Learning Conjugate Direction Fields for Planar Quadrilateral Mesh Generation

Jiong Tao[1], **Yong-Liang Yang**[1], **Bailin Deng**[2*]

[1] Department of Computer Science, University of Bath, United Kingdom
[2] School of Computer Science and Informatics, Cardiff University, United Kingdom
jt2337@bath.ac.uk, y.yang@cs.bath.ac.uk, DengB3@cardiff.ac.uk

Abstract

Planar quadrilateral (PQ) mesh generation is a key process in computer-aided design, particularly for architectural applications where the goal is to discretize a freeform surface using planar quad faces. The conjugate direction field (CDF) defined on the freeform surface plays a significant role in generating a PQ mesh, as it largely determines the PQ mesh layout. Conventionally, a CDF is obtained by solving a complex non-linear optimization problem that incorporates user preferences, i.e., aligning the CDF with user-specified strokes on the surface. This often requires a large number of iterations that are computationally expensive, preventing the interactive CDF design process for a desirable PQ mesh. To address this challenge, we propose a data-driven approach based on neural networks for controlled CDF generation. Our approach can effectively learn and fuse features from the freeform surface and the user strokes, and efficiently generate quality CDF respecting user guidance. To enable training and testing, we also present a dataset composed of 50000+ freeform surfaces with ground-truth CDFs, as well as a set of metrics for quantitative evaluation. The effectiveness and efficiency of our work are demonstrated by extensive experiments using testing data, architectural surfaces, and general 3D shapes.

Code — https://github.com/jiongtj/Learning-CDF

1 Introduction

Planar quadrilateral (PQ) meshes are special piecewise linear surface representations that consist entirely of planar quad faces. They are widely used in computer-aided design, particularly for modeling discrete architectural surfaces, due to several key advantages (Pottmann 2013; Pottmann et al. 2015). First, the planarity of quad faces significantly reduces the cost of fabricating panels with physical materials such as glass. Second, compared to common triangle meshes, PQ meshes have lower vertex valence, resulting in reduced complexity for conjoining supporting beams along each edge. Third, the edges of PQ meshes often form intuitive layouts that align with human aesthetic preferences.

Given a freeform surface representing the desired design geometry, discretizing it into a PQ mesh is a challenging problem that has been studied extensively (Liu et al. 2006; Zadravec, Schiftner, and Wallner 2010; Liu et al. 2011). The process generally consists of two main stages. In the first stage, an initial quad mesh with a plausible distribution of elements (vertices, edges, and faces) is generated to serve as the mesh layout. In the second stage, a geometric optimization is performed to refine the layout and enforce planarity of each quad face by modifying vertex positions. Due to the non-convex nature of the optimization problem, a key challenge is ensuring a high-quality initial layout, such that the subsequent optimization can successfully generate a PQ mesh that closely approximates the input freeform surface.

Prior works control the edge distribution of the initial mesh layout using a conjugate direction field (CDF) such as the principal (curvature) direction field (PDF) (Liu et al. 2006), where the mesh edges align with two families of polylines that are discrete counterparts of integral curves of the CDF. The conjugacy of the CDF ensures the initial mesh faces are approximately planar, which is crucial for the success of subsequent PQ mesh optimization (Liu et al. 2006). However, unlike PDFs which are uniquely determined by surface geometry except in isotropic regions, CDFs are not unique and have high degrees of freedom over the surface. Therefore, additional conditions are needed to derive a CDF that is consistent with the user's design intent. A common approach is to introduce user-specified strokes on some parts of the surface to indicate preferred CDF directions. These strokes serve as constraints for a non-linear optimization that computes a smooth CDF suitable for quad mesh initialization (Liu et al. 2011). However, this process is highly time-consuming, often requiring many iterations of user input and optimization to obtain a satisfactory CDF.

To improve the efficiency of CDF design, we propose a learning-based approach that can directly generate CDFs without complex optimization. Instead of hand-crafting features, we use neural networks to learn surface features from geometric information such as vertex positions and normals. Our approach supports conditional field generation, where user-provided strokes guide the CDF computation. This capability enables generating diverse fields with different flow behaviors on the same surface, which is valuable for design exploration. To provide meaningful control, we propose a global stroke representation that can be seamlessly integrated with the surface representation for feature ex-

*Corresponding author.

traction. Since collecting real-world data from actual design sessions is impractical, we construct a synthetic dataset inspired by (Deng et al. 2023). The synthesis process emulates real design workflows by tracing streamlines of a ground truth CDF to approximate user strokes. This dataset enables comprehensive training and testing of our learning-based CDF generation approach. Extensive experiments demonstrate that our method can effectively generate CDFs that respect user constraints, while being an order of magnitude faster than traditional optimization techniques. Moreover, it generalizes well to real-world architectural surfaces and general 3D shapes, making it well-suited for practical design applications.

In summary, our main contributions are:

- A novel approach for efficient CDF generation using deep neural networks, avoiding complex optimization.
- A conditional CDF generation method that incorporates user-specified stroke constraints based on a novel global stroke representation.
- A large-scale synthetic dataset that enables training and testing for data-driven CDF generation.

2 Related Work

PQ Mesh Planar quadrilateral (PQ) meshes, where all faces are quadrilaterals with coplanar vertices, are commonly used in architectural design and fabrication (Glymph et al. 2004; Pottmann et al. 2007), particularly for glass structures. Unlike general quadrilateral meshes that can be structured in different ways (Bommes et al. 2013) (e.g., using Morse-Smale complexes (Dong et al. 2006; Huang et al. 2008), scalar fields (Dong, Kircher, and Garland 2005; Tong et al. 2006), vector/cross fields (Alliez et al. 2003; Ray et al. 2006; Kaelberer, Nieser, and Polthier 2007; Bommes, Zimmer, and Kobbelt 2009; Ebke et al. 2013; Lyon et al. 2019), hybrid representations (Fang et al. 2018)), PQ meshes are a discrete counterpart of conjugate curve networks on surfaces (Bobenko and Suris 2008). (Liu et al. 2006) derived quad mesh along the principal curvature lines and applied vertex perturbation optimization to achieve face planarity. (Zadravec, Schiftner, and Wallner 2010) formulated the design of conjugate curve networks as a vector field design problem, and obtained CDF by optimizing the vector field. Later, (Liu et al. 2011) extracted general PQ meshes by computing smooth CDF on reference surfaces, satisfying user-defined directional constraints. (Diamanti et al. 2014) utilized a 2^2 Polyvector field as initialization, deforming it to the closest CDF. However, the aforementioned methods mainly formulated the CDF design as a non-linear optimization problem involving numerous variables and a set of nonlinear constraints. Due to its non-linearity, solving such an optimization problem is highly challenging. Moreover, as the model size increases, the computational cost becomes prohibitively high for interactive applications. Recently, a novel learning-based method for PQ mesh design using a single sketch has been proposed by (Deng et al. 2023). It allows users to draw structural lines as the surface boundary and contours, annotate occlusions, and define sparse feature lines indicating the directions of PQ mesh edges. During training, labeled PQ mesh data is required as ground truth, making it challenging to generate a sufficient number of PQ meshes for constructing a comprehensive training dataset. Moreover, this method takes a single-view 2D sketch as input and only predicts projected edge directions in 2D, which limits its usage for intuitive CDF generation on 3D freeform surfaces and general shapes.

Deep Learning on 3D Data In recent years, various deep learning methods on 3D data have been proposed to address 3D vision and graphics problems, such as point cloud processing (Charles et al. 2017; Wang et al. 2019), connectivity construction from discrete data (Sharp and Ovsjanikov 2020; Rakotosaona et al. 2021), neural mesh processing (Potamias, Ploumpis, and Zafeiriou 2022; Chen et al. 2023), 3D shape generation (Hui et al. 2022; Zhang et al. 2023), just to name a few. A comprehensive review is beyond the scope of this paper and more background information can be found in related surveys (Bronstein et al. 2017; Guo et al. 2020; Xiao et al. 2020; Xu, Mu, and Yang 2023). In our work, we apply the widely used DGCNN (Wang et al. 2019) to extract both local and global latent features from the input shape while seamlessly fusing features respecting the stroke guidance.

Learning Directional Fields With the advances of 3D deep learning, the generation of directional fields based on neural networks has also attracted research attention. (Girard et al. 2020) learned a frame field from satellite images to align with object boundaries, which improves segmentation accuracy. (Dielen et al. 2021) proposed a learning-based frame fields generation method for field-guided quadrangulation. However, this method is trained and tested merely on a human body dataset which highly restricts its generalization capabilities for freeform PQ mesh generation. Also, the frame field is directly predicted based on the geometric shape (i.e., the human body) without user guidance, thus unsuitable for interactive architectural design purposes. A self-supervised cross field learning framework was developed by (Wei et al. 2024) for feature-aligned point cloud unsampling. A novel neural framework for learning vector/cross fields, based on vector heat diffusion, was designed by (Gao et al. 2024), offering the invariance to rigid transformation and isometric deformation of the inputs. (Dong et al. 2025) introduced a self-supervised cross field generation method, exploring the connections between the neural representation of a triangle mesh and its PDF for quadrangulation. The cross fields learned by these methods stem from PDFs, and thus cannot be generalized for CDF-based PQ mesh generation as in our work.

3 Preliminary

Direction fields on surfaces are fundamental to quadrilateral mesh generation, as they can guide the orientation and flow of mesh edges. Among them, Conjugate Direction Fields (CDF) provide a compact representation that encodes the locally planar structures essential for generating Planar Quadrilateral (PQ) meshes.

We first formally define a CDF. On a triangle mesh $\mathcal{M} = \{\mathcal{V}, \mathcal{F}\}$, a pair of direction fields $\{(\mathbf{u}_j, \mathbf{v}_j)\}$ defined on each

mesh face f_j is called *conjugate* if the following condition is satisfied:

$$\kappa_{j,1}(\mathbf{u}_j \cdot \mathbf{d}_{j,1})(\mathbf{v}_j \cdot \mathbf{d}_{j,1}) + \kappa_{j,2}(\mathbf{u}_j \cdot \mathbf{d}_{j,2})(\mathbf{v}_j \cdot \mathbf{d}_{j,2}) = 0, \quad (1)$$

where $\mathbf{d}_{j,1}, \mathbf{d}_{j,2}$ are unit principal direction vectors on f_j, and $\kappa_{j,1}, \kappa_{j,2}$ are the corresponding principal curvatures.

The Principal Direction Field (PDF), which consists of the principal direction pairs $(\mathbf{d}_{j,1}, \mathbf{d}_{j,2})$ over the surface, is a special and well-known case of a CDF, as it trivially satisfies the conjugacy condition (1). However, unlike a PDF, which is uniquely determined by surface geometry (except at umbilic points), a general CDF is not unique. This non-uniqueness provides high degrees of freedom, offering significant design flexibility for PQ mesh generation.

This flexibility, however, also presents a significant challenge: a CDF is ill-defined by geometry alone. To specify a single, desirable CDF from this high-dimensional space, additional conditions are required. Typical conditions include:

1. *Field Smoothness*: To ensure a regular and coherent mesh layout while avoiding unnecessary singularities, the field must be smooth across neighboring faces.
2. *User-Defined Constraints*: To capture design intent, the field should adhere to various user-specified conditions. One primary example is the alignment with user-provided strokes on the surface.

Traditional methods formulate the generation of a smooth CDF that respects user constraints as a complex, constrained non-linear optimization problem (Liu et al. 2011):

$$\begin{aligned} \min_{\mathbf{u},\mathbf{v}} \quad & E_s(\mathbf{u}, \mathbf{v}) \\ \text{s.t.} \quad & \text{Conjugacy constraints,} \\ & \text{User-specified constraints.} \end{aligned} \quad (2)$$

Here, E_s represents a smoothness energy term that measures the variation of the field across the surface.

Due to the non-linearity of this optimization problem (particularly the conjugacy constraint), its solution is highly challenging and computationally expensive. This process often becomes a significant bottleneck, preventing the interactive CDF design and iterative exploration required in practical architectural applications. To address this challenge, this paper proposes a learning-based method to produce high-quality CDFs efficiently, avoiding the costly numerical optimization process entirely.

After obtaining the CDF, PQ mesh generation aims to construct a quad mesh whose edges are aligned with the optimized field. Generally, an initial quad mesh is generated via a global parameterization method from CDFs and then refined through numerical optimization to produce high-quality, planar quads (Liu et al. 2011), as illustrated in Fig. 1.

4 Method

4.1 Overview

The pipeline of our learning-based method is shown in Fig. 2. Given a triangle mesh $\mathcal{M} = \{\mathcal{V}, \mathcal{F}\}$ representing the reference freeform surface and a set of user-specified stroke curves $\mathcal{S} = \{\mathbf{S}_i\}$ on the mesh, our goal is to learn two families of (piecewise-linear) direction fields $\{(\mathbf{u}_j, \mathbf{v}_j)\}$ defined on each mesh face f_j. These direction fields should satisfy two properties: i) the local conjugacy to form a valid CDF, and ii) the alignment with the user-specified stroke curves to reflect design intent. Once the CDF is computed, we derive a global parameterization whose iso-parameter lines align with the CDF (Bommes, Zimmer, and Kobbelt 2009), and then extract a quad mesh from this parameterization (Ebke et al. 2013) with mesh edges forming a discrete counterpart of the conjugate curve network. The initial quad mesh is further optimized by perturbing vertex positions to enforce quad face planarities (Deuss et al. 2015), resulting in the final PQ mesh.

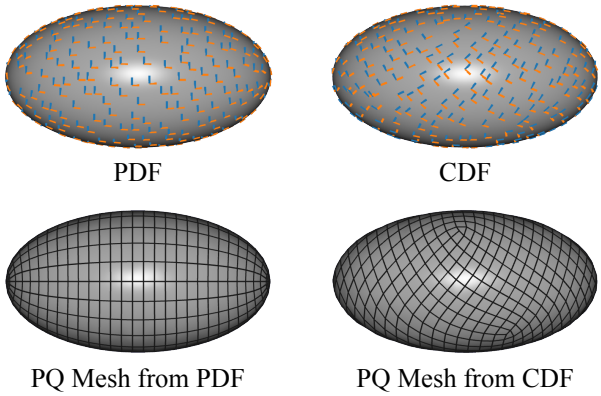

Figure 1: Visualization of direction fields and generated meshes on an ellipsoid.

The key advantages of our learning-based approach are twofold. First, by avoiding costly nonlinear optimization, it significantly reduces computation time for CDF generation. Second, by taking user strokes as input, it empowers designers with control over the CDF and, consequently, the PQ mesh structure. This is valuable for design exploration and for generating PQ meshes aligned with design intent. The rest of the subsections elaborate on our feature representation, network architecture, and loss functions, respectively.

4.2 Feature Representation

To accurately predict the CDF over the surface while respecting user-specified stroke constraints, we must carefully design the input features for our neural network. The most fundamental feature is the vertex position $\mathbf{p}_i \in \mathbb{R}^3$, which captures the geometry of the mesh.

Besides vertex positions, we also include vertex normals $\mathbf{n}_i \in \mathbb{R}^3$ as input features. Vertex normals encode local surface orientation and help the network better understand the shape of the surface in the neighborhood of each vertex. By providing both positions and normals, we give the network a more complete characterization of the local geometry.

To represent the user-specified stroke curves, we compute a vertex-wise stroke feature as follows. For each vertex $\mathbf{p}_i$, we find its closest point $\mathbf{p}_i^*$ among all stroke curve points. We then compute a projection vector $\mathbf{l}_i = \mathbf{p}_i^* - \mathbf{p}_i$ that connects the vertex to its closest point on the stroke curves (see Fig. 2a). This projection vector encodes the spatial relation-

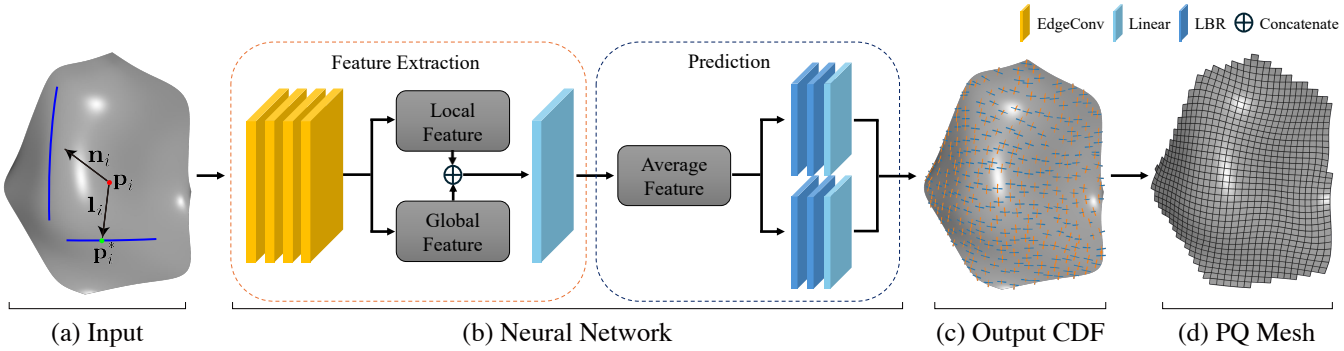

Figure 2: Our pipeline for inferring the conjugate direction field for PQ mesh generation. There are two core components in our learning model, including feature extraction and CDF prediction. In the first component, we take vertex positions, vertex normals, and the projected vectors to the user strokes as input, obtaining the latent feature for each vertex. In the second component, we use two MLPs to infer a CDF over the surface based on the latent features. In the figure, LBR means the combination of Linear, BatchNorm, and ReLU layers.

ship between the vertex and the stroke curves, providing a meaningful stroke representation at each vertex.

Our final input feature vector for each vertex is a concatenation of the vertex position $\mathbf{p}_i$, vertex normal $\mathbf{n}_i$, and stroke projection vector $\mathbf{l}_i$, resulting in a 9-dimensional feature vector. This compact yet informative feature representation enables our network to learn to generate CDFs that align with user strokes while adhering to the surface geometry.

4.3 Network Architecture

Our network architecture, illustrated in Fig. 2b, takes the 9-dimensional per-vertex feature vectors as input and predicts the CDF vectors $(\mathbf{u}_j, \mathbf{v}_j)$ on each face. The network consists of two main components: a feature extraction module and a direction field prediction module.

Feature Extraction Module Inspired by the method of (Dielen et al. 2021), our feature extraction module is designed to capture both local and global geometric information. However, instead of using separate networks for local and global features as in (Dielen et al. 2021), we employ a single unified architecture based on the Dynamic Graph CNN (DGCNN) (Wang et al. 2019)

DGCNN is a powerful graph neural network that operates directly on unstructured point clouds. It extracts features by applying EdgeConv operations on dynamic local graph neighborhoods, which are recomputed at each layer based on the learned point embeddings. This allows the network to learn both local and global shape properties adaptively. DGCNN has achieved strong performance across a range of 3D shape analysis tasks thanks to its ability to capture multi-scale geometric features.

Our network takes the vertex positions $\{\mathbf{p}_i\}$, vertex normals $\{\mathbf{n}_i\}$ and projection vectors $\{\mathbf{l}_i\}$ as input, and outputs a 256-dimensional latent feature for each vertex. Specifically, following DGCNN (Wang et al. 2019), we use a sequence of four EdgeConv layers to extract vertex features. However, instead of transforming the vertex features into a global shape feature by applying a fully-connected layer with ReLU activation (Nair and Hinton 2010) and a pooling strategy as in (Wang et al. 2019), we concatenate the local feature of each vertex with the global shape feature over the entire surface to construct an integrated feature. Finally, a fully-connected layer is applied to the integrated feature to obtain the final feature representation for each vertex. It is important to note that this feature integrates both global shape information and local geometry information.

Prediction Module The direction field prediction module takes the 256-dimensional per-vertex features as input and predicts the CDF vectors on $(\mathbf{u}_j, \mathbf{v}_j)$ on each triangle face. To obtain face features from the vertex features, we simply average the vertex features for each triangle.

We use two separate MLPs to predict the two families of direction vectors $\{\mathbf{u}_j\}$ and $\{\mathbf{v}_j\}$ independently. Each network consists of three layers that map the input 256-dimensional face feature to intermediate representations of dimensions 128 and 64, and finally output a 3-dimensional vector representing either $\mathbf{u}_j$ or $\mathbf{v}_j$. The first two layers use BatchNorm (Ioffe and Szegedy 2015) and ReLU (Nair and Hinton 2010) as the activation function. Since the last layer directly creates the directional vectors, we remove any BatchNorm and activation function in that layer. Finally, we normalize each predicted vector to unit length.

4.4 Loss Functions

We employ a combination of loss terms to train our network to generate smooth, locally conjugate direction fields that align with the user-specified strokes.

Direction Alignment Our first requirement is to align the predicted direction field $\{(\mathbf{u}_j, \mathbf{v}_j)\}$ with the ground truth directions $\{(\mathbf{u}_j^*, \mathbf{v}_j^*)\}$. Different from direct computation of the cosine similarity between two fields to measure alignment, as performed in (Dielen et al. 2021), we use the following function to measure the alignment between $\mathbf{u}_j$ and $\mathbf{u}_j^*$ and between $\mathbf{v}_j$ and $\mathbf{v}_j^*$:

$$E_j = (\mathbf{u}_j \cdot \mathbf{u}_j^{*\perp})^2 + (\mathbf{v}_j \cdot \mathbf{v}_j^{*\perp})^2. \tag{3}$$

Here $\mathbf{u}_j^{*\perp}$, $\mathbf{v}_j^{*\perp}$ are 90-degree rotations of the ground-truth unit directions $\mathbf{u}_i^*$, $\mathbf{v}_i^*$ about the face normal $\mathbf{n}_j$.

Note that our use of rotated ground truth vectors to measure alignment enables us to handle the sign ambiguity for the ground-truth vectors, i.e., it allows the $\mathbf{u}_j$ to be close to either $\mathbf{u}_j^{*\perp}$ or $-\mathbf{u}_j^{*\perp}$ (and the same for $\mathbf{v}_j$ and $\mathbf{v}_j^{*\perp}$). This is beneficial as both configurations are feasible for the subsequent parameterization and quad mesh generation.

However, for more effective learning, we shall not prescribe specific correspondence between the predicted direction vectors $(\mathbf{u}_j, \mathbf{v}_j)$ and the ground truth directions $(\mathbf{u}_j^*, \mathbf{v}_j^*)$. Instead, we only need the predicted pair to align well with the ground truth pair as a whole, regardless of which predicted vector corresponds to which ground truth vector. To handle this ambiguity, we also measure the alignment for the other correspondence with the following term:

$$E_j' = (\mathbf{u}_j \cdot \mathbf{v}_j^{*\perp})^2 + (\mathbf{v}_j \cdot \mathbf{u}_j^{*\perp})^2. \tag{4}$$

We then use the minimum value between E_j and E_j' to indicate the alignment between the vector pairs $(\mathbf{u}_j, \mathbf{v}_j)$ and $(\mathbf{u}_j^*, \mathbf{v}_j^*)$ as a whole, and introduce the following direction alignment term to measure the alignment between the predicted direction field with the ground truth:

$$\mathcal{L}_d = \frac{1}{m} \sum\nolimits_{j=1}^{m} \min(E_j, E_j'), \tag{5}$$

where m is the number of mesh faces.

Direction Consistency with Normals The output direction field $\{(\mathbf{u}_j, \mathbf{v}_j)\}$ should be orthogonal to the corresponding unit face normals $\{\mathbf{n}_j\}$. This can be enforced using the following loss term:

$$\mathcal{L}_{dn} = \frac{1}{m} \sum\nolimits_{j=1}^{m} (\mathbf{u}_j \cdot \mathbf{n}_j)^2 + (\mathbf{v}_j \cdot \mathbf{n}_j)^2. \tag{6}$$

Direction Smoothness A smooth CDF is essential to avoid unnecessary singularities and ensure a coherent layout for the final PQ mesh. For a pair of direction vectors $(\mathbf{u}_j, \mathbf{v}_j)$ on face f_j, and another pair $(\mathbf{u}_k, \mathbf{v}_k)$ on a neighboring face f_k that share a common edge with f_j, the local CDF smoothness between f_j and f_k can be defined as (Jakob et al. 2015):

$$E_{jk} = (\hat{\mathbf{u}}_j \cdot \mathbf{u}_k^{\perp})^2 + (\hat{\mathbf{v}}_j \cdot \mathbf{v}_k^{\perp})^2, \tag{7}$$

where $\hat{\mathbf{u}}_j$, $\hat{\mathbf{v}}_j$ are the vectors obtained by parallel transporting $\mathbf{u}_j$, $\mathbf{v}_j$ from face f_i to face f_k, respectively (Jakob et al. 2015). $\mathbf{u}_k^{\perp}$, $\mathbf{v}_k^{\perp}$ are the 90-degree rotations of $\mathbf{u}_k$, $\mathbf{v}_k$ about the face normal $\mathbf{n}_k$. Here we measure the direction alignment in the same way as Eq. 3.

As discussed in Sec. 4.4, the correspondence between direction vectors $(\mathbf{u}_j, \mathbf{v}_j)$ and $(\mathbf{u}_k, \mathbf{v}_k)$ on neighboring faces is ambiguous. Therefore, we also define the smoothness measure for the other correspondence as follows:

$$E_{jk}' = (\hat{\mathbf{u}}_j \cdot \mathbf{v}_k^{\perp})^2 + (\hat{\mathbf{v}}_j \cdot \mathbf{u}_k^{\perp})^2. \tag{8}$$

Finally, the overall direction smoothness of the CDF on the entire mesh is defined as:

$$\mathcal{L}_{ds} = \frac{1}{|\mathcal{N}|} \sum\nolimits_{(j,k)\in\mathcal{N}} \min(E_{jk}, E_{jk}'), \tag{9}$$

where $\mathcal{N}$ is the index set for neighboring faces.

Direction Consistency with Strokes To ensure that the inferred CDF closely aligns with the user-specified strokes, we introduce an additional consistency loss between the output direction vectors and the strokes. For a pair of predicted directions $(\mathbf{u}_k, \mathbf{v}_k)$ on face f_k which contains a segment $\mathbf{s}_k$ (represented as a vector) of a stroke $\mathbf{S}_i$, one of these two directions should align with the segment. Following the alignment measurement in Eq. 3, this can be expressed as:

$$D_k = \min((\mathbf{u}_k \cdot \mathbf{s}_k^{\perp})^2, (\mathbf{v}_k \cdot \mathbf{s}_k^{\perp})^2) / \|\mathbf{s}_k\|, \tag{10}$$

where $\mathbf{s}_k^{\perp}$ is 90-degree rotation of the stroke segment $\mathbf{s}_k$ about the face normal $\mathbf{n}_k$. The overall direction consistency with strokes can be written as:

$$\mathcal{L}_{dc} = \frac{1}{|\mathcal{S}|} \sum\nolimits_{\mathbf{S}_i\in\mathcal{S}} \frac{1}{|\mathcal{T}_i|} \sum\nolimits_{k\in\mathcal{T}_i} D_k, \tag{11}$$

where $\mathcal{T}_i$ denotes the index set of faces that contain stroke segments of $\mathbf{S}_i$.

Field Regularization To avoid obtaining a trivial zero direction field, we also introduce a regularization loss:

$$\mathcal{L}_{fr} = \frac{1}{m} \sum\nolimits_{j=1}^{m} (\|\mathbf{u}_j\| - 1)^2 + (\|\mathbf{v}_j\| - 1)^2. \tag{12}$$

Total Loss Overall the full loss for our model is given by:

$$\mathcal{L}_{total} = \mathcal{L}_d + \lambda_1\mathcal{L}_{dn} + \lambda_2\mathcal{L}_{ds} + \lambda_3\mathcal{L}_{dc} + \lambda_4\mathcal{L}_{fr}, \tag{13}$$

where $\lambda_1, \lambda_2, \lambda_3, \lambda_4$ are the loss weights.

5 Dataset

To facilitate the development of learning-based approaches for CDF generation on 3D surfaces, we construct a new dataset consisting of freeform triangle mesh surfaces, ground truth CDFs, and corresponding guiding strokes.

5.1 Freeform Surface Generation

Following the approach in (Deng et al. 2023), we generate freeform shapes using B-splines. The control points of the B-spline surfaces are distributed such that the resulting shapes resemble height-field-like patches. We also apply random 3D deformations to introduce more shape variations. B-splines are advantageous for representing smooth freeform shapes, particularly architectural surfaces, and allow for easy shape control through manipulating control points.

After obtaining the B-spline surface, we further construct a triangle mesh as its discrete counterpart by evenly sampling 2601 points with 5000 faces. Our dataset comprises 50000, 2500, and 300 freeform triangle mesh surfaces for training, validation, and testing, respectively.

5.2 Data Normalization

To facilitate the learning process, we normalize the position, orientation, and size of each triangle mesh as follows. Given the vertex coordinates $\mathcal{V} = \{\mathbf{p}_i \in \mathbb{R}^3\}$ of the surface mesh $\mathcal{M} = \{\mathcal{V}, \mathcal{F}\}$, we perform Principal Component Analysis (PCA) to estimate the three principal directions $\mathbf{d}_1$, $\mathbf{d}_2$, $\mathbf{d}_3$ based on their corresponding eigenvalues in descending order. We then rotate the mesh to align the principal directions with the global x-, y-, and z-axes, respectively. Finally, we translate and scale the mesh to fit within a unit sphere.

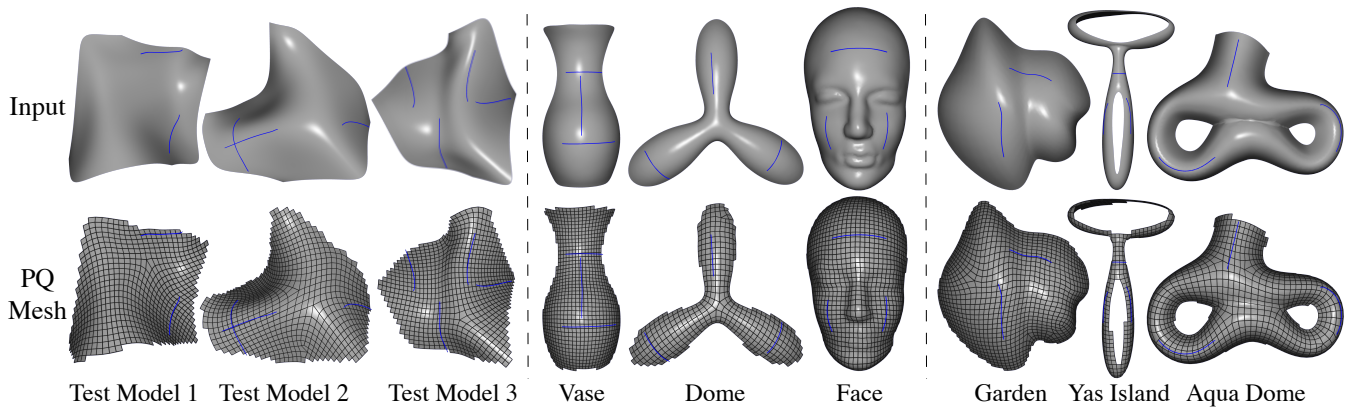

Figure 3: PQ meshes generated using conjugate direction field inferred by our network on different types of surfaces as input. The blue lines on the surface are input strokes, which are streamlines traced from ground-truth CDF for comparison purposes. Left: B-spline surfaces from our test set; Middle: Open-boundary surfaces; Right: Real architectural surfaces. The input triangle meshes and strokes, as well as the output PQ meshes, are provided in the supplementary code and data.

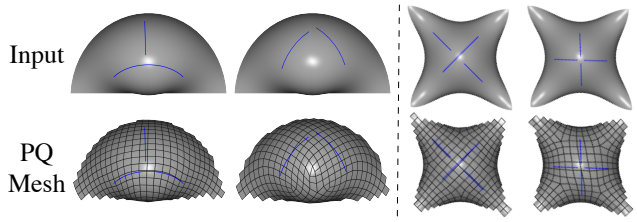

Figure 4: The examples where the different PQ mesh layouts are generated on the same surface with two different types of strokes as inputs.

5.3 CDF Generation and Stroke Mimicking

To obtain diverse ground truth CDFs while simulating user-specified strokes as design guidance, we first randomly sample 1 to 5 anchor points on each triangle mesh. Then we initialize conjugate direction vectors at each anchor point by randomly assigning one direction and computing its conjugate. Subsequently, we apply the CDF optimization method from (Diamanti et al. 2014), as implemented in (Vaxman et al. 2022), to generate a dense ground truth CDF subject to the direction constraints at the anchor points. To mimic user-defined strokes in practical design scenarios, we trace streamlines of the ground truth CDF starting from the anchor points. The visualization of some representative results is presented in the supplementary materials.

6 Experiments

In our experiments, we train the network described in Sec. 4.3 for 200 epochs, using 50000 training examples from the dataset generated using the approach described in Sec. 5. Throughout the training phase, we utilize the Adam optimizer (Kingma and Ba 2014) with a fixed learning rate of 1.0×10^{-4}. For the loss weights, we set $\lambda_1 = \lambda_2 = \lambda_3 = \lambda_4 = 1.0$. To validate the effectiveness and efficiency of our work, we conduct a set of evaluations on a desktop PC with a 24-core Intel Core i9-14900K at 3.20GHz, and an NVIDIA GeForce RTX 4090 GPU with 24GB of video memory.

Model	#Faces	Optimization	Ours
Test Model 1	5000	2.851s	0.200s (14.3×)
Test Model 2	5000	2.855s	0.194s (14.7×)
Test Model 3	5000	2.858s	0.195s (14.6×)
Vase	23642	17.326s	0.254s (68.2×)
Dome	44490	30.198s	0.417s (72.4×)
Face	60077	40.412s	0.571s (70.8×)
Garden	8322	4.946s	0.206s (24.0×)
Yas Island	7029	3.766s	0.204s (18.5×)
Aqua Dome	10790	6.522s	0.217s (30.1×)

Table 1: Time cost comparison for CDF generation between the method in (Diamanti et al. 2014) and our method on the surfaces shown in Fig. 3.

6.1 Results

We test our method on a diverse set of surfaces, including B-spline surfaces from our test set, general open-boundary surfaces, and real architectural surfaces. Fig. 3 demonstrates the PQ meshes generated using our method on these different surface types. Moreover, benefiting from the efficient CDF generation under user-specified stroke configurations, our method allows easy exploration of different PQ mesh layouts based on the same input shape, as shown in Fig. 4. More results on general models such as Stanford Bunny can be found in the supplementary materials.

6.2 Computational Efficiency

We also validate the computational efficiency of our learning-based CDF generation method by comparing with the traditional optimization-based method (Diamanti et al. 2014), which we also employed in our dataset generation pipeline (Section 5.3) and thus serves as the most direct baseline for comparison Tab. 1 shows the time differences on examples provided in Fig. 3. Owing to the efficiency of

Model	η_{mean}		η_{max}		δ	θ
	before	after	before	after		
Test Model 1	0.0062	0.0024	0.0425	0.0096	5.61°	8.96°
Test Model 2	0.0060	0.0025	0.0483	0.0123	10.76°	11.08°
Test Model 3	0.0065	0.0030	0.0566	0.0136	5.14°	8.99°
Vase	0.0035	0.0015	0.0259	0.0058	5.32°	9.02°
Dome	0.0108	0.0036	0.0399	0.0127	8.21°	12.04°
Face	0.0162	0.0037	0.2098	0.0253	6.82°	19.74°
Garden	0.0082	0.0028	0.1015	0.0168	7.21°	23.88°
Yas Island	0.0151	0.0017	0.1845	0.0110	16.10°	22.55°
Aqua Dome	0.0140	0.0032	0.4529	0.0203	17.70°	14.39°
Test Set	0.0067	0.0023	0.0591	0.0118	8.31°	11.30°

Table 2: Quantitative measurements on the PQ meshes shown in Fig. 3. The average measurements on our test set are shown in the last row. The planarity before and after applying vertex perturbation optimization (Deuss et al. 2015) are both listed. δ, θ are measured in degrees.

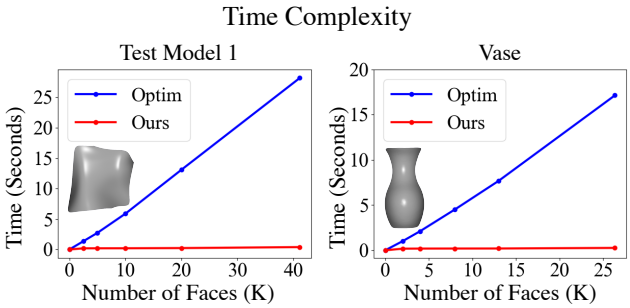

Figure 5: The time complexity illustration of the CDF generation between our method and the traditional optimization-based method (Diamanti et al. 2014).

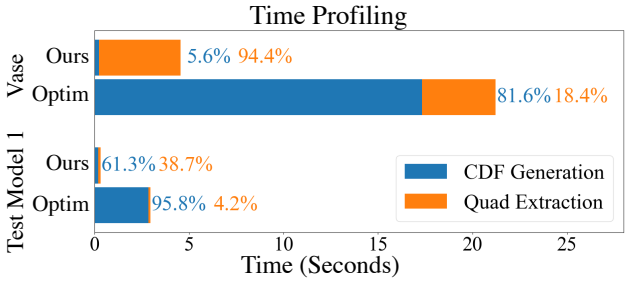

Figure 6: Time profiling of CDF generation and quad mesh extraction on our method and optimization method (Diamanti et al. 2014). The percentages are shown to the right.

the parallel inferences of the learned model, our method can generate CDF much faster (generally over an order of magnitude, and even close to two orders of magnitude for those cases with a larger number of faces). This feature particularly favors the real-world application that requires an iterative CDF design process.

Additionally, Fig. 5 illustrates the time complexity of our learning-based method over the traditional optimization-based method. For two test shapes with increased face numbers, we can see a clear growth (approximately linear) of the optimization cost for CDF generation. In contrast, due to the highly parallelized feedforward process of our learning-based approach, the time cost increases only marginally. Fig. 6 compares the cost of CDF generation and quad mesh extraction for two test models shown in Fig. 3. We can see that for the optimization approach, CDF generation (even one round) takes the majority of the computational cost, and becomes a major bottleneck in the quad mesh design pipeline with iterative CDF attempts, especially for a bigger reference mesh such as Vase with ∼20k faces. On the contrary, our method can effectively address this issue and significantly reduce the time consumption, enabling a controllable and interactive CDF generation practice for user-friendly architectural design (see the supplementary video).

6.3 Result Quality

To evaluate the effectiveness of our method, we define a set of quantitative metrics to demonstrate the quality of our results. First, we utilize the planarity of the quad mesh after quadrangulation to reflect the quality of the inferred CDF. For each quad face f_j, we compute the distance between two diagonals and divide it by the average edge length, resulting in η_j, where $\eta_j = 0$ means f_j is fully planar. Then we obtain the maximum planarity error η_{max} and the mean planarity error η_{mean} for the entire quad mesh. We also evaluate the inferred CDF by directly measuring its consistency with the input strokes and closeness with the ground-truth CDF. To make the metrics more geometrically meaningful, we use the angle between directions to measure the direction alignment instead of computing their inner-product as in Sec. 4.4. For the stroke consistency, we define δ as the average angle between the inferred CDF and the strokes, where the directions are restricted on those mesh faces containing stroke segments. For CDF closeness, we define θ as the average angle between directions of the inferred CDF and the ground-truth CDF over all mesh faces.

Tab. 2 shows the quantitative evaluations on different models visualized in Fig. 3, as well as the overall performance on our test set with 300 models. The results show that our inferred CDFs are of high quality for guiding the

	# of Sings	δ	θ
Without $\mathcal{L}_{ds}$	7.02	7.38°	10.55°
Without $\mathcal{L}_{dc}$	4.67	10.32°	11.48°
PCT (Guo et al. 2021)	8.97	20.98°	19.06°
Ours	4.91	8.31°	11.30°

Table 3: Quantitative measurements of ablation studies. For the inferred CDFs on the test set, we measure the average number of singularities, δ - the consistency with the input strokes, and θ - the closeness to the ground truth CDF. The ablated methods are without the direction smoothness loss $\mathcal{L}_{ds}$, without stroke consistency loss $\mathcal{L}_{dc}$, and without our context-aware stroke feature but using PCT (Guo et al. 2021) for stroke feature extraction, respectively.

quadrangulation, as the initial quad mesh layouts already exhibit good planarity. Also, the initial quad meshes serve as a good starting point for vertex perturbation optimization (Deuss et al. 2015) where the planarity can be further improved. The CDF closeness and consistency values also demonstrate that our method can directly infer CDFs adhering to the ground truth while respecting the stroke guidance.

6.4 Ablation Study

To validate the design choices of our method, we perform an ablation study on the loss functions defined in Sec. 4.4. Note that the normal consistency loss $\mathcal{L}_{dn}$ is necessary for the inferred CDF directions to lie on the local tangent planes. Therefore, we only ablate the loss functions $\mathcal{L}_{ds}$ and $\mathcal{L}_{dc}$. We measure the quality of the resulting CDF using δ and θ as defined in the previous section, as well as the number of singularities of the CDF. The comparison of the full method and the ablated methods are shown in Tab. 3. We can see that with the help of loss function $\mathcal{L}_{ds}$, our result can produce far fewer singularities in the resulting PQ meshes. Furthermore, we can also generate a CDF with much better stroke consistency by adding loss $\mathcal{L}_{dc}$. Some qualitative results are presented in the supplementary materials.

In addition, to evaluate the effectiveness of the feature representation of strokes described in Sec. 4.2, we test an alternative approach where the strokes are simply treated as sequential data by themselves, and the feature is extracted from a Point Cloud Transformer (PCT) (Guo et al. 2021). As shown in Tab. 3, our method performs much better in all metrics, which verifies the superiority of our novel feature representation over PCT for surface strokes, as the surface context is also addressed in our representation.

6.5 Comparison with Other Methods

We also compare with recent learning-based field generation methods (Gao et al. 2024; Dong et al. 2025) by contacting the authors and running their codes. The results are shown in Fig. 7. VectorHeat (Gao et al. 2024) aims to generate rotation-invariant vector/cross fields. There is no guarantee that the generated cross field is a CDF, except for PDF which is a special type of CDF. Therefore, it cannot be used for PQ mesh optimization. NeurCross (Dong et al. 2025) is restricted to PDF generation, thus cannot be used for controllable CDF generation as our method. In contrast, our method is capable of producing flexible CDF variations (and their corresponding PQ meshes) that align with the input strokes.

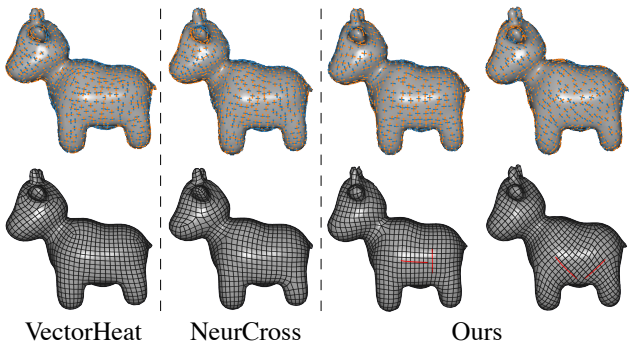

Figure 7: Comparison with recent learning-based field generation methods VectorHeat (Gao et al. 2024) and NeurCross (Dong et al. 2025). The generated direction field overlaid on the input reference surface (top row) and the resultant quad meshes (bottom row) are shown for each method. For clarity, the input strokes of our method (highlighted in red) are visualized on the final meshes.

7 Conclusion

In this paper, we propose a novel learning-based method to infer conjugate direction fields for PQ mesh generation, avoiding complicated non-linear optimization. Moreover, we design a unique feature representation for user-specified strokes that can be integrated with other geometry information for effective local and global feature encoding. We also present a synthetic dataset with a large number of freeform surfaces with ground truth CDFs and their guiding strokes, enabling the development of learning-based approaches. Our work not only offers a much more efficient framework for CDF generation, but also contributes to intuitive control of the PQ mesh layout with user-specified strokes, largely benefiting interactive design applications in practice.

In the future, we would like to make improvements on surfaces with sharp features by introducing additional strokes along sharp edges. Besides, while the direction smoothness loss term $\mathcal{L}_{ds}$ can eliminate or reduce the singularities on the output PQ mesh, there is no explicit control of the number and location of singularities, which is worth exploration. Finally, we plan to investigate the unsupervised approach for CDF generation with better generalization capabilities.

Acknowledgments

We thank Zhi Deng for assistance with dataset preparation and Cherril Pope for proofreading the manuscript. This work was supported by China Scholarship Council under Grant No. 202108340019.

References

Alliez, P.; Cohen-Steiner, D.; Devillers, O.; Lévy, B.; and Desbrun, M. 2003. Anisotropic polygonal remeshing. *ACM Trans. Graph.*, 22(3): 485–493.

Bobenko, A. I.; and Suris, Y. B. 2008. *Discrete Differential Geometry*. American Mathematical Society.

Bommes, D.; Lévy, B.; Pietroni, N.; Puppo, E.; Silva, C.; Tarini, M.; and Zorin, D. 2013. Quad-Mesh Generation and Processing: A Survey. *Computer Graphics Forum*, 32(6): 51–76.

Bommes, D.; Zimmer, H.; and Kobbelt, L. 2009. Mixed-integer quadrangulation. *ACM Trans. Graph.*, 28(3): 1–10.

Bronstein, M. M.; Bruna, J.; LeCun, Y.; Szlam, A.; and Vandergheynst, P. 2017. Geometric Deep Learning: Going beyond Euclidean Data. *IEEE Signal Processing Magazine*, 34(4): 18–42.

Charles, R. Q.; Su, H.; Kaichun, M.; and Guibas, L. J. 2017. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 77–85.

Chen, Y.-C.; Kim, V.; Aigerman, N.; and Jacobson, A. 2023. Neural Progressive Meshes. In *ACM SIGGRAPH 2023 Conference Proceedings*, SIGGRAPH '23, 1–9.

Deng, Z.; Liu, Y.; Pan, H.; Jabi, W.; Zhang, J.; and Deng, B. 2023. Sketch2PQ: Freeform Planar Quadrilateral Mesh Design via a Single Sketch. *IEEE Transactions on Visualization & Computer Graphics*, 29(09): 3826–3839.

Deuss, M.; Deleuran, A. H.; Bouaziz, S.; Deng, B.; Piker, D.; and Pauly, M. 2015. ShapeOp-A Robust and Extensible Geometric Modelling Paradigm. In *Modelling Behaviour: Design Modelling Symposium 2015*, 505–515. Springer International Publishing.

Diamanti, O.; Vaxman, A.; Panozzo, D.; and Sorkine-Hornung, O. 2014. Designing N-PolyVector fields with complex polynomials. *Computer Graphics Forum*, 33(5): 1–11.

Dielen, A.; Lim, I.; Lyon, M.; and Kobbelt, L. 2021. Learning Direction Fields for Quad Mesh Generation. *Computer Graphics Forum*, 40(5): 181–191.

Dong, Q.; Wen, H.; Xu, R.; Chen, S.; Zhou, J.; Xin, S.; Tu, C.; Komura, T.; and Wang, W. 2025. NeurCross: A Neural Approach to Computing Cross Fields for Quad Mesh Generation. *ACM Trans. Graph.*, 44(4): 1–17.

Dong, S.; Bremer, P.-T.; Garland, M.; Pascucci, V.; and Hart, J. C. 2006. Spectral surface quadrangulation. *ACM Trans. Graph.*, 25(3): 1057–1066.

Dong, S.; Kircher, S.; and Garland, M. 2005. Harmonic functions for quadrilateral remeshing of arbitrary manifolds. *Computer Aided Geometric Design*, 22(5): 392–423.

Ebke, H.-C.; Bommes, D.; Campen, M.; and Kobbelt, L. 2013. QEx: Robust quad mesh extraction. *ACM Trans. Graph.*, 32(6): 1–10.

Fang, X.; Bao, H.; Tong, Y.; Desbrun, M.; and Huang, J. 2018. Quadrangulation through morse-parameterization hybridization. *ACM Trans. Graph.*, 37(4): 1–15.

Gao, A.; Chu, M.; Kapadia, M.; Lin, M. C.; and Liu, H.-T. D. 2024. An Intrinsic Vector Heat Network. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, 14638–14650. PMLR.

Girard, N.; Smirnov, D.; Solomon, J.; and Tarabalka, Y. 2020. Regularized building segmentation by frame field learning. In *IGARSS 2020-2020 IEEE International Geoscience and Remote Sensing Symposium*, 1805–1808.

Glymph, J.; Shelden, D.; Ceccato, C.; Mussel, J.; and Schober, H. 2004. A parametric strategy for free-form glass structures using quadrilateral planar facets. *Automation in Construction*, 13(2): 187–202.

Guo, M.-H.; Cai, J.-X.; Liu, Z.-N.; Mu, T.-J.; Martin, R. R.; and Hu, S.-M. 2021. PCT: Point cloud transformer. *Computational Visual Media*, 7(2): 187–199.

Guo, Y.; Wang, H.; Hu, Q.; Liu, H.; Liu, L.; and Bennamoun, M. 2020. Deep Learning for 3D Point Clouds: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(12): 4338–4364.

Huang, J.; Zhang, M.; Ma, J.; Liu, X.; Kobbelt, L.; and Bao, H. 2008. Spectral quadrangulation with orientation and alignment control. *ACM Trans. Graph.*, 27(5): 1–9.

Hui, K.-H.; Li, R.; Hu, J.; and Fu, C.-W. 2022. Neural Wavelet-domain Diffusion for 3D Shape Generation. In *SIGGRAPH Asia 2022 Conference Papers*, SA '22, 1–9.

Ioffe, S.; and Szegedy, C. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, 448–456. PMLR.

Jakob, W.; Tarini, M.; Panozzo, D.; Sorkine-Hornung, O.; et al. 2015. Instant field-aligned meshes. *ACM Trans. Graph.*, 34(6): 1–15.

Kaelberer, F.; Nieser, M.; and Polthier, K. 2007. QuadCover - Surface Parameterization using Branched Coverings. *Computer Graphics Forum*, 26(3): 375–384.

Kingma, D. P.; and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Liu, Y.; Pottmann, H.; Wallner, J.; Yang, Y.-L.; and Wang, W. 2006. Geometric modeling with conical meshes and developable surfaces. *ACM Trans. Graph.*, 25(3): 681–689.

Liu, Y.; Xu, W.; Wang, J.; Zhu, L.; Guo, B.; Chen, F.; and Wang, G. 2011. General planar quadrilateral mesh design using conjugate direction field. *ACM Trans. Graph.*, 30(6): 1–10.

Lyon, M.; Campen, M.; Bommes, D.; and Kobbelt, L. 2019. Parametrization quantization with free boundaries for trimmed quad meshing. *ACM Trans. Graph.*, 38(4): 1–14.

Nair, V.; and Hinton, G. E. 2010. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning*, ICML'10, 807–814. Omnipress.

Potamias, R. A.; Ploumpis, S.; and Zafeiriou, S. 2022. Neural Mesh Simplification. In *2022 IEEE/CVF Conference on*

Computer Vision and Pattern Recognition (CVPR), 18562–18571.

Pottmann, H. 2013. Architectural Geometry and Fabrication-Aware Design. *Nexus Network Journal*, 15(2): 195–208.

Pottmann, H.; Eigensatz, M.; Vaxman, A.; and Wallner, J. 2015. Architectural geometry. *Computers & Graphics*, 47: 145–164.

Pottmann, H.; Liu, Y.; Wallner, J.; Bobenko, A.; and Wang, W. 2007. Geometry of multi-layer freeform structures for architecture. *ACM Trans. Graph.*, 26(3): 65–es.

Rakotosaona, M.-J.; Aigerman, N.; Mitra, N. J.; Ovsjanikov, M.; and Guerrero, P. 2021. Differentiable surface triangulation. *ACM Trans. Graph.*, 40(6): 1–13.

Ray, N.; Li, W. C.; Lévy, B.; Sheffer, A.; and Alliez, P. 2006. Periodic global parameterization. *ACM Trans. Graph.*, 25(4): 1460–1485.

Sharp, N.; and Ovsjanikov, M. 2020. PointTriNet: Learned Triangulation of 3D Point Sets. In *Computer Vision – ECCV 2020*, 762–778. Springer International Publishing.

Tong, Y.; Alliez, P.; Cohen-Steiner, D.; and Desbrun, M. 2006. Designing quadrangulations with discrete harmonic forms. In *Proceedings of the Fourth Eurographics Symposium on Geometry Processing*, SGP '06, 201–210. Eurographics Association.

Vaxman, A.; et al. 2022. Directional: A library for Directional Field Synthesis, Design, and Processing.

Wang, Y.; Sun, Y.; Liu, Z.; Sarma, S. E.; Bronstein, M. M.; and Solomon, J. M. 2019. Dynamic graph cnn for learning on point clouds. *ACM Trans. Graph.*, 38(5): 1–12.

Wei, G.; Pan, H.; Zhuang, S.; Zhou, Y.; and Li, C. 2024. iPUNet: Iterative Cross Field Guided Point Cloud Upsampling. *IEEE Transactions on Visualization and Computer Graphics*, 30(9): 6089–6103.

Xiao, Y.-P.; Lai, Y.-K.; Zhang, F.-L.; Li, C.; and Gao, L. 2020. A survey on deep geometry learning: From a representation perspective. *Computational Visual Media*, 6(2): 113–133.

Xu, Q.-C.; Mu, T.-J.; and Yang, Y.-L. 2023. A survey of deep learning-based 3D shape generation. *Computational Visual Media*, 9(3): 407–442.

Zadravec, M.; Schiftner, A.; and Wallner, J. 2010. Designing Quad-dominant Meshes with Planar Faces. *Computer Graphics Forum*, 29(5): 1671–1679.

Zhang, B.; Tang, J.; Niessner, M.; and Wonka, P. 2023. 3DShape2VecSet: A 3D shape representation for neural fields and generative diffusion models. *ACM Trans. Graph.*, 42(4): 1–16.