reuse the $\mathbf{U}_1, \mathbf{U}_2, \mathbf{U}_3$ to get $\mathbf{a} = \|\mathbf{U}_1 - \mathbf{U}_2\|$, $\mathbf{b} = \|\mathbf{U}_2 - \mathbf{U}_3\|$, $\mathbf{c} = \|\mathbf{U}_3 - \mathbf{U}_1\|$ and use Equation (18) to calculate $|\mathcal{T}|$. Then we perform backpropagation to get $\frac{d|\mathcal{T}|}{d\mathbf{U}_i}$. Finally, $\frac{dE_{hd}^{\mathcal{T}}}{d\mathbf{C}_i}$ can be obtained by assembling all the values together following Equation (19).

The reason why we do not start the backpropagation from $E_{hd}^{\mathcal{T}} = |\mathcal{T}|F_{hd}^{\mathcal{T}}$ to derive $\frac{dE_{hd}^{\mathcal{T}}}{d\mathbf{C}_i}$ only, instead we take one step backward, is because the gradients for each $\mathbf{C}_i$ between these two ways could have some differences. So, our auto differentiation for the backpropagation makes the optimization convergence faster and more accurate. The detailed computational procedure for the derivative of $\mathbf{C}_i$ is provided in the following subsection.

D.2 Derivative of $\mathbf{C}_i$

The partial derivative of $\frac{dE_{hd}^{\mathcal{T}}(\overline{\mathbf{x}}_0, \mathbf{C}_1, \mathbf{C}_2, \mathbf{C}_3)}{d\overline{\mathbf{X}}}$ also requires the partial gradient from vertices of facet triangles. For 3D case, every facet triangle f can be described by their support plane $(\mathbf{N}_f, \mathbf{b}_f)$ of an equation $\mathbf{N}_f\mathbf{x} + \mathbf{b}_f = 0$. The vertices of the facet triangles can be expressed by the intersections of bisectors and support planes according to its configuration type (i.e., $\mathbf{C}_1, \mathbf{C}_2, \mathbf{C}_3$) and $d\mathbf{C}_i$ can be obtained by taking derivative with respect to a ternary linear system. However, for a facet triangle embedded in the high-d space, its normal vector $\mathbf{N}_f \in \mathbb{R}^d$ cannot be easily obtained. We use Sutherland-Hodgman's re-entrant clipping [Sutherland and Hodgman 1974]. We first recall the re-entrant clipping algorithm. Given the bisector $[\mathbf{x}_0, \mathbf{x}_1]$ and an edge with two endpoints $\mathbf{A}$ and $\mathbf{B}$ that intersects the bisector, and the intersection $\mathbf{I}$ can be obtained by:

$$\mathbf{I} = \lambda_1 \mathbf{A} + \lambda_2 \mathbf{B}, \tag{20}$$

$$\text{where} : \begin{cases} \lambda_1 = \frac{l_1}{|\mathbf{AB}|} \\ \lambda_2 = \frac{l_2}{|\mathbf{AB}|} \end{cases},$$

where l_1 is the perpendicular distance from $\mathbf{A}$ to bisector $[\overline{\mathbf{x}}_0, \overline{\mathbf{x}}_1]$, l_2 is the perpendicular distance from $\mathbf{B}$ to bisector $[\overline{\mathbf{x}}_0, \overline{\mathbf{x}}_1]$. To calculate the perpendicular distance, we can calculate the distance between two parallel planes. The bisector can be defined as $d = -\frac{1}{2}(\overline{\mathbf{x}}_0 + \overline{\mathbf{x}}_1) \cdot \overrightarrow{\mathbf{n}}$, and $\overrightarrow{\mathbf{n}}$ is the direction of $\overrightarrow{\overline{\mathbf{x}}_0\overline{\mathbf{x}}_1}$. The plane that passes through endpoint $\mathbf{A}$ and is parallel to the bisector is $d_1 = -\mathbf{A} \cdot \overrightarrow{\mathbf{n}}$. l_1 and l_2 can be obtained by:

$$l_i = |d_i - d|. \tag{21}$$

For three types of configurations for $\mathbf{C}_i$, where $i = 1, 2, 3$, we should use the re-entrant clipping in different ways to ensure the gradient achieved from backpropagation is correct. For configuration $\mathbf{C}_1$, i.e., the original vertex of the surface S, it yields no derivative for site points $\overline{\mathbf{x}}_i$. For configuration $\mathbf{C}_2$, it can be seen as an original edge from the surface S intersects with a bisector $[\overline{\mathbf{x}}_0, \overline{\mathbf{x}}_1]$, which is simply one pass of re-entrant clipping. We set $\overline{\mathbf{x}}_0$ and $\overline{\mathbf{x}}_1$ as dependent variables, two endpoints of the edges e_1 and e_2 as independent variables to get the intersection $\mathbf{C}_2$. Then, we can apply backpropagation from $\mathbf{C}_2$, and have the gradients of $\frac{d\mathbf{C}_2}{d\overline{\mathbf{x}}_0}$ and $\frac{d\mathbf{C}_2}{d\overline{\mathbf{x}}_1}$.

For configuration $\mathbf{C}_3$, the edges to be clipped with are not the original edges from surface S. The facet triangle is clipped by two bisectors $\mathbf{b}_1 = [\overline{\mathbf{x}}_0, \overline{\mathbf{x}}_1]$ and $\mathbf{b}_2 = [\overline{\mathbf{x}}_0, \overline{\mathbf{x}}_2]$. Each bisector intersects with a facet triangle at its two edges, and the intersections are labeled as $\mathbf{I}_1^{b_1}, \mathbf{I}_2^{b_1}$ and $\mathbf{I}_1^{b_2}, \mathbf{I}_2^{b_2}$. The intersection between $[\mathbf{I}_1^{b_1}, \mathbf{I}_2^{b_1}]$ and $[\mathbf{I}_1^{b_2}, \mathbf{I}_2^{b_2}]$ coincides with $\mathbf{C}_3$. We can treat the process as re-entrant clipping that the bisector $\mathbf{b}_1 = [\overline{\mathbf{x}}_0, \overline{\mathbf{x}}_1]$ intersects with edge $[\mathbf{I}_1^{b_2}, \mathbf{I}_2^{b_2}]$, and the gradient of $(\frac{d\mathbf{C}_3}{d\overline{\mathbf{x}}_0})^{b_1}$ and $\frac{d\mathbf{C}_3}{d\overline{\mathbf{x}}_1}$ can be obtained by backpropagation starting from $\mathbf{C}_3$. The same process should be applied to bisector $\mathbf{b}_2 = [\overline{\mathbf{x}}_0, \overline{\mathbf{x}}_2]$ and edge $[\mathbf{I}_1^{b_1}, \mathbf{I}_2^{b_1}]$, and we can obtain the gradient of $(\frac{d\mathbf{C}_3}{d\overline{\mathbf{x}}_0})^{b_2}$ and $\frac{d\mathbf{C}_3}{d\overline{\mathbf{x}}_2}$. The final gradients for $\overline{\mathbf{x}}_0$, $\overline{\mathbf{x}}_1$, and $\overline{\mathbf{x}}_2$ are $(\frac{d\mathbf{C}_3}{d\overline{\mathbf{x}}_0})^{b_1} + (\frac{d\mathbf{C}_3}{d\overline{\mathbf{x}}_0})^{b_2}$, $\frac{d\mathbf{C}_3}{d\overline{\mathbf{x}}_1}$, and $\frac{d\mathbf{C}_3}{d\overline{\mathbf{x}}_2}$, respectively.

E ADDITIONAL RESULTS

E.1 Thingi10k Dataset

Tab. 4 shows a full version of quantitative comparison with our NASM method, NASM without high-d normal metric CVT, and SIFHDE^2 method [Zhong et al. 2018]. We use 100%, 80%, 60%, 40%, 20%, and 10% of vertex count from input meshes and make the quantitative report. Our full NASM framework outperforms other cases / methods at all levels of the resolution. One of the main advantages of NASM (w/ or w/o NM CVT) over SIFHDE^2 method is: our results indicate that as long as most of the metrics in the dataset are accurate, NASM performs well during the inference. In contrast, SIFHDE^2 method is highly dependent on the quality of the metric for each individual model, which is more sensitive to inaccuracies of metrics. It is noted that the neural network can find a more general, accurate, and robust embedding than SIFHDE^2.

We also observe an increase of anisotropic mesh quality when using CVT for meshing, instead of using high-d normal metric CVT. Based on our analysis, it is actually a trade-off between anisotropic mesh quality and surface accuracy (feature preserving). Normal metric CVT can push the vertices towards sharp feature, which may distort the anisotropic metric direction and stretching ratio. So that the mesh quality evaluation may become lower. Fig. 11 shows the qualitative comparison on NASM method, NASM without high-d normal metric CVT (with CVT), and SIFHDE^2 method [Zhong et al. 2018]. Meanwhile, the neural high-d embedding computation is much faster (about 1,500× speedup) than traditional SIFHDE^2 method. The normal metric CVT computation takes slightly more time than the general CVT computation.

Fig. 12 shows the additional anisotropic triangular meshes and their corresponding anisotropic RVD results of our NASM on some complicated surfaces from the Thingi10k dataset. The restricted Voronoi cells can well capture the sharp features and curvature anisotropies.

E.2 MGN Dataset

Tab. 5 shows the quantitative evaluation on Multi-Garment Net (MGN) dataset [Bhatnagar et al. 2019] with 154 cloth models (including 96 pants and 58 tops). Fig. 13 shows the additional anisotropic surface meshing results of our NASM method on MGN dataset. The examples of our results can well capture the open boundaries, and detailed cloth wrinkles and folds from tops and pants models.

E.3 FAUST Dataset

Tab 6 shows the quantitative evaluation on FAUST dataset [?] with 200 human scan models of 10 different subjects in 30 different poses. Each scan is a high-resolution and non-watertight triangular mesh. Even though there are some issues in the original meshes, such as feet are not complete, and some fingers are attached to each other, etc., our NASM method can still handle these non-manifold meshes. We use QEM [Garland and Heckbert 1997] to reduce the mesh resolution (e.g., 5% of the original mesh elements) to feed into the neural network. The evaluation is conducted between NASM results and reduced-resolution meshes. Fig. 14 shows the anisotropic surface meshing results of our NASM method on FAUST dataset. Our results demonstrate promising results to capture geometric anisotropies and features around hands, arms, legs, and thin clothes wrinkles on 3D human models.

E.4 Synthetic Models

Fig. 15 shows the anisotropic surface meshing results of our NASM method on synthetic mathematical torus models with different stretching ratios, e.g., 1:2, 1:6, and 1:11. Our results show that the final mesh elements can well match the stretching ratios and directions of the curvature metric fields.

F ADDITIONAL ANALYSES

F.1 Timing

Fig. 16 shows the inference times of NASM embedding computation with respect to the input number of vertices for all three testing datasets, i.e., Thingi10k dataset (80 models), MGN dataset (154 models), and FAUST dataset (200 models). The colormap indicates the maximum anisotropic stretching ratio of the corresponding mesh. It is clear to see that the relationship between the inference time and the input number of mesh vertices is consistently linear among all three datasets.

F.2 Ablation Study

Tab. 7 shows the analysis of w_{lap} value setting in high-d Euclidean embedding loss function, i.e., Equation (4) in the main paper. We compare with different weights of Laplacian loss, such as 0.1, 0.3, and 0.5 on Thingi10k dataset. Fig. 17 shows the qualitative comparison with different weights of Laplacian loss w_{lap}. Finally, it is clear to see that 0.1 is the optimal setting for w_{lap} in our task.

Fig. 18 shows the qualitative results of ablation study on our NASM method with different loss functions, w/o data augmentation, and w/o high-d normal metric CVT. Our proposed dot product loss and full version of NASM demonstrate better performance than other cases.

F.3 Failure Case

Fig. 19 shows that our NASM method may fail on a CAD-like model with very sparse input vertices, since a high-quality curvature-based embedding is very challenging to be predicted in such cases. Once we increase the mesh resolution to some extent, we can obtain a good-quality anisotropic mesh result.

Table 4. Quantitative comparison with our NASM method, NASM without high-d normal metric CVT, and SIFHDE^2 method [Zhong et al. 2018] on 80 models selected from Thingi10k dataset, including smooth surfaces and surfaces with sharp and weak features. All the evaluate metrics are average values of all models from the dataset. The best results are highlighted in bold per different $\#V_{out}$. Note: $\#V_{in}$ is the average number of vertices of all input meshes, $\#V_{out}$ is the average number of vertices of all output meshes, 'Stretch' is the average anisotropic stretching ratios of all models, CD ($\times 10^5$), HD ($\times 10^2$), ECD ($\times 10^2$).

Method	$\#V_{in}$	$\#V_{out}$	Stretch	CD ↓	F1 ↑	NC ↑	HD ↓	ECD ↓	EF1 ↑	T_{em} (s) ↓	G_{avg} ↑	T_{mesh} (s) ↓
NASM	5,982	5,982	12.736	**0.709**	**0.978**	**0.993**	**0.725**	**0.066**	**0.897**	**0.029**	0.745	14.022
	5,982	4,786	12.736	**0.712**	**0.978**	**0.992**	**0.739**	**0.074**	**0.882**	**0.029**	0.747	11.648
	5,982	3,590	12.736	**0.720**	**0.978**	**0.991**	**0.779**	**0.086**	**0.850**	**0.029**	0.748	7.366
	5,982	2,395	12.736	**0.741**	**0.976**	**0.990**	**0.842**	**0.102**	**0.804**	**0.029**	0.749	5.751
	5,982	1,202	12.736	**0.882**	**0.967**	**0.984**	**1.127**	**0.148**	**0.676**	**0.029**	0.744	3.499
	5,982	608	12.736	**1.538**	**0.928**	**0.974**	**1.774**	**0.207**	**0.501**	**0.029**	0.732	**2.092**
NASM	5,982	5,982	12.736	0.779	0.972	0.989	0.882	0.137	0.687	**0.029**	**0.758**	3.780
w/o NM CVT	5,982	4,786	12.736	0.812	0.968	0.988	0.940	0.145	0.635	**0.029**	**0.760**	3.587
(w/ CVT)	5,982	3,590	12.736	0.875	0.963	0.987	1.055	0.155	0.571	**0.029**	**0.764**	**3.354**
	5,982	2,395	12.736	1.026	0.951	0.983	1.207	0.173	0.469	**0.029**	**0.768**	**3.185**
	5,982	1,202	12.736	1.684	0.905	0.975	1.613	0.215	0.290	**0.029**	**0.776**	3.127
	5,982	608	12.736	3.692	0.779	0.962	2.352	0.267	0.170	**0.029**	**0.779**	2.922
SIFHDE^2	5,982	5,982	12.736	0.808	0.969	0.988	0.949	0.146	0.612	49.25	0.729	**3.718**
	5,982	4,786	12.736	0.850	0.965	0.987	1.008	0.154	0.561	49.25	0.731	**3.525**
	5,982	3,590	12.736	0.928	0.959	0.985	1.145	0.166	0.487	49.25	0.732	3.441
	5,982	2,395	12.736	1.109	0.945	0.982	1.330	0.185	0.388	49.25	0.733	3.252
	5,982	1,202	12.736	1.878	0.893	0.975	1.848	0.222	0.249	49.25	0.734	**3.005**
	5,982	608	12.736	4.144	0.766	0.963	2.674	0.270	0.157	49.25	0.730	2.935

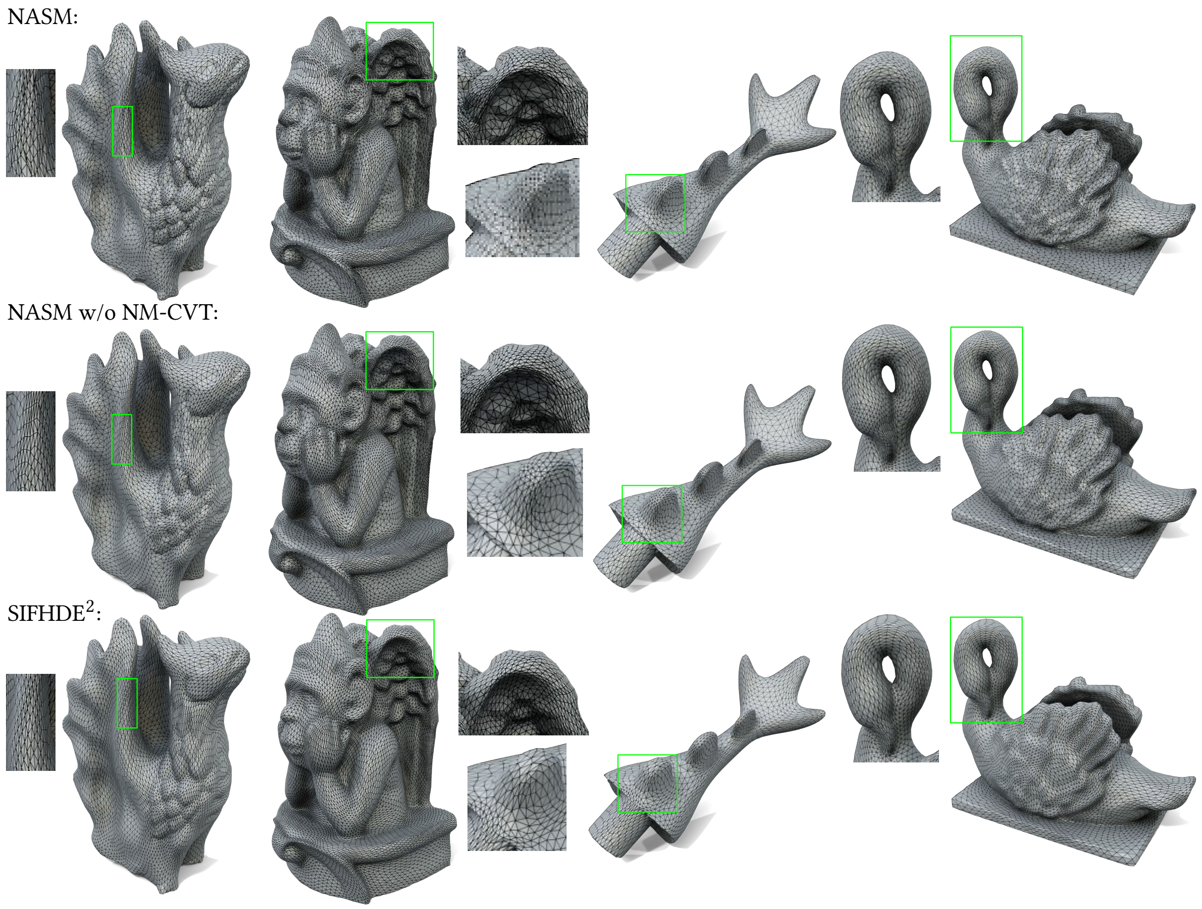


Fig. 11. Qualitative comparison with NASM method, NASM without high-d normal metric CVT (with CVT), and SIFHDE^2 method [Zhong et al. 2018] (from top row to bottom row).

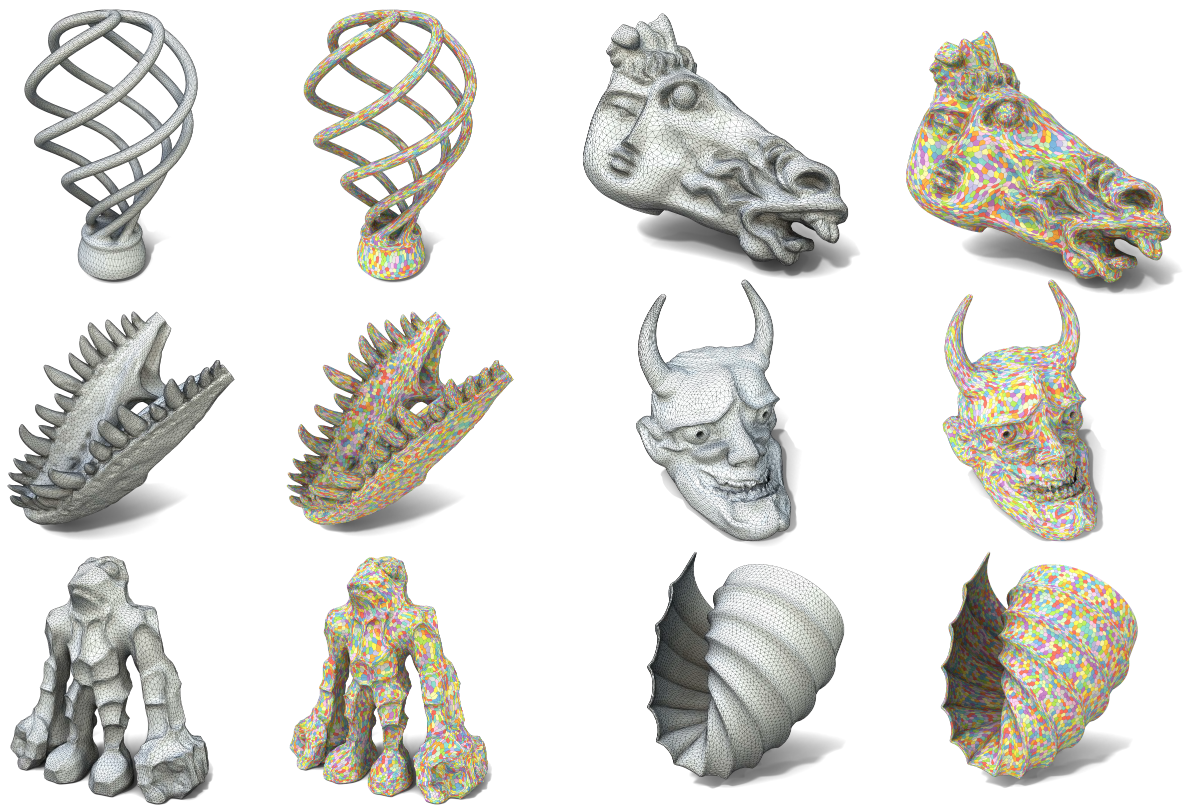


Fig. 12. Anisotropic triangular meshes and their corresponding anisotropic RVD results of our NASM on some complicated surfaces from Thingi10k dataset.

Table 5. Quantitative evaluation on Multi-Garment Net (MGN) dataset with 154 cloth models (including 96 pants and 58 tops). All the evaluate metrics are average values of all models from the dataset. Note: CD ($\times 10^5$), HD ($\times 10^2$), ECD ($\times 10^2$).

#Meshes	$\#V_{in}$	$\#V_{out}$	CD ↓	F1 ↑	NC↑	HD ↓	ECD ↓	EF1 ↑	G_{avg} ↑
154	6,033	6,020	0.549	0.993	0.993	1.216	0.098	0.788	0.725

Table 6. Quantitative evaluation on FAUST dataset with 200 human scans models of 10 different subjects in 30 different poses. All the evaluate metrics are average values of all models from the dataset. Note: CD ($\times 10^5$), HD ($\times 10^2$), ECD ($\times 10^2$).

#Meshes	$\#V_{in}$	$\#V_{out}$	CD ↓	F1 ↑	NC↑	HD ↓	ECD ↓	EF1 ↑
200	8,522	8,505	0.377	0.997	0.987	2.390	0.099	0.791

Table 7. Ablation study on the weight of Laplacian loss with 0.1, 0.3, and 0.5 on 80 models selected from Thingi10k dataset. All the evaluate metrics are average values of all models from the dataset. The best results are highlighted in bold. Note: CD ($\times 10^5$), HD ($\times 10^2$), ECD ($\times 10^2$).

w_{lap}	$\#V_{in}$	$\#V_{out}$	CD ↓	F1 ↑	NC↑	HD ↓	ECD ↓	EF1 ↑	G_{avg} ↑
0.1	5,982	5,982	**0.709**	**0.978**	**0.993**	**0.725**	**0.066**	**0.897**	**0.745**
0.3	5,982	16,398	1.677	0.973	0.991	2.148	0.113	0.773	0.700
0.5	5,982	35,057	2.146	0.967	0.986	6.195	0.200	0.518	0.534

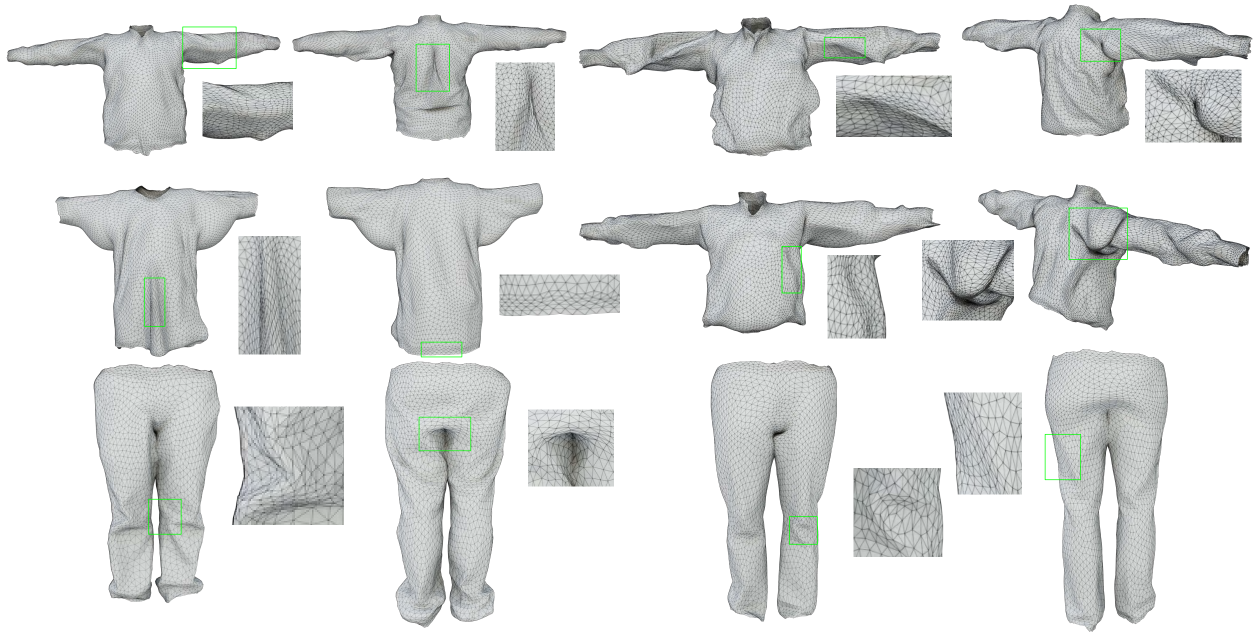


Fig. 13. Additional anisotropic surface meshing results of our NASM method on an unseen testing dataset, e.g., MGN dataset. Several long and short sleeve tops and pants meshes are shown to well capture the open boundaries and detailed cloth wrinkles and folds.

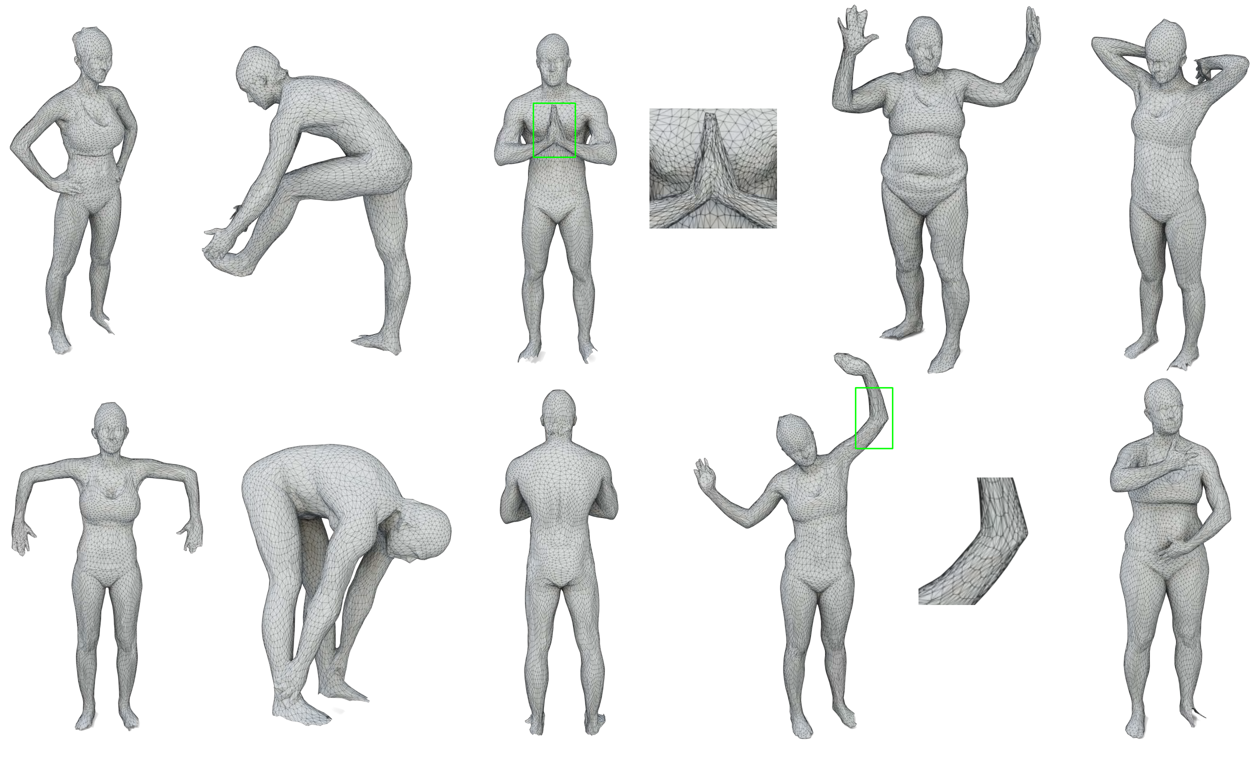


Fig. 14. Additional anisotropic surface meshing results of our NASM method on another unseen testing dataset, e.g., FAUST dataset. Several human subject meshes in different poses are shown to well capture geometric anisotropies and features around hands, arms, legs, and thin clothes wrinkles on 3D human models.

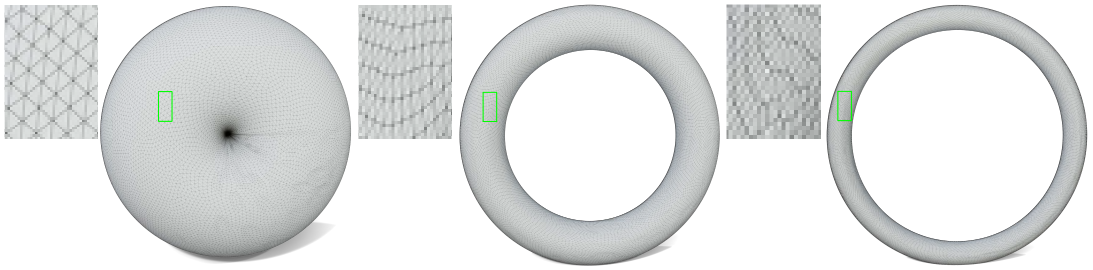


Fig. 15. The qualitative results of our NASM method on synthetic torus models with different stretching ratios. From left to right, the stretching ratios are: 1:2, 1:6, and 1:11, respectively.

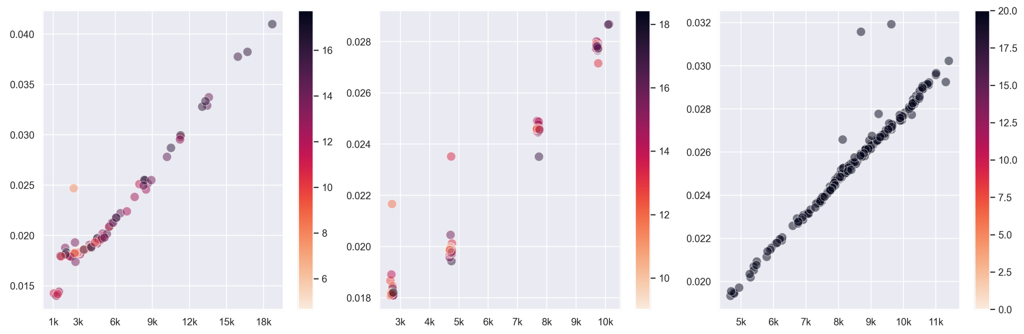


Fig. 16. Inference times of NASM embedding computation with respect to the number of input mesh vertices from three datasets (left to right: 80 models from Thingi10k, 154 models from MGN, and 200 models from FAUST). The colormap indicates the maximum anisotropic stretching ratio of the corresponding mesh.

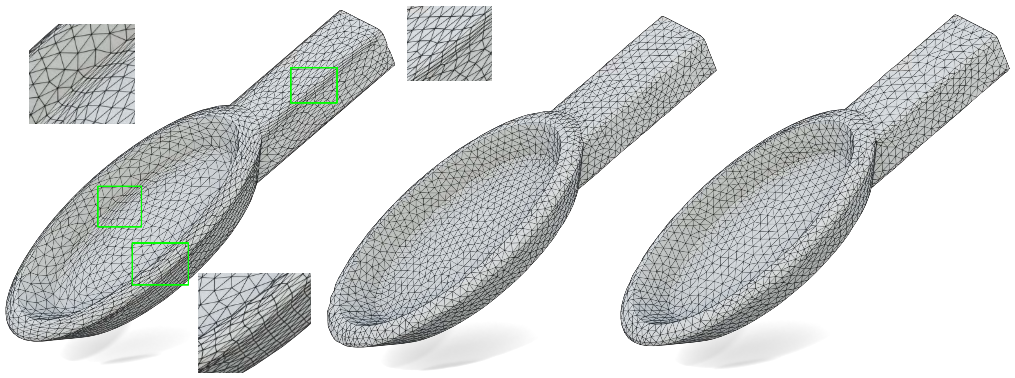


Fig. 17. The qualitative results of ablation study on our NASM method with different weights for Laplacian loss: 0.1, 0.3, and 0.5 (from left to right).

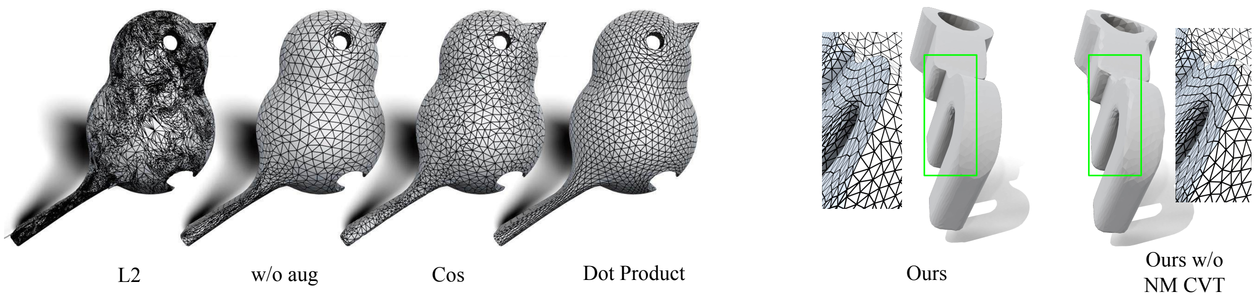


Fig. 18. The qualitative results of ablation study on our NASM method with different loss functions, without data augmentation, and without high-d normal metric CVT.

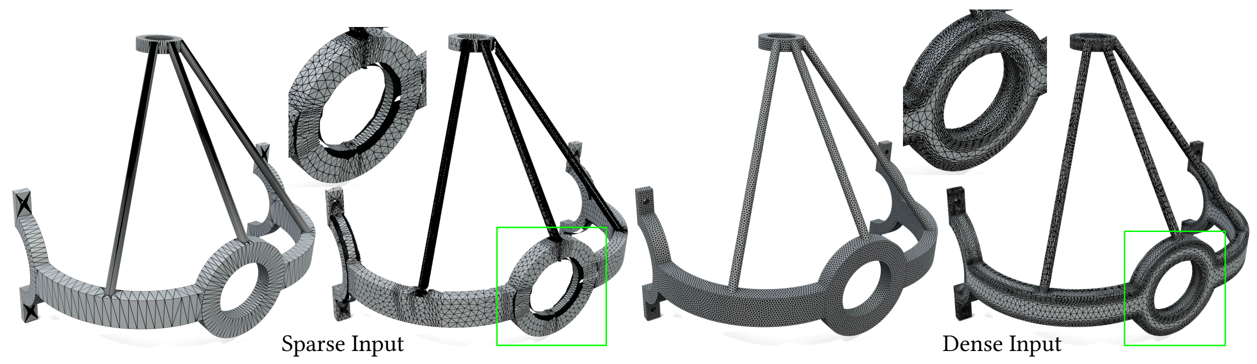


Fig. 19. Failure case: our method cannot handle the CAD-like model with a triangulation with extremely sparse vertices, e.g., 2,406 vertices (left); while our method can successfully handle a denser triangulation, e.g., 20,000 vertices (right).