An Intrinsic Vector Heat Network

Alexander Gao [1 2] **Maurice Chu** [3] **Mubbasir Kapadia** [3] **Ming C. Lin** [2] **Hsueh-Ti Derek Liu** [1]

Abstract

Vector fields are widely used to represent and model flows for many science and engineering applications. This paper introduces a novel neural network architecture for learning tangent vector fields that are intrinsically defined on manifold surfaces embedded in 3D. Previous approaches to learning vector fields on surfaces treat vectors as multi-dimensional scalar fields, using traditional scalar-valued architectures to process channels individually, thus fail to preserve fundamental intrinsic properties of the vector field. The core idea of this work is to introduce a trainable vector heat diffusion module to spatially propagate vector-valued feature data across the surface, which we incorporate into our proposed architecture that consists of vector-valued neurons. Our architecture is invariant to rigid motion of the input, isometric deformation, and choice of local tangent bases, and is robust to discretizations of the surface. We evaluate our Vector Heat Network on triangle meshes, and empirically validate its invariant properties. We also demonstrate the effectiveness of our method on the useful industrial application of quadrilateral mesh generation.

1. Introduction

Tangent vector fields on Riemannian manifolds are a fundamental ingredient in scientific computation, with applications ranging from modeling physical processes on earth (Sabaka et al., 2010), to robotic navigation on complex terrains (van den Berg et al., 2008), to mesh generation (de Goes et al., 2016). The majority of works on *learning* of tangent vector fields rely on neural network architectures that consist of *scalar* neurons (e.g., the standard multilayer perceptron). Despite being straightforward to implement, these scalar-valued architectures assume that each scalar channel can be processed *independently* of the others (e.g. as with RGB color channels in images). However, the multiple channels of a directional vector must be considered jointly rather than independently, as they represent entangled properties (length and orientation). As an example, a simple rotation requires adjusting all channels of the vector to maintain its length; a scalar-valued neural network does not inherently disentangle such vector properties. Scalar-valued architectures also disregard fundamental invariances of tangent vector fields, such as the choice of local tangent bases in which the vectors are expressed (see Fig. 1). This prohibits their generalization to unseen data that does not share the same (arbitrary) choice of bases.

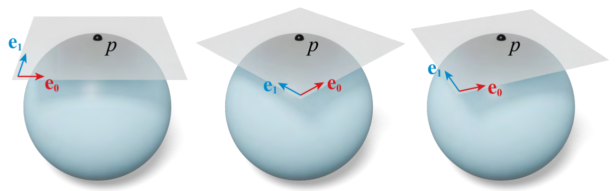


Figure 1: *The tangent plane $T_p\mathcal{M}$ at point p on a manifold $\mathcal{M}$ does not have a canonical choice of basis vectors $\mathbf{e}_0, \mathbf{e}_1$. Our proposed architecture for learning tangent vector fields is invariant to choice of tangent bases.*

In this work, we present a neural network architecture for processing tangent vector fields defined on 2-manifolds embedded in $\mathbb{R}^3$. The key idea is to maintain vector-valued features throughout the architecture, and utilize a trainable vector heat diffusion process (not to be confused with Diffusion Models (Yang et al., 2022)) to ensure our architecture maintains necessary invariances for tangent vector field processing. We begin by reviewing necessary background on tangent vector field processing in Sec. 3. In Sec. 4, we detail our proposed architecture. In Sec. 5, we show that our approach is invariant to choice of local tangent bases, rigid transformation, and isometric deformation of the input, and is robust to different *discretizations* of the manifold surface. Finally, in Sec. 6, we highlight an application of our work in quadrilateral mesh generation for animation.

2. Related Work

Our work is an instance of geometric deep learning (Bronstein et al., 2021) focused on learning tangent vector fields defined on discrete surfaces embedded in $\mathbb{R}^3$. Such sur-

[1]Roblox Research [2]Department of Computer Science, University of Maryland, College Park, USA [3]Roblox Core AI. Correspondence to: Alexander Gao <awgao@umd.edu>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

faces can be represented as point clouds (Guo et al., 2021), implicits (Xie et al., 2022), or analytical functions (Cohen et al., 2019) for simple shapes, e.g. a sphere. While other researchers have focused on learning vector fields defined on the entire volume, such as (Yang et al., 2023), we focus on tangent vector fields defined on the *triangle mesh*, a widely used surface representation for graphics, scientific computing, and engineering applications.

A majority of works on this subject focus on developing fundamental operators in neural networks (such as convolutions) to process *scalar* fields defined on triangle mesh elements, such as vertices (Gong et al., 2019; Lahav & Tal, 2020), edges (Hanocka et al., 2019; Liu et al., 2020; Ludwig et al., 2023), and faces (Feng et al., 2019; Hertz et al., 2020; Hu et al., 2022). As a triangle mesh is merely a graph with triangular faces, graph neural networks (GNNs) (Wu et al., 2021) have also been deployed to learning tasks on meshes, such as (Milano et al., 2020; Pfaff et al., 2021). Despite being effective, the fact that these operations are defined based on the *connectivity* of the mesh makes them sensitive to the quality of the triangulation.

In lieu of this, spectral approaches (Wiersma et al., 2022; Sharp et al., 2022) operate on the *functional space* characterized by differential operators. Network architectures that operate on these function spaces, such as the eigenvectors of the Laplace operator, are more robust to discretization and are able to generalize to other domains (e.g., point clouds) as long as the differential operators are defined.

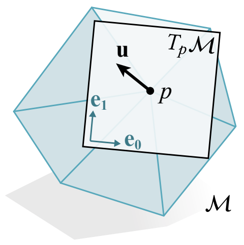
The basic building block of our architecture is the *vector* heat diffusion process with the *connection Laplacian*. This shares similarities with GNN architectures motivated by diffusion processes. As pointed out in Hansen & Gebhart (2020), the graph convolution can be viewed as heat diffusion with the connection Laplacian using *forward Euler* integration. This perspective motivates the development of *Sheaf Neural Networks* (Hansen & Ghrist, 2019; Bodnar et al., 2022; Battiloro et al., 2023) which learn a graph- and task-specific connection Laplacian to improve expressiveness of GNNs. But given domain knowledge of the underlying graph (e.g., a graph that approximates a Riemannian manifold), Barbero et al. (2022) demonstrate that a pre-determined connection Laplacian could lead to better performance and generalization. Accordingly, since our method indeed focuses on a specific type of graph, manifold triangle meshes, we build a connection Laplacian derived from differential geometry (Sharp et al., 2019). Our architecture with a deterministic connection Laplacian leads to superior generalization across triangle meshes, compared to approaches that rely on learning graph-specific Laplacians.

Despite the existence of several learning approaches for scalar data, few approaches have been proposed for learning vector fields on surfaces. Previous methods such as (Dielen et al., 2021) rely on scalar-valued architectures to output multiple scalar channels that are naively interpreted as vectors. Such approaches treat each channel independently and thus fail to capture key invariances (see Sec. 5). This severely hinders generalization to unseen triangulations and shapes. In contrast, our architecture maintains the vector-valued features throughout the forward pass, similar to (Deng et al., 2021), ensuring invariance to isometry, rigid motion, and choice of tangent bases.

3. Preliminaries

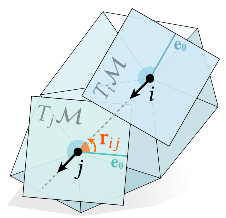
We draw inspiration from differential geometry to define the building blocks of our architecture, based on the notion of tangent vectors (Sec. 3.1), parallel transport (Sec. 3.2), and the heat equation (Sec. 3.3), which we describe next.

3.1. Basis Invariant Tangent Vectors



Given a 2-manifold surface $\mathcal{M}$ embedded in $\mathbb{R}^3$, the tangent plane T_p^M for a given point $p \in \mathcal{M}$ is a 2D space that is orthogonal to the surface normal of $\mathcal{M}$ at point p. A tangent plane T_p^M can be defined by the spanning of an *arbitrary choice* of two orthogonal basis vectors $\mathbf{e}_0, \mathbf{e}_1$. A 2D tangent vector $\mathbf{u} \in T_p^M$ can then be expressed as a linear combination of bases vectors $\mathbf{u} = u_0\mathbf{e}_0 + u_1\mathbf{e}_1$. For computational convenience, these tangent vectors $\mathbf{u}$ are often represented as a complex number $\mathbf{u} = u_0 + \imath u_1 \in \mathbb{C}$ of their coefficients u_0, u_1 (Knöppel et al., 2013; Vaxman et al., 2016). Note that the coefficients u_0, u_1 will change, by a coordinate transformation, depending on the choice of tangent bases $\mathbf{e}_0, \mathbf{e}_1$ in order to represent the same tangent vector. Since the choice of bases is arbitrary, an important invariance for tangent vector processing is to guarantee that the method is independent of the choice of bases (see Fig. 1).

3.2. Parallel Transport of Tangent Vectors



An edge vector $\mathbf{e}_{ij} = \mathbf{v}_j - \mathbf{v}_i \in \mathbb{R}^3$ between two adjacent vertices i, j with locations $\mathbf{v}_i, \mathbf{v}_j \in \mathbb{R}^3$ can be expressed as a tangent vector in T_i^M and T_j^M via the *logarithmic* map (sometimes alternately referred to as the *exponential* map) (Schmidt et al., 2006). A simple way to compute the logarithmic map is to represent the edge vector $\mathbf{e}_{ij}$ in the polar coordinate (l_{ij}, θ_j) of the tangent plane T_i^M, where $l_{ij} = \|\mathbf{e}_{ij}\|$ is the edge length and $\theta_j \in [0, 2\pi)$ denotes the angular coordinate (normalized to 2π) of this edge vector from a (arbitrarily chosen) tangent basis $\mathbf{e}_0$, see (Knöppel et al.,

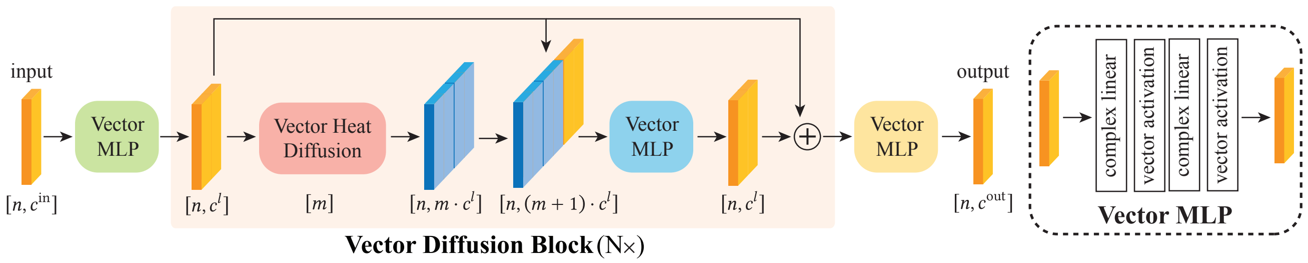


Figure 2: *Our vector heat network is a neural network of complex-valued neurons (Bassey et al., 2021) with (1) a Vector Heat Diffusion module (see Sec. 4.1) and (2) a vector MLP module (see Sec. 4.2). Starting with a Vector MLP to transform input features from $\mathbb{C}^{n\times c^{in}}$ to $\mathbb{C}^{n\times c^l}$, our method consists of several layers of the Vector Heat Diffusion (red) and Vector MLP (blue) with skip connections, followed by another Vector MLP to map the feature to output dimensions.*

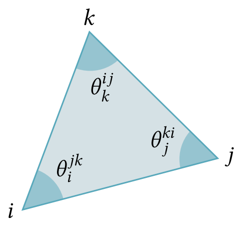
2013) for more details. As the edge vector $\mathbf{e}_{ij}$ exists in both tangent planes T_i^M, T_j^M, one can compute the angular difference between T_i^M, T_j^M and obtain the coordinate transformation $\mathbf{r}_{ij} \in \mathbb{C}$ (a rotation) needed to make sure that $log_i(\mathbf{e}_{ij}) \in T_i^M$ is mapped to $log_j(\mathbf{e}_{ij}) \in T_j^M$ (see inset). Transporting tangent vectors from T_i^M to T_j^M following the computed rotation $\mathbf{r}_{ij}$ leads to as-parallel-as-possible transport, or *parallel transport*. Note that only the angle of these vectors varies as we traverse the curved surface; their length remains unchanged. This motivates constructing a network that can preserve these disentangled vector properties.

3.3. Vector Heat Diffusion

The heat equation on a tangent vector field $u : \mathcal{M} \to \mathbb{C}$ can be written as

$$\frac{d}{dt}u = \Delta_c u. \tag{1}$$

where Δ_c is the *connection Laplacian*. This vector heat equation characterizes how a "vector" gets diffused in space. Intuitively, diffusing a single vector heat source will smear it out to its neighborhood with smaller magnitude while maintaining its direction to be as-parallel-as-possible (see Fig. 3). This is different from classic heat diffusion where no notion of parallel transport is captured and can cause inconsistent vector directions if one naively applies scalar heat diffusion to each vector field channel independently.



Discretization Several previous works (Knöppel et al., 2015; Sharp et al., 2019; Stein et al., 2020) have defined the vector heat equation on a triangle mesh. The main ingredient is a *discrete* connection Laplacian $\mathbf{L}_c \in \mathbb{C}^{n\times n}$ which is an n x n complex-valued matrix, where n denotes the number of vertices in the mesh. One way to build $\mathbf{L}_c$ is by accumulating a 3-by-3 complex matrix

$$-\frac{1}{2}\begin{bmatrix} c_j + c_k & -c_k\mathbf{r}_{ij} & -c_j\mathbf{r}_{ik} \\ -c_k\mathbf{r}_{ji} & c_k + c_i & -c_i\mathbf{r}_{jk} \\ -c_j\mathbf{r}_{ki} & -c_i\mathbf{r}_{kj} & c_i + c_j \end{bmatrix} \tag{2}$$

associated to each triangle ijk into the corresponding entries defined by the vertex index. We use $c_i = \cot\theta_i^{jk}, c_j = \cot\theta_j^{ki}, c_k = \cot\theta_k^{ij}$ to shorten the expression, and $\mathbf{r}_{ij} \in \mathbb{C}$ (so as $\mathbf{r}_{jk}, \mathbf{r}_{ki}$) to denote a the 2D rotation (represented as a unit complex number) that *parallel transports* a tangent vector from the tangent plane T_i^M at vertex i to the tangent plane T_j^M at vertex j (see Sec. 3.2).

As the *forward Euler* method is well-known to be unstable under large time steps, we compute the numerical solution to the vector heat equation using the *implicit Euler* method. Specifically, a single step of the diffusion is defined as

$$\mathbf{u}_{t+1} = (\mathbf{M} + s\mathbf{L}_c)^{-1}\mathbf{M}\mathbf{u}_t \tag{3}$$

where s to denotes the time-step size, $\mathbf{M} \in \mathbb{C}^{n\times n}$ is a n x n diagonal mass matrix whose entries are complex-valued, with vertex area as the real component, and zero imaginary component. Eq. 3 diffuses a vector into smaller magnitudes and as-parallel-as-possible orientations (see Fig. 3).

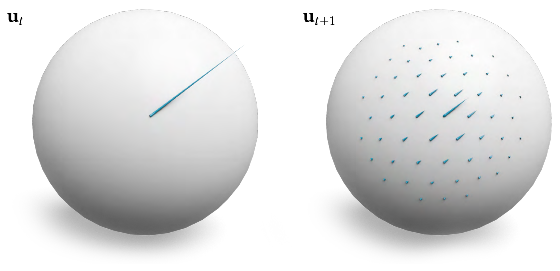


Figure 3: *The vector heat diffusion process presented in Eq. 3 smears out a tangent vector field $\mathbf{u}_t$ to its neighbors to obtain another tangent vector field $\mathbf{u}_{t+1}$.*

3.4. Generalization to N-Rosy Fields

In several applications, one may want to learn a *N-way rotational symmetry* fields (Palacios & Zhang, 2007) that are invariant under rotation of an integer multiple of $2\pi/N$ (see Fig. 4). For instance, one of our applications in quadrilateral meshing requires to output a 4-Rosy fields in order to mesh a surface with the "cross" pattern on most vertices.

The generalization to N-Rosy fields is straightforward given the complex number representation (Sec. 3.1). Since multiplication with (unit) complex numbers represents rotations, raising a complex number to the power of N factors out all the N-ways rotational symmetry (de Goes et al., 2016). Thus, to measure the difference between, e.g., 4-Rosy fields, one simply measures the difference between $\mathbf{u}^4$ (see Sec. 6).

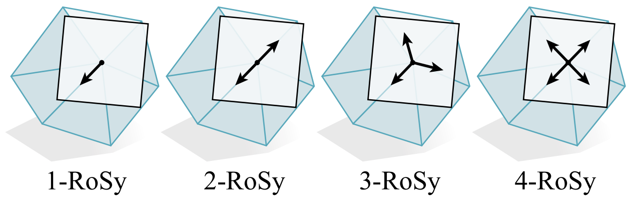


Figure 4: *N-Rosy fields refer to tangent vectors that are* N-way rotationally symmetric *(Palacios & Zhang, 2007). For instance, $N = 1$ refers to the usual 2D tangent vector, $N = 2$ to a straight line, and $N = 4$ to a "cross" field.*

4. Vector Heat Network

Our vector-valued neural network for processing tangent vector fields utilizes vector-valued neurons (similar to (Deng et al., 2021; Bassey et al., 2021)) with vector operations (e.g., parallel transport and the vector heat equation). Maintaining the vector nature of our data throughout results in an architecture that is invariant to isometries, rigid transformations, and the choice of tangent bases (see Sec. 5)

The input to our network is a set of tangent vectors defined on the vertices of a surface triangle mesh. A Vector Heat Network consists of several blocks of learned heat diffusion to harness local information, and *vector MLPs* to increase the expressiveness (see Fig. 2). The output is a set of tangent vector fields defined on vertices. These output fields could be regular tangent vectors or tangent vectors with N rotational symmetry (see Sec. 3.4) defined on the tangent plane. In this section, we will illustrate individual components of our **Vector Heat Network** in more details.

4.1. Learned Vector Heat Diffusion

Starting with a set of tangent vectors $\mathbf{u} \in \mathbb{C}^{n \times c^l}$ (represented as complex numbers, see Sec. 3.1), where n denotes the number of vertices and c^l denotes the number of tangent vector fields. We harness information from local neighborhoods by solving the vector heat equation with the implicit Euler integration in Eq. 3, inspired by (Sharp et al., 2022). As the implicit Euler requires an expensive step of solving a linear system, we utilize spectral acceleration (Donati et al., 2022) to speed up the process. Specifically, given the solution of the generalized eigenvalue problem of the discrete connection Laplacian $\mathbf{L}_c$ (see Eq. 2),

$$\mathbf{L}_c \Phi = \mathbf{M} \Phi \Sigma \tag{4}$$

where $\Phi = \{\Phi_i\} \in \mathbb{C}^{n \times k}$ is the stack of k eigenvectors with the lowest frequencies and $\Sigma = diag(\lambda_i) \in \mathbb{C}^{k \times k}$ is a diagonal matrix with corresponding eigenvalues. We use $\mathbf{M}$ to denote the mass matrix of vertex areas such that $\Phi_i^\top \mathbf{M} \Phi_i = 1$. k is a user-defined number to specify how many eigenvalues/eigenvectors are in use. Then the vector diffusion process in Eq. 3 can be approximated with

$$\mathbf{u}_{t+1} = \Phi \begin{bmatrix} e^{-\lambda_1 s} \\ e^{-\lambda_2 s} \\ \vdots \\ e^{-\lambda_k s} \end{bmatrix} \odot (\Phi^\top \mathbf{M} \mathbf{u}_t) \tag{5}$$

where $\odot$ denotes element-wise multiplication. Such a spectral acceleration replaces linear solves with matrix multiplications, thus is significantly faster for small k. In our implementation, we set $k = 128$.

Inspired by (Sharp et al., 2022), we treat time-step size s in Eq. 5 as trainable parameters. Intuitively, the network learns whether to diffuse the vectors over a small or large local neighborhood. Specifically, each *Vector Heat Diffusion* module (Fig. 2) consists of m trainable time steps. Each time step s_i will diffuse the input feature $\mathbf{X}^l \in \mathbb{C}^{n \times c^l}$ at layer l into a set of diffused features $\mathbf{Y}_i^l \in \mathbb{C}^{n \times c^l}$ via Eq. 5. Thus, a collection of m time steps $[s_1, \cdots, s_m]$ will turn an input feature $\mathbf{X}^l$ with size n x c^l into a set of diffused features $\mathbf{Y}^l = [\mathbf{Y}_1^l, \cdots, \mathbf{Y}_m^l]$ with size n x m_{c^l}.

4.2. Vector Linear Layers and Non-linearity

After the vector heat diffusion module, we use a vector-valued MLP that consists of a per-vertex linear layer

$$\mathbf{Z}^l = \mathbf{Y}^l \mathbf{W}^l \tag{6}$$

where $\mathbf{W}^l \in \mathbb{R}^{mc^l \times c^{l+1}}$ is a matrix of size mc^l-by-c^{l+1} that linearly combine the complex-valued features at each vertex into $\mathbf{Z}^l \in \mathbb{C}^{n \times c^{l+1}}$. Then we follow the idea presented by (Wiersma et al., 2022) to apply non-linearities σ (e.g., ReLU) on the magnitude of each complex feature as

$$\mathbf{X}_{ij}^{l+1} = \sigma(\|\mathbf{Z}_{ij}^l\| - b_j^l) \cdot \frac{\mathbf{Z}_{ij}^l}{\|\mathbf{Z}_{ij}^l\|} \tag{7}$$

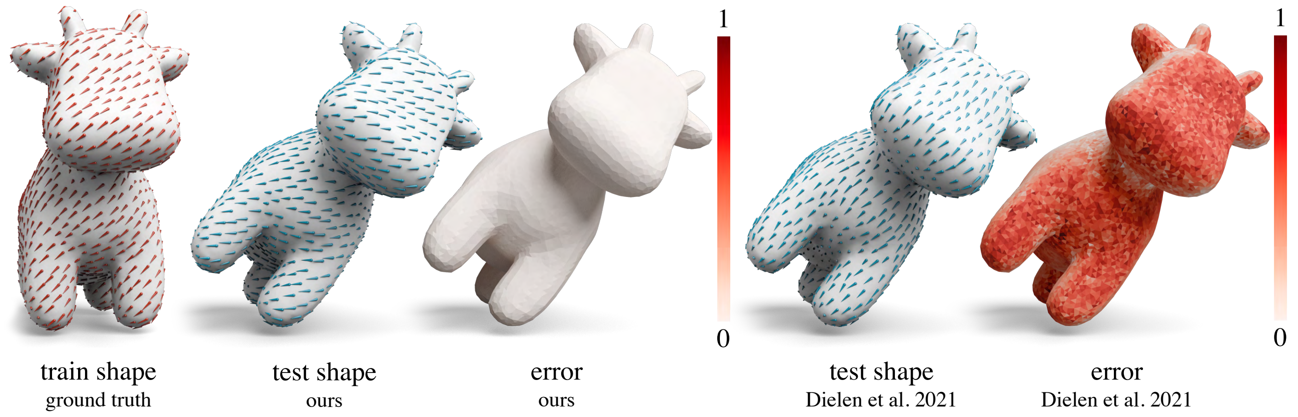


Figure 5: *Our architecture is invariant to rigid transformation. A model trained on a mesh at one orientation (1st) generalizes to its rigidly transformed counterpart (2nd), outputting a tangent vector field with no error (3rd). This differs from the baseline method of Dielen et al. (2021), which outputs a different vector field (4th) with high error (5th).*

where we use $\mathbf{Z}^l_{ij}$ to denote the entry corresponding to the ith row and the jth column in $\mathbf{Z}$, $b^l_j \in \mathbb{R}$ is a bias term for each channel added to the feature norm. In our experiments, we use the ReLU activation – if the complex feature norm $\|\mathbf{Z}^l_{ij}\|$ is smaller than the bias b^l_j, the complex feature is set to $\mathbf{0}$, otherwise it is unchanged in the output $\mathbf{X}^{l+1}$.

In summary, our overall architecture (see Fig. 2) consists of several **Vector Diffusion Blocks**. Each block contains a vector heat diffusion layer (see Sec. 4.1) and a vector-valued MLP with two hidden layers (see Sec. 4.2). We also have two extra vector MLPs (green, yellow) to adapt to a different number of input/output channels. Our architecture is invariant to the choice of tangent bases because the Vector Heat Diffusion module has parallel transport baked in (see Sec. 3.2), making it invariant to the bases. Furthermore, the vector heat diffusion process is *intrinsic*. This ensures that the entire architecture is also invariant to rigid transformations and isometries of the underlying shape. These desirable invariances result in a general network architecture for processing tangent vector fields on manifolds.

5. Experiment: Invariance Properties

Our architecture possesses several fundamental invariances, which we highlight and empirically validate here, distinguishing it from the scalar-valued approach of (Dielen et al., 2021). For comparison, we faithfully re-implement their method (Figure 5). These invariances arise from the fact that all of our operations (gradient, heat diffusion, and the per-vertex linear layer) are *intrinsic*, which implies that our architecture is invariant to how the mesh sits in the space.

Invariance to Rigid Motion If rigid motion invariant input features are used, then our method will be invariant to rigid transformations of the underlying shape (see Fig. 5).

Invariance to Tangent Bases We leverage the characteristic that parallel transport has already factored out the influence of choice of tangent bases, which is *baked in* to our connection Laplacian. This property makes our architecture invariant to the choice of tangent bases (Fig. 6).

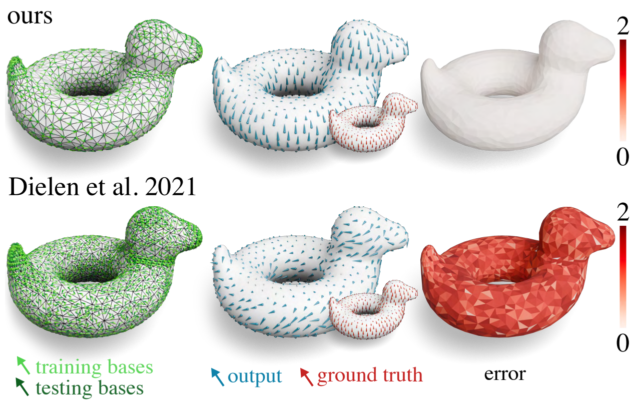


Figure 6: *Our method is invariant to choice of tangent bases. Given a model trained on the default bases (light green), where the arrow indicates direction of* ***e0****, our method produces the same result (blue) as GT (red) even when the model is evaluated under different choice of bases (dark green), in contrast to (Dielen et al., 2021) that outputs a different result with high error (right).*

Invariance to Isometry Our method is invariant to isometric deformation (isometries) of the input (see Fig. 7), due to the intrinsic construction of our method.

Robustness to Discretizations Our main ingredient for "message passing" relies on the vector heat diffusion with the connection Laplacian. In contrast to the method by Bodnar et al. (2022) that learns the connection Laplacian for a graph,

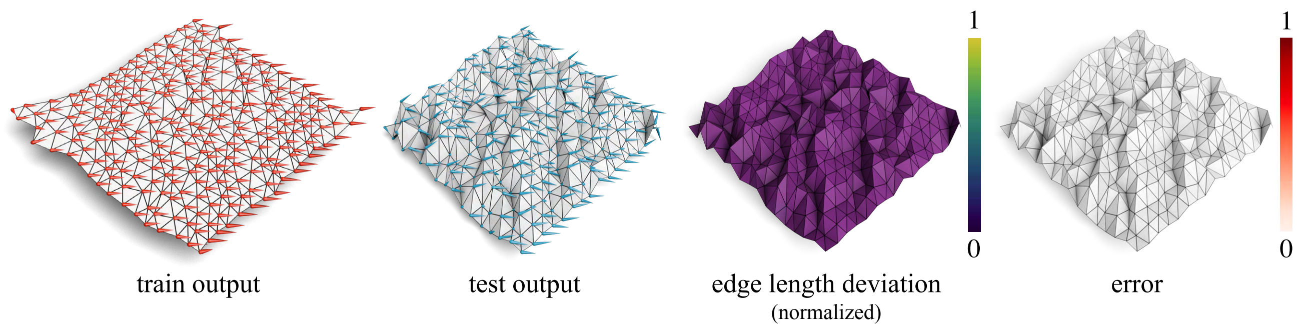


Figure 7: *Our method is invariant to isometric deformation. We train on a flat paper (first) and evaluate on its crumpled counterpart (second). Since these two meshes are nearly isometric (third), our method produces consistent results (fourth).*

the connection Laplacian in our set-up is *deterministic* by the underlying shape, with *connections* determined by the parallel transport (Sec. 3.2). This characteristic ensures that our learned parameters are generalizable to meshes with different connectivities. This property is crucial to tangent vector field processing because the choice of tangent bases is arbitrary (see Fig. 1): one can find an infinite number of *valid* tangent bases $\mathbf{e}_0, \mathbf{e}_1$ that are orthogonal to the normal vector. On a single vertex, there is already an infinite number of choices and the total combinations of basis choices also grows exponentially with the number of vertices in the mesh. This implies that baking in the property of basis invarance is important because solving it with data augmentation is intractable due to the infinite number of basis combinations.

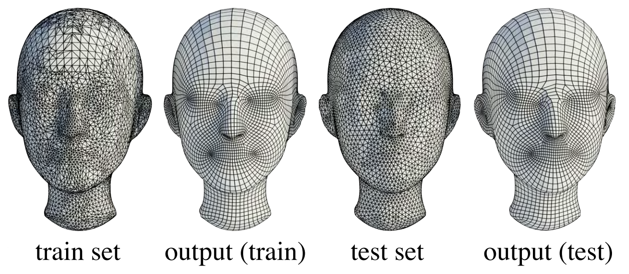


Figure 8: *This work exploits the benefit of the spectral method (see Eq. 5) and gains robustness to different discretizations. Trained on one triangulation (1st), our model generalizes to a different one (3rd), producing consistent output (quad meshing results from Sec. 6) shown here.)*

Baseline Implementation Details Our re-implementation of the architecture by Dielen et al. (2021) is as consistent as possible with theirs: for the local feature network ((Gong et al., 2019)), we use a spiral sequence length of $k = 20$ vertices, with 4 spiral convolution layers of intermediate size $[16, 256, 512, 1024]$, where the first 3 layers use vertex-centric spiral indices, and the last layer uses face-centric spiral indices; the global feature ((Qi et al., 2017)) consists of 1024 channels; and each input vertex is represented as its 3D position and normal direction.

6. Experiment: Quadrilateral Remeshing

In this section, we evaluate our method on triangle meshes, though the same principles apply to other domains where the connection Laplacian is available (e.g. point clouds).

6.1. Experiment Setup

To demonstrate our method's effectiveness, we evaluate it on the task of quadrilateral remeshing. Given a triangle mesh $\mathcal{M}$, we use the per-channel gradient of the first $c^{\text{in}} = 15$ channels of the Heat Kernel Signature (HKS) (Sun et al., 2009) as input features to our network, giving c^{in} vector-valued features per vertex. The output is a 4-Rosy cross field defined on each vertex (see Sec. 3.4 and Fig. 9). We then interpolate the per-vertex cross field onto faces (see Equation 6.1), followed by off-the-shelf algorithms by Bommes et al. (2009) and (Ebke et al., 2013) to turn the per-face cross field into a quadrilateral mesh (see Fig. 9).

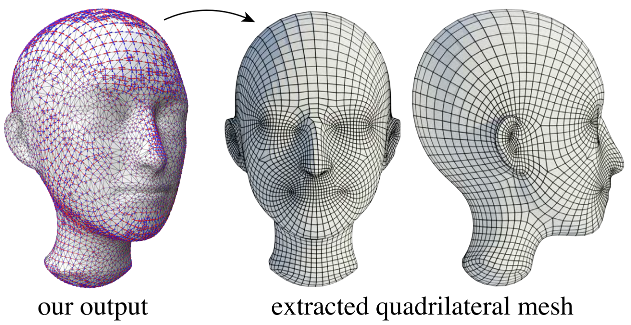


Figure 9: *After obtaining the output cross field by our network, we use off-the-shelf quad mesh extraction method by Bommes et al. (2009) to obtain a quadrilateral mesh with edges aligned with the predicted cross field.*

Dataset We train our network on a dataset generated from the workflow described in (Dielen et al., 2021), with two modifications: (1) Instead of the DFAUST dataset used by (Dielen et al., 2021), we assemble a custom library of artist-created template avatar heads, around which we wrap the SMPL (Loper et al., 2023) head topology; (2) for each of the template meshes, we create $100 - 1,000$ augmentations/variations, using a custom tool for deforming faces, based upon normal-driven ARAP deformation (Liu & Jacobson, 2021). This expands the training data distribution away from the parametric SMPL model. The training dataset consists of 1100 triangle meshes with associated ground truth vector fields. The test dataset consists of 115 samples.

Loss Function Since our output is a 4-Rosy field, we define the loss function as the mean squared error (MSE) on the output tangent vector field $\mathbf{u}^4$ raised to the power of 4 (see Sec. 3.4). In addition to measuring errors on the directions, we also want to measure errors on the magnitude of the cross field (smaller crosses lead to smaller polygon). Combining the two leads to our following loss function

$$\mathcal{L}(\mathbf{u}, \hat{\mathbf{u}}) = \sum_{i=1}^{N} \frac{\mathbf{M}_{ii}}{A} \left[\underbrace{\left| \frac{\|u_i\| - \|\hat{u}_i\|}{\|u_i\|} \right|}_{\text{magnitude}} + \underbrace{\left(1 - \frac{u_i^4}{\|u_i^4\|} \cdot \frac{\hat{u}_i{}^4}{\|\hat{u}_i{}^4\|}\right)}_{\text{direction}} \right] \quad (8)$$

where $\mathbf{M}_{ii}$ denotes the vertex area at vertex i, $A = \mathrm{Tr}(\mathbf{M})$ is the total area of the mesh, $u_i, \hat{u}_i \in \mathbb{C}$ denote the output and the ground truth tangent vector fields on the vertex i, respectively. The first "magnitude" term simply measure the relative difference in magnitude between the output and the ground truth fields. The second "direction" term measures their angular difference with the cosine similarity.

Transporting Vectors from Vertices to Faces Our model predicts vectors at mesh vertices, but in order to directly leverage existing methods for field-guided quadrilateral remeshing (Bommes et al., 2009), they should be expressed on the face tangent planes. Naively averaging the three vector predictions from a given face's three incident vertices will not produce a correct result, as the vectors are expressed with respect to their individual vector tangent planes, so they cannot be averaged directly. We must therefore account for the parallel transport from each of the vertex tangent planes to the face tangent plane. Once all three vectors are in a shared frame, then we may simply average their values.

To transport a vector from its tangent plane $T_i\mathcal{M}$ at a vertex v_i to the tangent plane $T_{ijk}\mathcal{M}$ at face f_{ijk}, we choose an edge that is incident to both elements (e.g., edge e_{ij} or e_{ik}). The chosen edge can be expressed in terms of its angular rotation from the local basis in both the vertex $T_i\mathcal{M}$ and face tangent planes $T_{ijk}\mathcal{M}$. Thus, we leverage this angular difference to transport a vector from vertex to face tangent plane. This transport can be constructed as an operator in

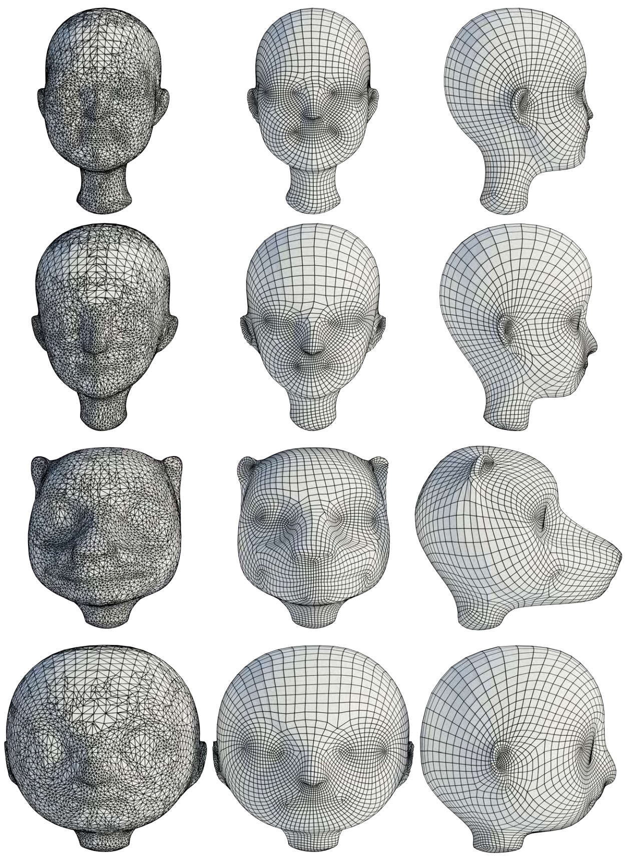


Figure 10: *Vector-field-guided quadrilateral meshing results of various character heads from the test dataset.*

a pre-processing step, as it only depends on the mesh, i.e. once we have computed the angular differences, they may be used to transport any vectors from vertices to faces. Let this matrix be $\mathbf{T}_{angle} \in \mathbb{R}^{F \times 3}$, where each row corresponds to a face, and each of the three values corresponds to the transition angle (in radians) needed to transport a vector from the incident vertex at that index, to the face. Then we can assemble the complex matrix *operator*:

$$\mathbf{T} = e^{\imath \mathbf{T}_{angle}} \quad (9)$$

To compute the transported, averaged result per face,

$$\hat{\mathbf{u}}_f = \frac{1}{3} \sum \mathbf{T} \odot \hat{\mathbf{u}}_v[\mathbf{F}] \quad (10)$$

where the $[\cdot]$ represents an indexed selection from the per-vertex predictions based on vertex indices from the face matrix $\mathbf{F}$, and the sum $\sum(\cdot)$ denotes a per-face addition along the last dimension of the product $\mathbf{T} \odot \hat{\mathbf{u}}_v[\mathbf{F}]$.

Implementation Details For our experiments, we use $N = 6$ vector diffusion blocks with a hidden dimension

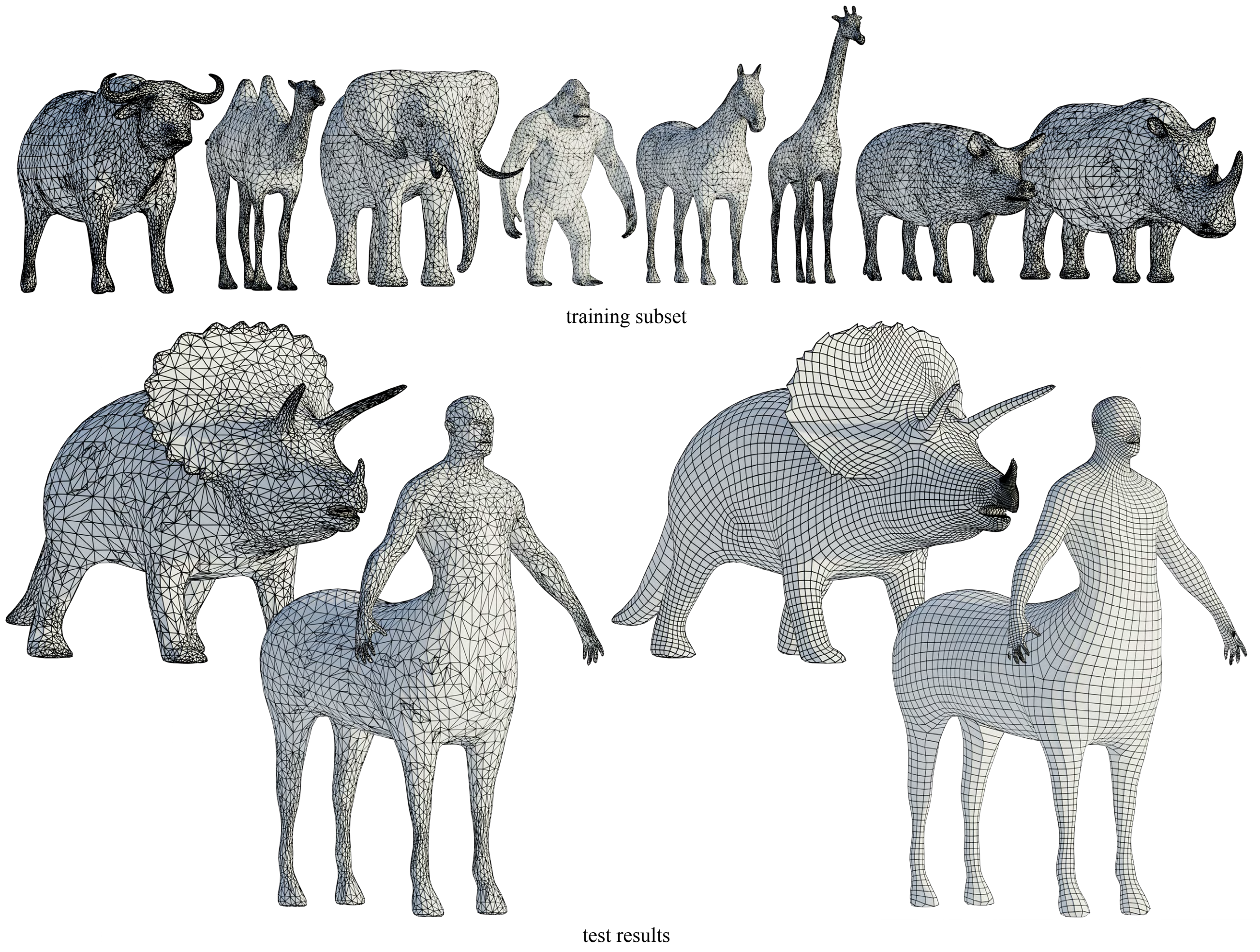


Figure 11: *Our learned vector-field-guided quadrilateral remeshing can generalize to other datasets and classes of objects.*

of $c^l = 256$ channels (see Fig. 2). We train for $3,000$ epochs, with initial learning rate of $1e-4$, decayed by a factor of 0.85 every 150 epochs. In the Vector MLP layer, we use Dropout (Srivastava et al., 2014) set to 0.5, and L2 regularization (weight decay) with a value of $1e-3$, which mitigates overfitting. We train on a single NVIDIA Tesla T4 GPU, for about 20 hours. For computing a parametrization from the predicted vector field, we rely on (Bommes et al., 2009), rather than (Campen et al., 2015) used by (Dielen et al., 2021), due to the latter not being open sourced.

6.2. Results

Character Heads In Fig. 10, we display several results of our learned vector field-guided quadrilateral meshing on the character head dataset. Our model generalizes across different types of character heads.

Animals Fig. 11 shows results on a different dataset that we constructed from 40 ground truth quadrilateral meshes of animals. The trained model generalizes to *centaur* with its human-like upper body and horse-like lower body, even though such a combination is not contained in the training dataset. The *triceratops* result generally looks suitable for downstream usage, though its frill (collar) shows some irregularity, likely due to lack of sufficient training data.

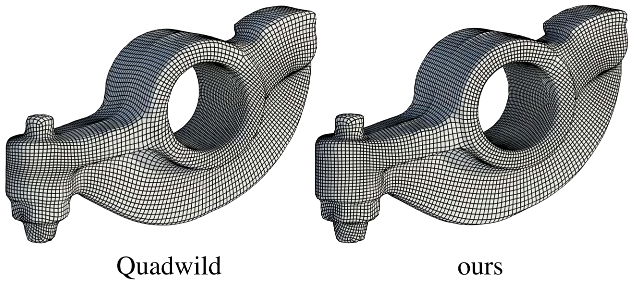


Figure 12: *We supervise our method with an optimization-based quad meshing algorithm – Quadwild by [Pietroni et al. 2021], faithfully reproducing their result.*

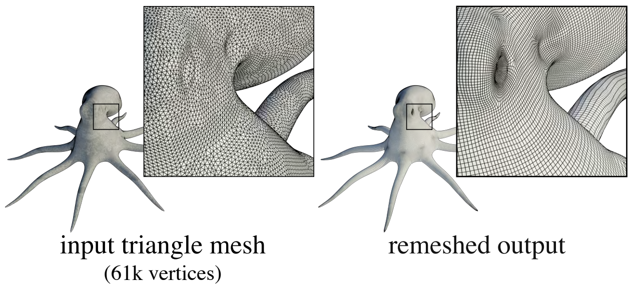


Figure 13: *Our method is scalable to high resolution input meshes. We demonstrate a quad meshing result on this octopus mesh with over 60k vertices.*

Additional experimental results Our method can faithfully reproduce the results from existing quad meshing algorithms (Fig. 12), such as Quadwild (Pietroni et al., 2021). We also note that our method imposes no restriction on the genus of the shape on which vector fields may be learned, as demonstrated by successful generalization to genus-one shapes in Fig. 6 and Fig. 12. We also show that our method easily scales to high resolution meshes (Fig. 13). Please see Appendix A for enlarged detail views.

Ablation Studies We compare the test set performance of various types of input features, including *principal curvature directions*, and the gradients of: *Heat Kernel Signature* (Sun et al., 2009), *Gaussian curvature*, and *mean curvature*. For multi-channeled feature types, we normalize each of its channels individually, such that its mean vector length across all vertices is unit. For all feature types *except PCD*, we also rotate each channel by $\frac{\pi}{2}$ radians, and concatenate these rotated vector features along the channel dimension. In principle, this means that each input feature channel and its rotated counterpart span the local tangent space, allowing the network to better exploit all degrees of freedom.

Input Feature Type	Direction Loss	Magnitude Loss
∇HKS	0.106 ± 0.278	$\mathbf{0.077 \pm 0.148}$
∇GC	0.139 ± 0.312	0.096 ± 0.261
∇MC	$\mathbf{0.105 \pm 0.276}$	0.077 ± 0.514
PCD	0.128 ± 0.313	0.090 ± 0.288

Table 1: *Input features are evaluated by comparing their mean test loss and associated variance of their magnitude and direction components.* ∇HKS *denotes channel-wise gradient of the scalar-valued* Heat Kernel Signature*,* ∇GC *denotes gradient of* Gaussian Curvature*,* ∇MC *denotes gradient of* Mean Curvature*, and* PCD *denotes scaled principal curvature directions. We find that* ∇HKS *leads to best overall performance. While* ∇MC *performs best on the directional loss component only, it displays high variance in the magnitude loss component.*

7. Conclusion

We present a neural network architecture based on vector heat diffusion to process tangent vector fields defined on manifold surfaces. Unlike existing works, our method is invariant to rigid transformations and the choice of tangent plane bases, and is robust to different triangulations. These properties jointly make this network a generalizable architecture for learning tangent vector fields across surfaces.

Additional Applications We show an application of our method to robot path planning on curved terrain (Fig. 14).

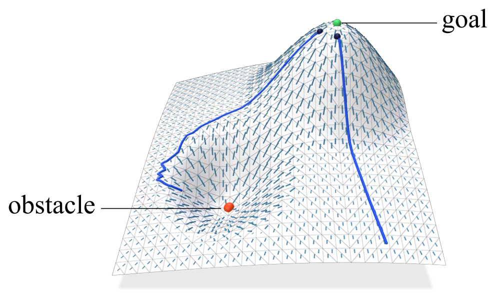


Figure 14: *Tangent vector fields are commonly used to guide a robot's path planning [Patil et al. 2010]. We train our method to generate such a navigation vector field on the terrain mesh to assist motion planing as shown.*

Future Directions As the vector heat equation can be defined in different domains, such as point clouds and graphs, generalizing our architecture to different domains could enable a even wider range of applications. Exploring novel architectures from *complex neural networks* (Bassey et al., 2021) or even generalizing to quaternions, such as (Zhu et al., 2018), could lead to novel variants of our architecture. For the quadrilateral meshing application, developing a larger, more diverse dataset could be an important step towards a generic learning-based remesher.

Acknowledgements

Thank you to Jing Liang for help executing the robot motion planning application in ROS, Jihyun Yoon for help assembling the 3D heads dataset, and Kiran Bhat for helpful discussion. This research is supported in part by Barry Mersky Endowed Professorship, Capital One Endowed Professorship, and Maryland E-Nnovate Initiative Fund.

Impact Statement

The goal of this work is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

References

Barbero, F., Bodnar, C., de Ocáriz Borde, H. S., Bronstein, M. M., Velickovic, P., and Liò, P. Sheaf neural networks with connection laplacians. In *Topological, Algebraic and Geometric Learning Workshops 2022, 25-22 July 2022, Virtual*, volume 196 of *Proceedings of Machine Learning Research*, pp. 28–36. PMLR, 2022.

Bassey, J., Qian, L., and Li, X. A survey of complex-valued neural networks. *CoRR*, abs/2101.12249, 2021. URL https://arxiv.org/abs/2101.12249.

Battiloro, C., Wang, Z., Riess, H., Lorenzo, P. D., and Ribeiro, A. Tangent bundle filters and neural networks: From manifolds to cellular sheaves and back. In *IEEE International Conference on Acoustics, Speech and Signal Processing ICASSP 2023, Rhodes Island, Greece, June 4-10, 2023*, pp. 1–5. IEEE, 2023.

Bodnar, C., Giovanni, F. D., Chamberlain, B. P., Lió, P., and Bronstein, M. M. Neural sheaf diffusion: A topological perspective on heterophily and oversmoothing in gnns. In Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., and Oh, A. (eds.), *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022.

Bommes, D., Zimmer, H., and Kobbelt, L. Mixed-integer quadrangulation. *ACM Transactions On Graphics (TOG)*, 28(3):1–10, 2009.

Bronstein, M. M., Bruna, J., Cohen, T., and Velickovic, P. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. *CoRR*, abs/2104.13478, 2021. URL https://arxiv.org/abs/2104.13478.

Campen, M., Bommes, D., and Kobbelt, L. Quantized global parametrization. *Acm Transactions On Graphics (tog)*, 34(6):1–12, 2015.

Cohen, T., Weiler, M., Kicanaoglu, B., and Welling, M. Gauge equivariant convolutional networks and the icosahedral CNN. In Chaudhuri, K. and Salakhutdinov, R. (eds.), *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pp. 1321–1330. PMLR, 2019.

de Goes, F., Desbrun, M., and Tong, Y. Vector field processing on triangle meshes. In *Special Interest Group on Computer Graphics and Interactive Techniques Conference, SIGGRAPH '16, Anaheim, CA, USA, July 24-28, 2016, Courses*, pp. 27:1–27:49. ACM, 2016.

Deng, C., Litany, O., Duan, Y., Poulenard, A., Tagliasacchi, A., and Guibas, L. J. Vector neurons: A general framework for so(3)-equivariant networks. In *2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021*, pp. 12180–12189. IEEE, 2021.

Dielen, A., Lim, I., Lyon, M., and Kobbelt, L. Learning direction fields for quad mesh generation. In *Computer Graphics Forum*, volume 40, pp. 181–191. Wiley Online Library, 2021.

Donati, N., Corman, E., Melzi, S., and Ovsjanikov, M. Complex functional maps: A conformal link between tangent bundles. *Comput. Graph. Forum*, 41(1):317–334, 2022.

Ebke, H.-C., Bommes, D., Campen, M., and Kobbelt, L. Qex: Robust quad mesh extraction. *ACM Transactions on Graphics (TOG)*, 32(6):1–10, 2013.

Feng, Y., Feng, Y., You, H., Zhao, X., and Gao, Y. Meshnet: Mesh neural network for 3d shape representation. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, pp. 8279–8286. AAAI Press, 2019.

Gong, S., Chen, L., Bronstein, M. M., and Zafeiriou, S. Spiralnet++: A fast and highly efficient mesh convolution operator. In *2019 IEEE/CVF International Conference on Computer Vision Workshops, ICCV Workshops 2019, Seoul, Korea (South), October 27-28, 2019*, pp. 4141–4148. IEEE, 2019.

Guo, Y., Wang, H., Hu, Q., Liu, H., Liu, L., and Bennamoun, M. Deep learning for 3d point clouds: A survey. *IEEE Trans. Pattern Anal. Mach. Intell.*, 43(12):4338–4364, 2021.

Hanocka, R., Hertz, A., Fish, N., Giryes, R., Fleishman, S., and Cohen-Or, D. Meshcnn: a network with an edge. *ACM Transactions on Graphics (ToG)*, 38(4):1–12, 2019.

Hansen, J. and Gebhart, T. Sheaf neural networks. *CoRR*, abs/2012.06333, 2020. URL https://arxiv.org/abs/2012.06333.

Hansen, J. and Ghrist, R. Learning sheaf laplacians from smooth signals. In *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2019, Brighton, United Kingdom, May 12-17, 2019*, pp. 5446–5450. IEEE, 2019.

Hertz, A., Hanocka, R., Giryes, R., and Cohen-Or, D. Deep geometric texture synthesis. *ACM Trans. Graph.*, 39(4): 108, 2020.

Hu, S., Liu, Z., Guo, M., Cai, J., Huang, J., Mu, T., and Martin, R. R. Subdivision-based mesh convolution networks. *ACM Trans. Graph.*, 41(3):25:1–25:16, 2022.

Knöppel, F., Crane, K., Pinkall, U., and Schröder, P. Globally optimal direction fields. *ACM Trans. Graph.*, 32(4): 59:1–59:10, 2013.

Knöppel, F., Crane, K., Pinkall, U., and Schröder, P. Globally optimal direction fields. *ACM Transactions on Graphics (ToG)*, 32(4):1–10, 2013.

Knöppel, F., Crane, K., Pinkall, U., and Schröder, P. Stripe patterns on surfaces. *ACM Trans. Graph.*, 34(4):39:1–39:11, 2015.

Lahav, A. and Tal, A. Meshwalker: deep mesh understanding by random walks. *ACM Trans. Graph.*, 39(6): 263:1–263:13, 2020.

Liu, H. D. and Jacobson, A. Normal-driven spherical shape analogies. *Comput. Graph. Forum*, 40(5):45–55, 2021.

Liu, H. D., Kim, V. G., Chaudhuri, S., Aigerman, N., and Jacobson, A. Neural subdivision. *ACM Trans. Graph.*, 39(4):124, 2020.

Loper, M., Mahmood, N., Romero, J., Pons-Moll, G., and Black, M. J. Smpl: A skinned multi-person linear model. In *Seminal Graphics Papers: Pushing the Boundaries, Volume 2*, pp. 851–866. 2023.

Ludwig, I., Tyson, D., and Campen, M. Halfedgecnn for native and flexible deep learning on triangle meshes. *Computer Graphics Forum*, 42(5):e14898, 2023.

Milano, F., Loquercio, A., Rosinol, A., Scaramuzza, D., and Carlone, L. Primal-dual mesh convolutional neural networks. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

Palacios, J. and Zhang, E. Rotational symmetry field design on surfaces. *ACM Trans. Graph.*, 26(3):55, 2007.

Pfaff, T., Fortunato, M., Sanchez-Gonzalez, A., and Battaglia, P. W. Learning mesh-based simulation with graph networks. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.

Pietroni, N., Nuvoli, S., Alderighi, T., Cignoni, P., Tarini, M., et al. Reliable feature-line driven quad-remeshing. *ACM Transactions on Graphics*, 40(4):1–17, 2021.

Qi, C. R., Su, H., Mo, K., and Guibas, L. J. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 652–660, 2017.

Sabaka, T. J., Hulot, G., and Olsen, N. Mathematical properties relevant to geomagnetic field modeling. In *Handbook of geomathematics*. 2010.

Schmidt, R. M., Grimm, C., and Wyvill, B. Interactive decal compositing with discrete exponential maps. *ACM Trans. Graph.*, 25(3):605–613, 2006.

Sharp, N., Soliman, Y., and Crane, K. The vector heat method. *ACM Transactions on Graphics (TOG)*, 38(3): 1–19, 2019.

Sharp, N., Attaiki, S., Crane, K., and Ovsjanikov, M. Diffusionnet: Discretization agnostic learning on surfaces. *ACM Transactions on Graphics (TOG)*, 41(3):1–16, 2022.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

Stein, O., Wardetzky, M., Jacobson, A., and Grinspun, E. A simple discretization of the vector dirichlet energy. *Comput. Graph. Forum*, 39(5):81–92, 2020.

Sun, J., Ovsjanikov, M., and Guibas, L. A concise and provably informative multi-scale signature based on heat diffusion. In *Computer graphics forum*, volume 28, pp. 1383–1392. Wiley Online Library, 2009.

van den Berg, J. P., Lin, M. C., and Manocha, D. Reciprocal velocity obstacles for real-time multi-agent navigation. In *2008 IEEE International Conference on Robotics and Automation, ICRA 2008, May 19-23, 2008, Pasadena, California, USA*, pp. 1928–1935. IEEE, 2008.

Vaxman, A., Campen, M., Diamanti, O., Bommes, D., Hildebrandt, K., Ben-Chen, M., and Panozzo, D. Directional field synthesis, design, and processing. In Mitra, N. J. (ed.), *SIGGRAPH ASIA 2016, Macao, December 5-8, 2016 - Courses*, pp. 15:1–15:30. ACM, 2016.

Wiersma, R., Nasikun, A., Eisemann, E., and Hildebrandt, K. Deltaconv: anisotropic operators for geometric deep learning on point clouds. *ACM Transactions on Graphics (TOG)*, 41(4):1–10, 2022.

Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., and Yu, P. S. A comprehensive survey on graph neural networks. *IEEE Trans. Neural Networks Learn. Syst.*, 32(1):4–24, 2021.

Xie, Y., Takikawa, T., Saito, S., Litany, O., Yan, S., Khan, N., Tombari, F., Tompkin, J., Sitzmann, V., and Sridhar,

S. Neural fields in visual computing and beyond. *Comput. Graph. Forum*, 41(2):641–676, 2022.

Yang, L., Zhang, Z., Song, Y., Hong, S., Xu, R., Zhao, Y., Shao, Y., Zhang, W., Yang, M., and Cui, B. Diffusion models: A comprehensive survey of methods and applications. *CoRR*, abs/2209.00796, 2022. doi: 10.48550/ARXIV.2209.00796. URL https://doi.org/10.48550/arXiv.2209.00796.

Yang, X., Lin, G., Chen, Z., and Zhou, L. Neural vector fields: Implicit representation by explicit learning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2023, Vancouver, BC, Canada, June 17-24, 2023*, pp. 16727–16738. IEEE, 2023.

Zhu, X., Xu, Y., Xu, H., and Chen, C. Quaternion convolutional neural networks. In Ferrari, V., Hebert, M., Sminchisescu, C., and Weiss, Y. (eds.), *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part VIII*, volume 11212 of *Lecture Notes in Computer Science*, pp. 645–661. Springer, 2018.

A. Additional experimental results (enlarged detail view)

Our method can faithfully reproduce the results from existing quad meshing algorithms (Fig. 15), such as Quadwild (Pietroni et al., 2021). We also show that our method easily scales to high resolution meshes (Fig. 16).

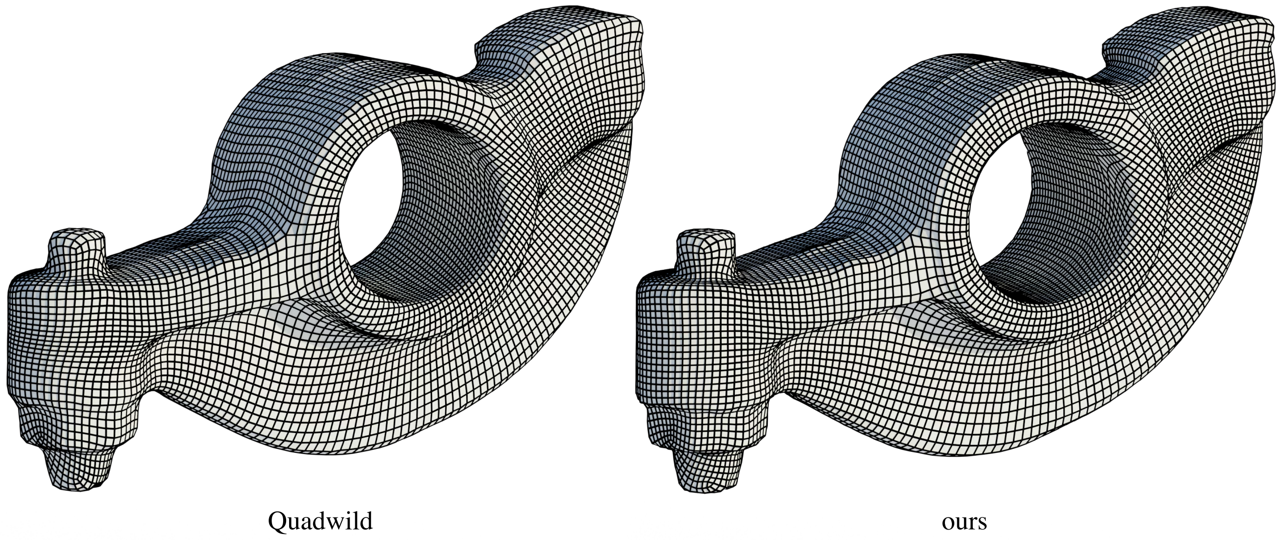


Figure 15: *We supervise our method with an optimization-based quad meshing algorithm – Quadwild by [Pietroni et al. 2021]. Our method is able to reproduce the result from Quadwild.*

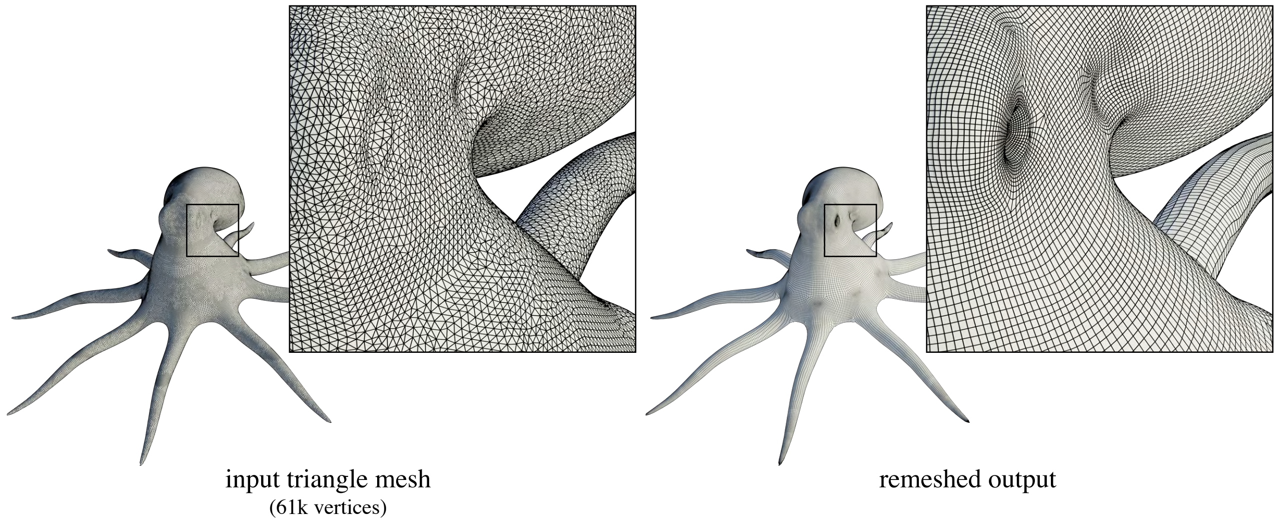


Figure 16: *Our method is scalable to high resolution input meshes. We demonstrate a quad meshing result on this octopus mesh with over 60k vertices.*