NeurCross: A Neural Approach to Computing Cross Fields for Quad Mesh Generation

QIUJIE DONG, Shandong University, China, The University of Hong Kong, China, and TransGP, China
HUIBIAO WEN, Shandong University, China
RUI XU, The University of Hong Kong, China
SHUANGMIN CHEN, Qingdao University of Science and Technology, China
JIARAN ZHOU, Ocean University of China, China
SHIQING XIN*, Shandong University, China
CHANGHE TU, Shandong University, China
TAKU KOMURA, The University of Hong Kong, China
WENPING WANG, Texas A&M University, United States of America

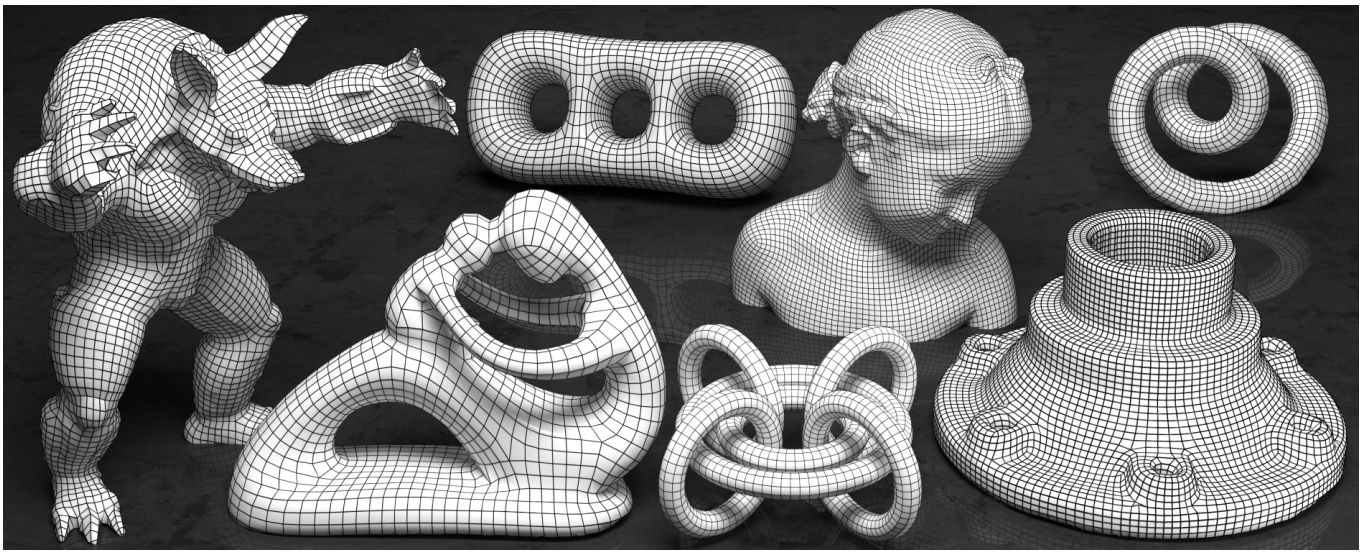


Fig. 1. Gallery of quad meshes generated with our NeurCros method. NeurCross excels in computing cross field for generating high-quality quad meshes. Its advantages include optimized singular point placement, insensitivity to surface noise and minor surface undulations, and faithful alignment with principal curvature directions and sharp feature curves.

Quadrilateral mesh generation plays a crucial role in numerical simulations within Computer-Aided Design and Engineering (CAD/E). Producing high-quality quadrangulation typically requires satisfying four key criteria. First, the quadrilateral mesh should closely align with principal curvature directions. Second, singular points should be strategically placed and effectively minimized. Third, the mesh should accurately conform to sharp feature edges. Lastly, quadrangulation results should exhibit robustness against noise and minor geometric variations. Existing methods generally involve first computing a regular cross field to represent quad element orientations across the surface, followed by extracting a quadrilateral mesh aligned closely with this cross field. A primary challenge with this approach is balancing the smoothness of the cross field with its alignment to pre-computed principal curvature directions, which are sensitive to small surface perturbations and often ill-defined in spherical or planar regions.

To tackle this challenge, we propose *NeurCross*, a novel framework that simultaneously optimizes a cross field and a neural signed distance function (SDF), whose zero-level set serves as a proxy of the input shape. Our joint optimization is guided by three factors: faithful approximation of the optimized SDF surface to the input surface, alignment between the cross field and the principal curvature field derived from the SDF surface, and smoothness of the cross field. Acting as an intermediary, the neural SDF contributes in two essential ways. First, it provides an alternative, optimizable base surface exhibiting more regular principal curvature directions for guiding the cross field. Second, we leverage the Hessian matrix of the neural SDF to implicitly enforce cross field alignment with principal curvature directions, thus eliminating the need for explicit curvature extraction. Extensive experiments

*Corresponding author: Shiqing Xin.

Authors' addresses: Qiujie Dong, Shandong University, Qingdao, Shandong, China and The University of Hong Kong, Hong Kong, China and TransGP, Hong Kong, China, qiujie.jay.dong@gmail.com; Huibiao Wen, Shandong University, Qingdao, Shandong, China, ericvein@163.com; Rui Xu, The University of Hong Kong, Hong Kong, China, xrvitd@163.com; Shuangmin Chen, Qingdao University of Science and Technology, Qingdao, Shandong, China, csmqq@163.com; Jiaran Zhou, Ocean University of China, Qingdao, Shandong, China, zhoujiaran@ouc.edu.cn; Shiqing Xin, Shandong University, Qingdao, Shandong, China, xinshiqing@sdu.edu.cn; Changhe Tu, Shandong University, Qingdao, Shandong, China, chtu@sdu.edu.cn; Taku Komura, The University of Hong Kong, Hong Kong, China, taku@cs.hku.hk; Wenping Wang, Texas A&M University, Texas, United States of America, wenping@tamu.edu.

demonstrate that NeurCross outperforms the state-of-the-art methods in terms of singular point placement, robustness against surface noise and surface undulations, and alignment with principal curvature directions and sharp feature curves.

CCS Concepts: • **Computing methodologies** → **Shape analysis**; **Mesh geometry models**.

Additional Key Words and Phrases: quadrangulation, neural network, cross field, signed distance function, principal curvature

1 INTRODUCTION

Quadrangulation is fundamental in both Computer-Aided Design (CAD) and Computer-Aided Engineering (CAE) [Bommes et al. 2013b; Vaxman et al. 2016], with significant applications in finite element analysis, isogeometric analysis, character animation, and physics simulations [Bommes et al. 2013a, 2009; Campen et al. 2015a; Jakob et al. 2015; Mu et al. 2023].

Existing approaches typically first compute a reliable cross field to represent quad element orientations across the surface, followed by extracting quad meshes aligned closely with the computed field [Bommes et al. 2013a, 2009; Huang et al. 2018; Jakob et al. 2015; Zhang et al. 2020]. Most methods require principal curvature directions as input. However, computing a desired cross field from principal curvature directions entails meeting four key requirements: First, the quadrilateral mesh should align closely with principal curvature directions. Second, singular points should be strategically placed and minimized. Third, the mesh should conform accurately to sharp feature edges. Lastly, quadrangulation results should be robust against noise and minor surface variations. These challenges are particularly pronounced in geometrically or topologically complex shapes.

Fig. 2 shows quadrangulation results of some existing methods. As shown, for instance, QuadWild [Pietroni et al. 2021] fails to align properly with principal curvature directions due to an overemphasis on the smoothness of the cross field. Although principal curvature directions provide useful geometric clues, precisely controlling their influence on the inferred cross field is difficult, especially in nearly spherical or planar regions, or on a surface with small undulations, where principal curvature directions become unstable.

We introduce an optimizable neural signed distance function (SDF) as the underlying shape representation to infer the desired cross field. The neural SDF serves as a proxy for the input shape, which often exhibits unstable principal curvature directions. Optimizing this SDF alongside the cross field provides a smooth approximation to the input shape, generating a regular principal curvature field to guide cross field generation. More specifically, our joint optimization is guided by three factors: faithful approximation of the optimized SDF surface to the input surface, alignment between the cross field and the principal curvature field derived from the SDF surface, and smoothness of the cross field. We integrate these requirements into a unified neural optimization framework, called *NeurCross*, that enables simultaneous optimization of the SDF and cross field. Fig. 3 illustrates our method's success on a dimpled ellipsoid with irregular curvature directions, compared to a naïve two-stage approach that first optimizes an SDF to properly fit the input shape and then uses the curvature field of this fixed SDF to guide the generation of the cross field. The key to the success of our method is its simultaneous optimization strategy that allows the cross field smoothness term to inform the optimal shape of the neural SDF as a proxy surface.

Additionally, a key advantage is that the SDF-based shape operator implicitly encodes principal curvature directions, enabling enforcement of alignment between principal curvature directions and the cross field by evaluating whether the cross at each point match well with the eigenvectors of the shape operator, bypassing the need for explicit extraction of principal curvature direction, a step susciptable to unstability in nearly spherical or planar regions.

We implement NeurCross using a SIREN-based [Sitzmann et al. 2020] module for SDF fitting and a U-Net-based [Ronneberger et al. 2015] module for cross field prediction. Over 10,000 iterations, both components are optimized simultaneously to satisfy quadrangulation criteria. Finally, we employ global-seamless parametrization from libigl [Jacobson et al. 2017] aligned with our cross field, followed by quad mesh extraction using libQEx [Ebke et al. 2013]. Fig. 4 illustrates this process. Extensive experiments validate NeurCross's effectiveness, demonstrating improvements in singular point placement, robustness to noise and geometric variations, and approximation accuracy, as shown in the teaser figure.

Our contributions are summarized as follows:

- We propose *NeurCross*, the first self-supervised neural network for learning cross fields.
- We implicitly enforce cross field alignment with principal curvature directions via an SDF-based shape operator, naturally addressing potential ambiguity.
- We leverage an optimizable neural SDF as an underlying representation to coordinate requirements, dynamically adjusting to minor surface variations.

2 RELATED WORK

This paper focuses on developing a neural representation of the cross field for quadrilateral mesh generation. In this section, we review two main categories of related work: quad mesh generation techniques and neural SDF representations.

2.1 Quad Mesh Generation

Quadrilateral mesh generation has attracted significant attention in recent years. While some methods, such as Dual Marching Cubes (DMC) [Nielson 2004], can directly extract quad facets without relying on direction fields, the resulting meshes often lack quality, particularly in aligning with principal directions. Most state-of-the-art approaches rely on a cross field [Lai et al. 2010; Palmer et al. 2021; Ray et al. 2008] to guide the generation of high-quality quad meshes, as it ensures edge alignment and proper placement of irregular vertices. Typically, after computing a cross field, a parameterization step [Bommes et al. 2009; Chien et al. 2016; Levi and Zorin 2014; Myles et al. 2014] aligns gradients with the direction field and traces integer iso-lines across multiple charts [Ebke et al. 2013]. Although several robust quadrangulation methods [Dong et al. 2006; Gurung et al. 2011; Ling et al. 2014; Owen et al. 1999; Remacle et al. 2012; Velho and Zorin 2001; Zhang et al. 2010] operate independently of direction fields, they often fail to achieve global smoothness. Below, we review related works on direction fields.

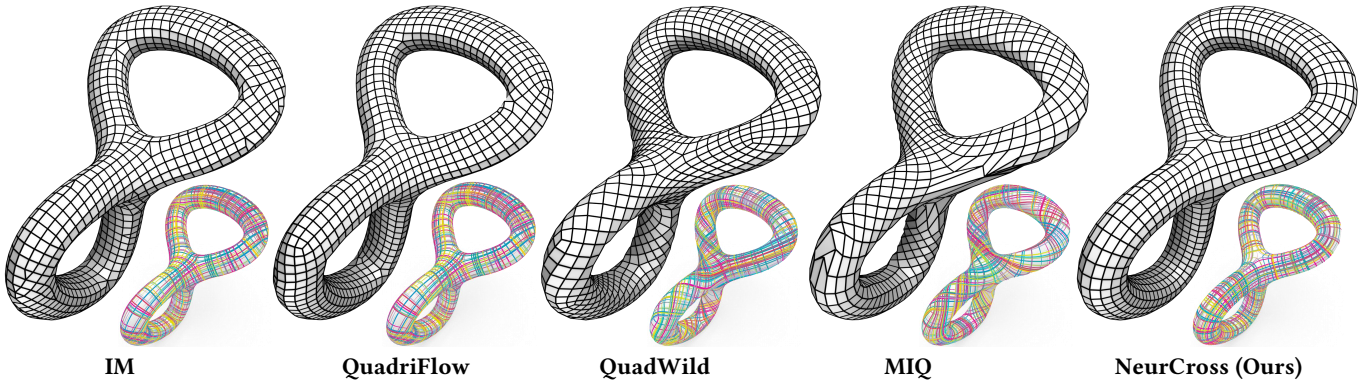


Fig. 2. Existing approaches typically rely on principal curvature directions as input. However, due to the inherent instability of these directions, current methods often prioritize the smoothness of the cross field at the cost of alignment with the principal curvature directions. To address this limitation, our approach avoids explicitly extracting principal curvature directions. Instead, we assess whether the cross field at each point can function as eigenvectors of the shape operator.

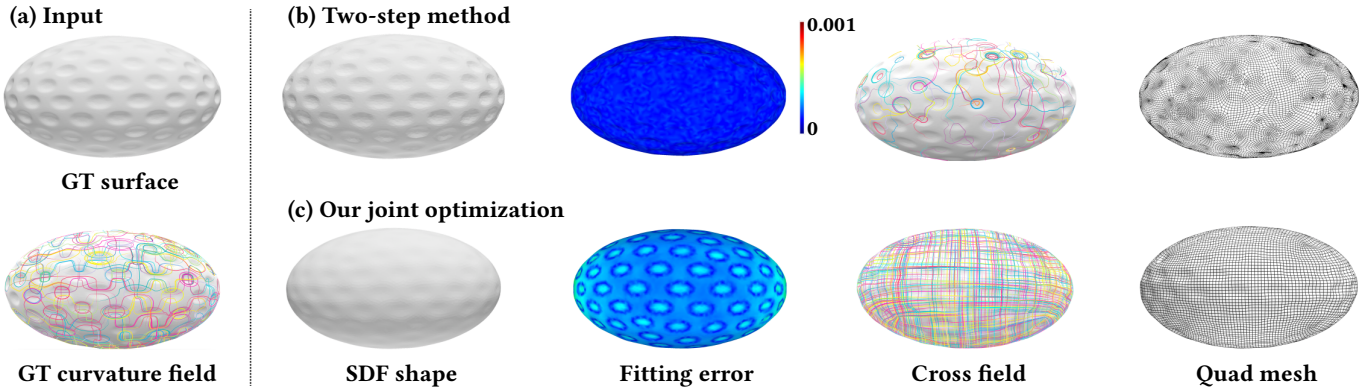


Fig. 3. (a) The input mesh and its ground-truth principal curvature directions. (b) Two-step optimization: by first precomputing an SDF that precisely fits the input shape, the subsequent optimization step still suffers from sensitivity to minor geometric variations, failing to yield the desired cross field. (c) Joint optimization: by treating the SDF as a proxy for the input shape, simultaneous optimization of the SDF and the cross field allows the SDF to approximate the input shape while remaining robust to minor geometric variations, resulting in the desired cross field. We visualize the fitting errors between the SDF surface and the original surface using a color-coded scheme.

A fundamental requirement for direction fields is to align edge directions with principal curvature directions [Bommes et al. 2013a, 2009; Fang et al. 2018; Hertzmann and Zorin 2000; Huang et al. 2018; Jakob et al. 2015; Kälberer et al. 2007; Lyon et al. 2019; Vaxman et al. 2016]. Lai et al. [2008] proposed an iterative relaxation scheme that incrementally aligns mesh edges with principal directions, though it requires additional post-processing to refine results. Jakob et al. [2015] introduced a unified local smoothing operator that optimizes both edge orientations and vertex positions in the output quad mesh. QuadriFlow [Huang et al. 2018] improved upon Instant Meshes [Jakob et al. 2015] by introducing linear and quadratic constraints, reducing singularities but struggling to preserve the original shape. Several methods [Bommes et al. 2013a; Huang et al. 2018; Jakob et al. 2015] optimize parametrization while incorporating integer constraints, a challenging mixed-integer programming (MIP) problem [Bommes et al. 2009] that is computationally intensive. These methods typically aim to minimize distortion and reduce singularities [Bommes et al. 2013a; Levi and Zorin 2014; Myles et al. 2014; Myles and Zorin 2013]. To address the computational complexity of IGM [Bommes et al. 2013a] on complex meshes, Ebke et al. [2016] proposed a framework using efficient decimation and coarse-to-fine mapping to improve interactive performance. Dielen et al. [2021] introduced a learning-based approach for predicting direction fields, demonstrating its potential for quad mesh generation. However, its reliance on domain-specific networks and canonical alignment limits its generalizability and robustness to non-rigid changes.

2.2 Neural SDF

The Signed Distance Function (SDF) is a widely used geometric representation in computer graphics, particularly for surface reconstruction. For example, radial basis functions (RBF) [Carr et al. 2001] approximate the SDF, enabling the extraction of the target surface as the zero-isosurface of the SDF.

SDFs have also been extensively employed in deep learning-based surface reconstruction, including supervised implicit surface reconstruction methods [Erler et al. 2020; Huang et al. 2022; Park et al.

(a) Triangle mesh **(b) Random initialization** **(c) Resultant cross field** **(d) Quad mesh**

Fig. 4. Given the input triangular surface in (a), starting with a randomly initialized cross field in (b), our *NeurCross* method produces a smooth cross field in (c) that is well aligned with the principal curvature directions of the input surface. We use the global-seamless parametrization from libigl to obtain a parametrization aligned with the computed cross field, and then use libQEx to extract the final quad mesh in (d).

2019] and self-supervised approaches [Ma et al. 2021, 2022; Wang et al. 2021]. For instance, IGR [Gropp et al. 2020] incorporates the Eikonal term to enforce implicit geometric regularization, providing an effective mechanism for surface reconstruction. SIREN [Sitzmann et al. 2020] demonstrates that periodic activation functions are well-suited for representing complex natural signals and their derivatives using implicit neural representations. DiGS [Ben-Shabat et al. 2022] integrates Laplacian energy as a soft constraint for the SDF, proving effective for reconstructing surfaces from unoriented point clouds. Neural-Singular-Hessian [Wang et al. 2023] ensures that the Hessian of the neural implicit function has a zero determinant for points near the surface, which is particularly useful for recovering details from unoriented point clouds. Additionally, Dong et al. [2024] proposed a zero Gaussian curvature constraint for reconstructing CAD-type surfaces from low-quality unoriented point clouds. All these methods leverage neural networks to approximate the SDF.

In this paper, SDFs play a central role in quad mesh generation, as the Hessian of the SDF fully encodes principal curvatures and their directions [Dong et al. 2024; Wang et al. 2023].

3 OUR APPROACH

3.1 Overview

The core idea of NeurCross is to leverage the optimizable neural Signed Distance Function (SDF) as an underlying representation to coordinate various requirements. On one hand, the adjustable SDF can effectively reduce sensitivity to minor surface variations. On the other hand, the SDF-based shape operator enables us to implicitly evaluate the difference between the principal curvature directions and the cross field. NeurCross consists of two core modules: a surface fitting module and an orientation prediction module, both centered around the SDF.

(1) **Surface Fitting Module:** This module aims to represent the input triangular surface using a neural SDF. Its loss incorporates the Dirichlet condition [Lipman 2021], the Eikonal condition [Gropp et al. 2020], and the singular Hessian condition [Wang et al. 2024] to ensure high fidelity to the input surface geometry. Together, these constraints guarantee an accurate surface representation.
(2) **Cross Field Prediction Module:** This module is designed to represent the cross field while implicitly enforcing alignment with principal curvature directions and spatial smoothness. It employs a U-Net architecture [Ronneberger et al. 2015] to predict a rotation angle for each triangular facet, yielding a geometry-aware cross field. Additionally, this module supports explicit alignment with geometric features.

These two modules are coordinated through a total loss function, which ensures simultaneous optimization of the SDF and the cross field during the training process. The interaction between the modules allows for dynamic updates to both the surface representation and the cross field, leading to improved accuracy and robustness. The overall network architecture is illustrated in Fig. 5.

Total Loss. Our total loss is defined as follows.

$$\mathcal{L} = \underbrace{\lambda_{\mathrm{E}}\mathcal{L}_{\mathrm{E}} + \lambda_{\mathrm{DM}}\mathcal{L}_{\mathrm{DM}} + \lambda_{\mathrm{DNM}}\mathcal{L}_{\mathrm{DNM}} + \tau\lambda_{\mathrm{AN}}\mathcal{L}_{\mathrm{AN}}}_{\text{SDF}} + \underbrace{\lambda_{\mathrm{AP}}\mathcal{L}_{\mathrm{AP}} + \lambda_{\mathrm{S}}\mathcal{L}_{\mathrm{S}}}_{\text{Cross Field}}, \quad (1)$$

where τ is the annealing factor [Dong et al. 2024; Wang et al. 2024, 2023]. The individual terms, along with their corresponding weights, will be detailed in the following subsections.

3.2 SDF Fitting

Let Θ denote the parameters of a neural SDF $f(\boldsymbol{x};\Theta) : \mathbb{R}^3 \to \mathbb{R}$, where $f = 0$ approximates the input triangular surface. We begin by sampling the centroid of each triangle, forming a point set $\mathcal{P}$, where each point is associated with a normal vector. In the following, we define a loss term to enforce alignment with the predefined surface normals, while leaving further details to Sec. 3.5.

SDF Based Shape Operator. The shape operator of a surface measures the rate of change of the unit normal in any direction, thereby describing how the shape changes in that direction. In differential geometry, it is very common to assume the surface has a parametric form, such that the shape operator defines a quadratic form on the tangent space, and the eigenvectors of the shape operator correspond exactly to the principal directions. In fact, the Hessian matrix of the SDF is closely related to the shape operator. For a point $\boldsymbol{p}$ on the base surface, the Hessian matrix $\boldsymbol{H_p}$ of the SDF has an eigenvalue of 0, with its corresponding eigenvector being the normal vector $\boldsymbol{n_p}$ [Dong et al. 2024; Wang et al. 2024, 2023]. Simultaneously, the other two eigenvectors of $\boldsymbol{H_p}$ correspond to the two principal curvature directions.

Alignment with Predefined Surface Normals. Since for a point $\boldsymbol{p}$ sufficiently close to the base surface, the eigenvector corresponding

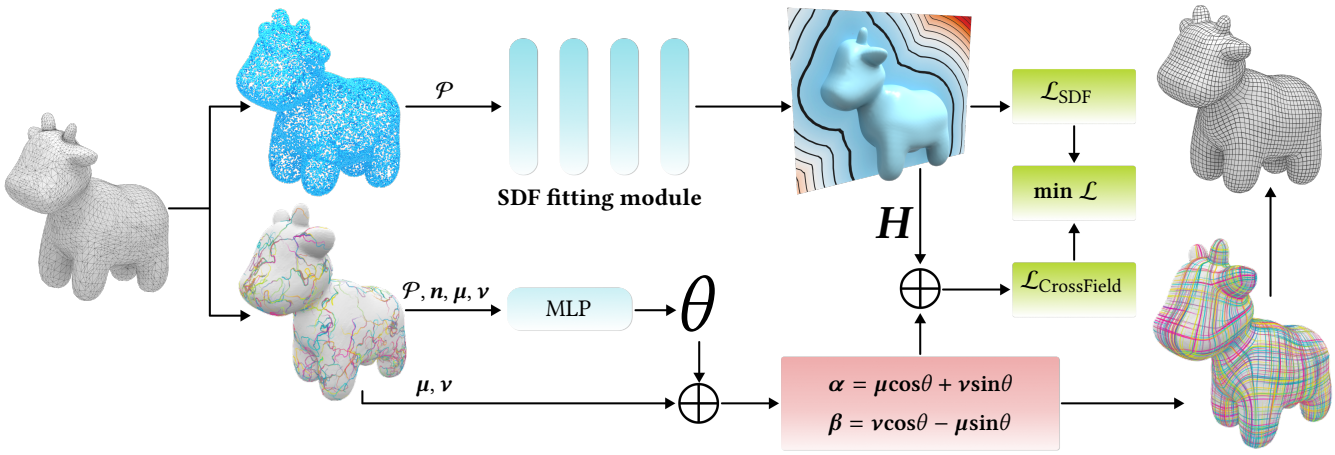


Fig. 5. Our self-supervised network pipeline for representing cross fields in quad mesh generation. All layers in the network are implemented as multi-layer perceptrons (MLPs), with the SDF fitting module utilizing the SIREN [Sitzmann et al. 2020] architecture. The circled "+" symbol denotes a data-combining operation.

to the zero eigenvalue of the Hessian matrix $\boldsymbol{H_p}$ of the SDF aligns with the normal vector $\boldsymbol{n_p}$ at $\boldsymbol{p}$. Given that the normal direction of $\boldsymbol{p} \in \mathcal{P}$ can be directly obtained from the input triangle mesh, we require the neural SDF to align with the predefined surface normal $\boldsymbol{n_p}$ as follows:

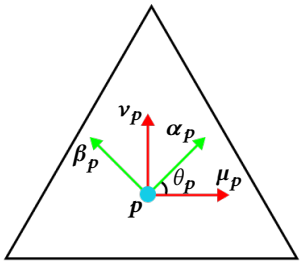
$$\boldsymbol{H_p} \cdot \boldsymbol{n_p} = \boldsymbol{0}. \tag{2}$$

The overall alignment with predefined surface normals can be quantified as:

$$\mathcal{L}_{\text{AN}} = \frac{1}{|\mathcal{P}|} \int_{\mathcal{P}} \left| \boldsymbol{H_p} \cdot \boldsymbol{n_p} \right| \mathrm{d}\boldsymbol{p}. \tag{3}$$

3.3 Cross Field Prediction

Local Coordinate System. Recall that each point $\boldsymbol{p} \in \mathcal{P}$ corresponds to the centroid of a triangular face. The task of computing the cross field involves inferring a pair of orthogonal vectors, $(\boldsymbol{\alpha_p}, \boldsymbol{\beta_p})$, that align as closely as possible with the principal curvature directions. To achieve this, we assume that each triangle has a pre-defined coordinate system with two axes, $\boldsymbol{\mu_p}$ and $\boldsymbol{\nu_p}$, which are mutually orthogonal unit vectors satisfying $\boldsymbol{\mu_p} \times \boldsymbol{\nu_p} = \boldsymbol{n_p}$. We introduce a rotation angle $\theta_{\boldsymbol{p}}$ to represent $\boldsymbol{\alpha_p}$ and $\boldsymbol{\beta_p}$ as follows:



$$\begin{cases} \boldsymbol{\alpha_p} = \boldsymbol{\mu_p} \cos\theta_{\boldsymbol{p}} + \boldsymbol{\nu_p} \sin\theta_{\boldsymbol{p}}, \\ \boldsymbol{\beta_p} = \boldsymbol{\nu_p} \cos\theta_{\boldsymbol{p}} - \boldsymbol{\mu_p} \sin\theta_{\boldsymbol{p}}. \end{cases} \tag{4}$$

See the inset figure for an illustration. Notably, $\boldsymbol{\alpha_p}$ and $\boldsymbol{\beta_p}$ are naturally mutually orthogonal unit vectors. As a result, the optimization of the cross field reduces to computing the rotation angle $\theta_{\boldsymbol{p}}$ for each triangle.

Implicit Alignment with Principal Directions. An explicit approach to implementing alignment with principal directions involves comparing the cross field with pre-extracted principal directions. However, most existing methods for extracting principal directions heavily rely on local shape variations, which can lead to instability, particularly when the local geometry is approximately planar or spherical. To address this limitation, we adopt an implicit alignment strategy by evaluating the compatibility between the cross field and the shape operator.

To align the cross field with the principal directions, we encourage $\boldsymbol{\alpha_p}$ and $\boldsymbol{\beta_p}$ to coincide with two of the eigenvectors of $\boldsymbol{H_p}$. To enforce collinearity between $\boldsymbol{H_p\alpha_p}$ and $\boldsymbol{\alpha_p}$, we impose the following condition:

$$\boldsymbol{H_p\alpha_p} \times \boldsymbol{\alpha_p} = \boldsymbol{0}. \tag{5}$$

Similarly, we require:

$$\boldsymbol{H_p\beta_p} \times \boldsymbol{\beta_p} = \boldsymbol{0}. \tag{6}$$

We define the loss term to measure alignment with the principal directions as follows:

$$\mathcal{L}_{\text{AP}}^{(1)} = \frac{1}{|\mathcal{P}|} \int_{\mathcal{P}} \left| \boldsymbol{H_p\alpha_p} \times \boldsymbol{\alpha_p} \right| + \left| \boldsymbol{H_p\beta_p} \times \boldsymbol{\beta_p} \right| \mathrm{d}\boldsymbol{p}. \tag{7}$$

Smoothness of the Cross Field. Consider two pairs of orthogonal unit vectors in a plane, denoted as $(\boldsymbol{\alpha}_1, \boldsymbol{\beta}_1)$ and $(\boldsymbol{\alpha}_2, \boldsymbol{\beta}_2)$. We say that the pair $(\boldsymbol{\alpha}_1, \boldsymbol{\beta}_1)$ aligns with $(\boldsymbol{\alpha}_2, \boldsymbol{\beta}_2)$ if either $\boldsymbol{\alpha}_1$ and $\boldsymbol{\alpha}_2$ are colinear, or $\boldsymbol{\alpha}_1$ and $\boldsymbol{\beta}_2$ are colinear.

Based on the above definition, it can be proved that $(\boldsymbol{\alpha}_1, \boldsymbol{\beta}_1)$ aligns with $(\boldsymbol{\alpha}_2, \boldsymbol{\beta}_2)$ if and only if

$$\left|\boldsymbol{\alpha}_1 \cdot \boldsymbol{\alpha}_2\right| + \left|\boldsymbol{\alpha}_1 \cdot \boldsymbol{\beta}_2\right| + \left|\boldsymbol{\beta}_1 \cdot \boldsymbol{\alpha}_2\right| + \left|\boldsymbol{\beta}_1 \cdot \boldsymbol{\beta}_2\right| \tag{8}$$

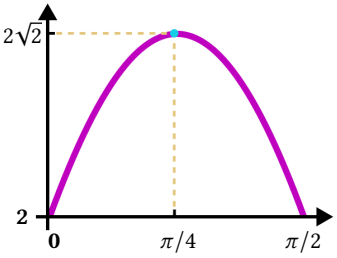
achieves the minimum. We explain the correctness as follows. Without loss of generality, we assume that $\boldsymbol{\alpha}_1 = (1, 0)$ and $\boldsymbol{\beta}_1 = (0, 1)$. By denoting $\boldsymbol{\alpha}_2$ as $(\cos\theta, \sin\theta)$, the above sum simplifies to

$$2(|\cos\theta| + |\sin\theta|). \tag{9}$$

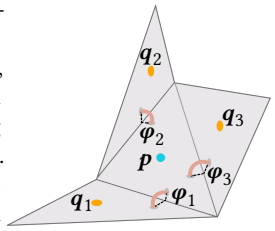
(a) Input mesh **(b) Cross field** **(c) Quad mesh**

Fig. 6. The smoothness constraint of the cross field better controls the distribution of singularity points. From left to right: (a) an input triangular mesh; (b) a cross field computed with NeurCross; (c) the quad mesh extracted from the cross field.

In the inset figure, we illustrate how the function value of $2(|\cos\theta| + |\sin\theta|)$ varies with θ. It can be observed that the minimum is achieved at $\theta = k\frac{\pi}{2}$, while the maximum occurs at $\theta = k\frac{\pi}{2} + \frac{\pi}{4}$. Therefore, it can be concluded that only when the sum reaches the minimum value of 2, one cross aligns with another ($\theta = k\frac{\pi}{2}$).

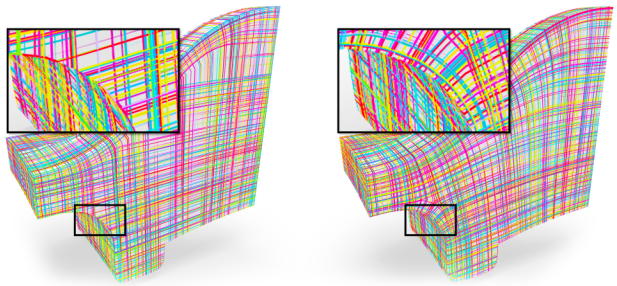


We denote the three neighboring points of $\boldsymbol{p}$ as $\boldsymbol{q}_1, \boldsymbol{q}_2, \boldsymbol{q}_3$. Since each $\boldsymbol{q}_i$ lies on a neighboring face, a rotation around the common edge is necessary before aligning the directions between $\boldsymbol{p}$ and $\boldsymbol{q}_i$. We define $\{\boldsymbol{R}_i \mid i = 1, 2, 3\}$ as the rotation matrices associated with dihedral angles $\{\boldsymbol{\varphi}_i \mid i = 1, 2, 3\}$ and shared edges, which can be precomputed. To this end, the smoothness loss can be written as



$$\mathcal{L}_{\mathrm{S}} = \frac{1}{3|\mathcal{P}|}\int_{\mathcal{P}}\sum_{i=1}^{3}\Big(\big|\boldsymbol{\alpha}_{\boldsymbol{p}}\cdot\boldsymbol{R}_i\boldsymbol{\alpha}_{\boldsymbol{q}_i}\big| + \big|\boldsymbol{\alpha}_{\boldsymbol{p}}\cdot\boldsymbol{R}_i\boldsymbol{\beta}_{\boldsymbol{q}_i}\big| + \big|\boldsymbol{\beta}_{\boldsymbol{p}}\cdot\boldsymbol{R}_i\boldsymbol{\alpha}_{\boldsymbol{q}_i}\big| + \big|\boldsymbol{\beta}_{\boldsymbol{p}}\cdot\boldsymbol{R}_i\boldsymbol{\beta}_{\boldsymbol{q}_i}\big| - 2\Big)\,\mathrm{d}\boldsymbol{p}. \tag{10}$$

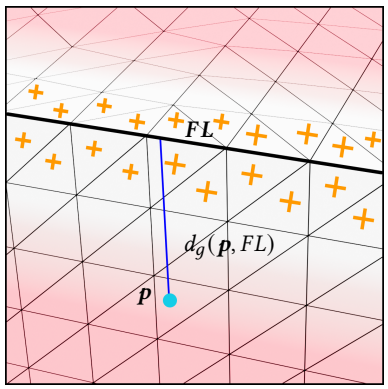
Remark: Consider a spherical surface, as shown in Fig. 6. At each point on such a surface, the principal curvature directions are not unique. In this case, the smoothness constraint of the cross field plays a crucial role in better controlling the distribution of singularity points. It is worth noting that our implicit principal curvature alignment is simultaneously satisfied. Moreover, Fig. 6 highlights the importance of effective cross field smoothing, which has also been addressed in prior work. Knöppel et al. [2013] and Diamanti et al. [2014] introduced convex smoothness energies to smooth N-RoSy fields. Knöppel et al. [2013]'s method achieves global optimality but requires a nonlinear transformation, which can cause extra singularities and distortion. Jakob et al. [2015] uses an extrinsic energy to align with surface features, but its strong reliance on local information often leads to suboptimal results and unwanted singularities. In contrast, our NeurCross smooths the cross field using Equ. 10, introducing singularities only in areas with high curvature variation. A detailed comparison is provided in Section 4.2.



(a) w/o feature line const. **(b) w/ feature line const.**

Fig. 7. NeurCross supports feature line constraints. (a) The results without (w/o) the feature line constraint (const.); and (b) The result with (w/) the feature line constraint.

Sharp Feature Alignment. As pointed out in [Pietroni et al. 2021], in the context of quad-meshing, it is important to incorporate feature lines of the input shape, such as crease angles in CAD models, when generating quadrangulation outcomes. However, reconciling this requirement with all other objectives is challenging. Generally, the influence of feature lines diminishes with increasing distance.



As illustrated in the inset figure, let FL be the feature lines, where the cross field at each point of FL has been specified. We use $d_g(\boldsymbol{p}, FL)$ to denote the geodesic distance between a surface point $\boldsymbol{p}$ and FL. We introduce

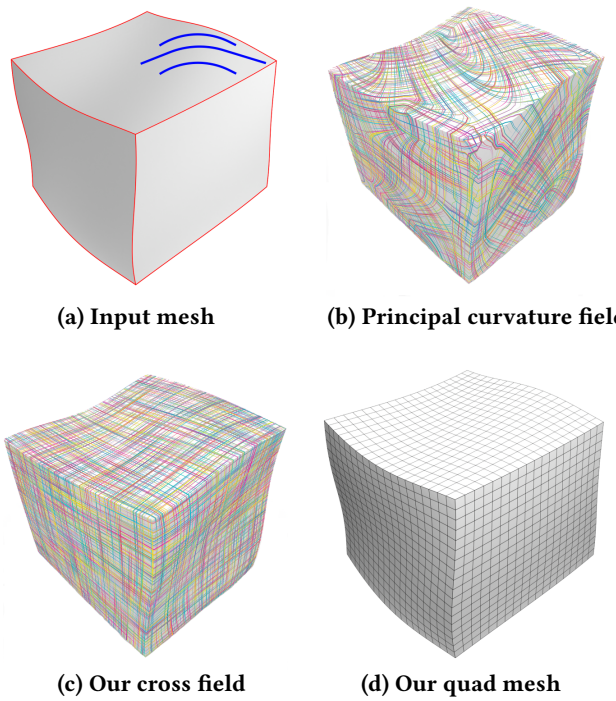
$$\boldsymbol{D}_{\boldsymbol{p}} = 1 - \exp\left(-\rho_{\text{feature}}\, d_g(\boldsymbol{p}, FL)\right), \tag{11}$$

and redefine the principal curvature direction alignment as follows:

$$\mathcal{L}_{\mathrm{AP}} = \frac{1}{|\mathcal{P}|}\int_{\mathcal{P}} \boldsymbol{D}_{\boldsymbol{p}}\left(\big|\boldsymbol{H}_{\boldsymbol{p}}\boldsymbol{\alpha}_{\boldsymbol{p}}\times\boldsymbol{\alpha}_{\boldsymbol{p}}\big| + \big|\boldsymbol{H}_{\boldsymbol{p}}\boldsymbol{\beta}_{\boldsymbol{p}}\times\boldsymbol{\beta}_{\boldsymbol{p}}\big|\right)\mathrm{d}\boldsymbol{p}, \tag{12}$$

where ρ_{feature} (10 by default) in $\boldsymbol{D}_{\boldsymbol{p}}$ serves as a sufficiently large constant to regulate the influence of the feature line. The color gradient in the insert figure represents the value of $\boldsymbol{D}_{\boldsymbol{p}}$, which increases with distance from the feature line, reflecting the gradual reduction in FL influence on the surface point. For ease of implementation, we approximate $d_g(\boldsymbol{p}, FL)$ using straight-line distances. As shown in Fig. 7, the crease line of the model is faithfully preserved in the neighborhood of FL, while its influence diminishes as the distance increases. Moreover, in regions where sharp features conflict with principal curvature directions, our NeurCross prioritizes feature alignment (see Fig. 8).

Rotation Angle Prediction. Drawing inspiration from various object segmentation works [Hu et al. 2021; Milano et al. 2020], we employ the U-Net architecture [Ronneberger et al. 2015] to construct our rotation angle prediction network. The input to our U-Net-based



(a) Input mesh (b) Principal curvature field

(c) Our cross field (d) Our quad mesh

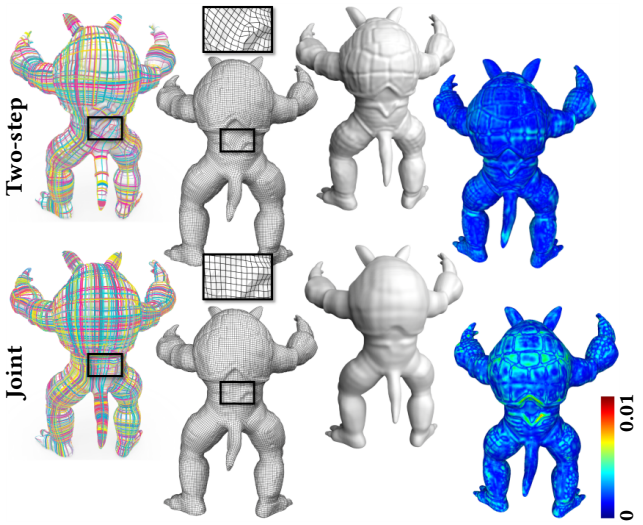
Fig. 8. NeurCross enforces alignment with sharp features even when they diverge from principal curvature directions. (a) An input mesh with conflicting principal curvature directions (blue) and feature curves (red); (b) The principal curvature field of the input surface; (c) The cross field computed by NeurCross; (d) The resulting quad mesh generated using NeurCross.

network includes the point cloud $\mathcal{P}$, along with the normal direction for each point, and the direction vectors $\boldsymbol{\nu}$ and $\boldsymbol{\mu}$ that represent the local coordinate system. The network outputs a scalar value $\omega_{\boldsymbol{p}} \in [0, 1]$ for each point $\boldsymbol{p}$, allowing the rotation angle $\theta_{\boldsymbol{p}}$ to be represented as $\theta_{\boldsymbol{p}} = 2\pi\omega_{\boldsymbol{p}}$. For this module, we initialize the orientation at a point $\boldsymbol{p}$ using a normal distribution with a mean of 0 and a standard deviation of 0.2.

3.4 SDF and Cross Field Joint Optimization

One challenge in quad meshing is balancing the overall simplicity of the cross field with alignment to the principal directions, a difficulty that becomes more pronounced for geometrically or topologically complex shapes. In this paper, we address this challenge by using an optimizable neural SDF as a bridge to achieve this balance. Notably, the neural SDF and the cross field are optimized simultaneously.

An alternative approach is to first fully optimize the SDF to accurately represent the input shape and then keep it fixed. However, in this case, the cross field may become severely constrained, as it must align with potentially irregular curvature lines of the pre-fixed SDF. As shown in Fig. 9, the fixed SDF, while providing an accurate representation, may introduce overly complex curvature lines, leading to an excessive number of singular points in the final cross field.



(a) Cross field (b) Quad mesh (c) SDF fitting (d) Fitting error

Fig. 9. Comparison between the two-step method (top row) and our joint optimization strategy (bottom row). (a) Cross field; (b) Quad mesh; (c) The underlying SDF surface; (d) Fitting error between the SDF and the input triangular mesh. The final quad mesh is extracted based on the computed cross field and the input triangular mesh. As shown, our joint optimization strategy balances the overall smoothness of the cross field with alignment to the principal curvature directions.

3.5 Implementation Details

SDF Loss Terms. The loss terms used to regularize the SDF include the Eikonal condition [Gropp et al. 2020], the Dirichlet condition [Lipman 2021], and the alignment condition [Wang et al. 2024, 2023]. For further details on these loss terms, we refer readers to the existing literature [Dong et al. 2024; Wang et al. 2024, 2023].

Sampling Strategy. A neural SDF is employed to approximate the base surface, with regularization at sample points, as detailed in previous works [Ben-Shabat et al. 2022; Boulch and Marlet 2022; Dong et al. 2024; Gropp et al. 2020; Hou et al. 2022; Huang et al. 2022; Kazhdan and Hoppe 2013; Ma et al. 2021; Sitzmann et al. 2020; Wang et al. 2024, 2023; Xu et al. 2022]. We extract centroids from all triangles in the mesh to define the sample set $\mathcal{P}$, chosen for their representative nature of the surface. For each point $\boldsymbol{p} \in \mathcal{P}$, the normal vector $\boldsymbol{n_p}$ is derived from the corresponding triangular face's normal.

Given that each triangular face has three neighboring faces, neighboring relationships between points in $\mathcal{P}$ can be thus established. The SDF is assumed differentiable within a narrow, thin-shell space Ω, which closely encloses the base surface. Following previous studies [Dong et al. 2024; Gropp et al. 2020; Ma et al. 2021; Wang et al. 2024, 2023], Ω is sampled using random displacements around each point $\boldsymbol{p} \in \mathcal{P}$. A Gaussian distribution centered at each $\boldsymbol{p}$, with a standard deviation based on the distance to its k-th nearest neighbor (typically $k = 50$), is used for this purpose. A one-point sampling technique is then applied to generate the sample set Ω, which is the same size as $\mathcal{P}$.

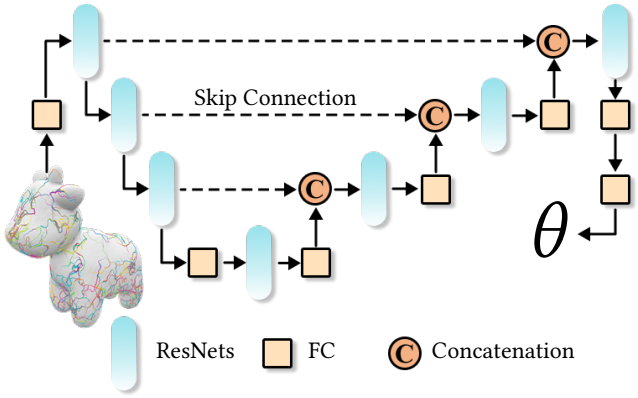


Fig. 10. An overview of our U-Net-based module designed for predicting the rotation angle θ. The network architecture incorporates the ResNet structure, with all layers being Multi-Layer Perceptrons (MLPs). The "ResNets" represents a combination of multiple ResNet blocks, the "FC" denotes the Fully Connected Layer, and the circled "C" symbol indicates the concatenation operation.

To prevent outlier zero iso-surfaces far from $\mathcal{P}$, we uniformly sample the bounding box (assuming input points are normalized within $[-0.5, 0.5]^3$), generating a sample set $\mathcal{Q}$.

Eikonal Condition. The Eikonal condition is crucial for ensuring that the SDF $f(\boldsymbol{x}; \Theta)$ maintains a unit gradient at every point, i.e., $\|\nabla f\| = 1$, particularly in the vicinity of the surface. The corresponding loss term is defined as:

$$\mathcal{L}_E = \frac{1}{|\mathcal{P}| + |\Omega|} \int_{\mathcal{P} \cup \Omega} \left| 1 - \|\nabla f(\boldsymbol{x}; \Theta)\| \right| \mathrm{d}\boldsymbol{x}, \tag{13}$$

where $\mathcal{P}$ represents the sample points (e.g., centroids of mesh triangles), and Ω encodes a narrow band around the surface where the SDF is differentiable. Notably, the point set $\mathcal{Q}$, which represents regions far from the base surface, is excluded from this loss term and serves to prevent outlier zero isosurfaces in distant regions.

Dirichlet Condition. For every point $\boldsymbol{p} \in \mathcal{P}$, it is essential that they lie as close as possible to the underlying surface, ideally satisfying $f(\boldsymbol{p}; \Theta) = 0$. Conversely, for points $\boldsymbol{q} \in \mathcal{Q}$, which lie away from the underlying surface, we aim to partition $\mathcal{Q}$ into interior and exterior regions, preventing f from degenerating. These conditions are formalized as the following loss terms:

$$\mathcal{L}_{\mathrm{DM}} = \frac{1}{|\mathcal{P}|} \int_{\mathcal{P}} \left| f(\boldsymbol{p}; \Theta) \right| \mathrm{d}\boldsymbol{p}, \tag{14}$$

and

$$\mathcal{L}_{\mathrm{DNM}} = \frac{1}{|\mathcal{Q}|} \int_{\mathcal{Q}} \exp\left(-\rho_{\mathrm{DNM}} \left| f(\boldsymbol{q}; \Theta) \right| \right) \mathrm{d}\boldsymbol{q}, \tag{15}$$

where ρ_{DNM} (defaulting to 100) is the exponential weight controlling the penalty for deviations from the surface.

SIREN-based Module. Similar to various implicit surface reconstruction methods [Ben-Shabat et al. 2022; Lipman 2021; Wang et al. 2024, 2022b, 2023], our NeurCross employs the SIREN [Sitzmann et al. 2020] network architecture, which consists of four hidden layers with 256 units each. The SIREN architecture is based on multi-layer perceptrons (MLPs), where inputs are first normalized to the range $[-1, 1]^3$ before being processed by the network. The activation function used in this architecture is the sine periodic function, which operates on the input point cloud $\mathcal{P}$ to produce the SDF field required for computing the Hessian matrix. For initializing this SIREN-based module, we follow SIREN's initialization strategy [Sitzmann et al. 2020], which ensures that the distribution of activations remains consistent across all layers of the network.

Table 1. The sizes of the building blocks within our U-Net-based module. $n_{\mathrm{bottleneck}}$ denotes the number of bottleneck layers within each ResNet block. From the 1st to the 7th block, the values of $n_{\mathrm{bottleneck}}$ are set to 3, 4, 6, 3, 3, 4, and 6, respectively.

Layer Name	Layer Architecture	Output Size
	1×1, input_size=12	256
ResNets #1, #2, #3	$\begin{bmatrix} 1\times1, 256 \\ 1\times1, 64 \\ 1\times1, 64 \end{bmatrix} \times n_{\mathrm{bottleneck}}$	256
ResNets #4, #5, #6, #7	$\begin{bmatrix} 1\times1, 512 \\ 1\times1, 128 \\ 1\times1, 128 \end{bmatrix} \times n_{\mathrm{bottleneck}}$	512
	1×1, 512	32
	1×1, 32	1

U-Net-based Module. We adopt a U-Net architecture [Ronneberger et al. 2015] as the backbone for predicting rotation angles. To address the vanishing gradient issue in deep networks, we incorporate ResNet blocks [He et al. 2016] as the core components of the U-Net. As shown in Fig. 10, our network consists of ResNet blocks (ResNets) and fully connected layers (FC), with all layers implemented using MLPs. A detailed configuration of each ResNet block is provided in Tab. 1.

Parameter Setting. In this paper, we set the weights as follows based on our tailored configurations: $\lambda_{\mathrm{E}} = 50$, $\lambda_{\mathrm{DM}} = 7000$, $\lambda_{\mathrm{DNM}} = 600$, $\lambda_{\mathrm{AN}} = 3$, $\lambda_{\mathrm{AP}} = 10$, and $\lambda_{\mathrm{S}} = 30$. The annealing factor τ remains 1 during the initial 20% of iterations, then linearly decreases to 3×10^{-4} from 20% to 40% of the iteration span, and finally drops to 0 towards the end. Throughout the training phase, we apply the Adam optimizer [Kingma and Ba 2014] with a default learning rate of 5×10^{-5} and complete 10,000 iterations.

Quad Mesh Extraction. The extraction of the quad mesh from our cross field follows a two-step scheme, as detailed in references Bommes et al. [2009], Ebke et al. [2013], and Dielen et al. [2021], to achieve a high-quality outcome. This process begins with a step of parametrization based on our cross field. In implementation, we utilize the global-seamless parametrization technique from libigl [Jacobson et al. 2017] to align the parametrization with our cross field. Subsequently, we employ libQEx [Ebke et al. 2013] to extract the quad mesh from this parameterization.

4 EXPERIMENTS

Evaluation Metrics and Platform. To evaluate the accuracy of the quad mesh, we utilize four primary metrics [Huang et al. 2018; Wang et al. 2023]: area distortion (Area), angle distortion (Angle),

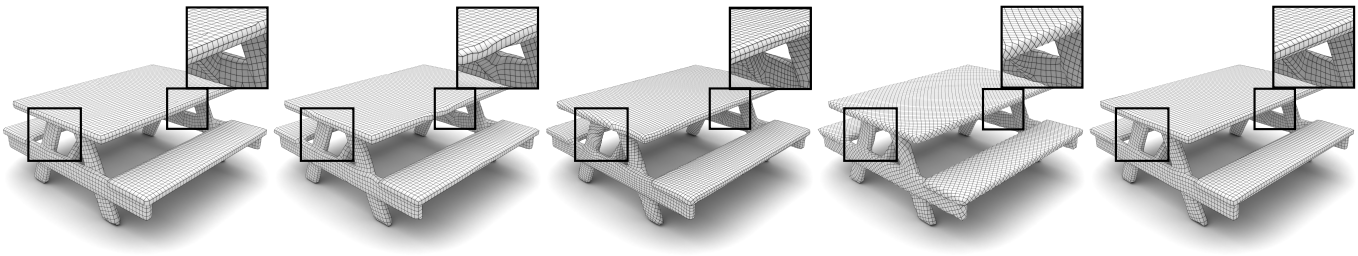


Fig. 11. Quad meshes generated by NeurCross and four other methods on the table model in the ShapeNet dataset [Chang et al. 2015].

the number of singularities (# of Sings), chamfer distance (CD), and Jacobian Ratio (JR). The area distortion metric, scaled by 10,000, represents the standard deviation of the areas of the quadrilateral faces within a mesh. The angle distortion is quantified using the formula $\sqrt{\frac{1}{N}\sum_i(\phi_i - \frac{\pi}{2})^2}$, where the summation extends over all angles ϕ in the quad mesh, and N denotes their count. Chamfer distance, scaled by 10,000 and calculated using the L_1-norm, quantifies the similarity between two surfaces. The Jacobian Ratio quantifies the uniformity of local deformation in quadrilateral elements. It is defined as the ratio of the smallest to the largest determinant of the Jacobian matrices at element corners, providing a dimensionless measure from 0 (degenerate element) to 1 (perfect parallelogram). The experiments detailed in this paper were executed on an NVIDIA GeForce RTX 3090 graphics card equipped with 24GB of video memory and powered by an AMD EPYC 7642 processor.

Datasets. We carry out quad mesh generation experiments on two popular datasets: ShapeNet [Chang et al. 2015] and Thingi10K [Zhou and Jacobson 2016]. To maintain uniformity in evaluation, all input meshes are scaled to fit within the range of $[-0.5, 0.5]^3$ ensuring a consistent and fair basis for comparison across all datasets.

4.1 Comparison on Open Datasets

We assess the efficacy of our proposed method, NeurCross, by conducting evaluations on two distinct datasets and comparing its performance against four contemporary state-of-the-art quadrilateral mesh generation methods. For the three methods, namely Instant Meshes (IM) [Jakob et al. 2015], QuadriFlow [Huang et al. 2018], and QuadWild [Pietroni et al. 2021], we employed the open-source implementations that are readily available. It is worth noting that QuadWild [Pietroni et al. 2021] is primarily a quadrangulation method rather than a cross field generation approach. The Mixed-Integer Quadrangulation (MIQ) method [Bommes et al. 2009] does not release its source code; therefore, we use the implementation provided by libigl [Jacobson et al. 2017]. However, the available implementation does not support the feature alignment constraint. For a fair comparison with MIQ, we employ the same parameterization and extraction techniques, namely global-seamless parameterization and libQEx [Ebke et al. 2013].

ShapeNet Dataset. The ShapeNet dataset [Chang et al. 2015] consists of a diverse range of human-made models. As the global-seamless parametrization from libigl [Jacobson et al. 2017] cannot handle non-manifold meshes, we use manifold ShapeNet meshes repaired with DualOctreeGNN [Wang et al. 2022a]. We apply our NeurCross to three randomly selected categories—airplane, bench, and cabinet—which together contain 7433 models. For a fair comparison, all generated quad meshes are standardized to contain an average of 6,000 vertices and 12,000 faces.

Table 2. Quantitative comparison on the ShapeNet dataset [Chang et al. 2015]. Within each column, the best scores are emphasized with bold and underlining (**best**), whereas the second-best scores are highlighted in bold (**second best**). The quad mesh generated by all the methods comprises an average of 6,000 vertices and 12,000 faces.

	Area ↓	Angle ↓	# of Sings ↓	CD ↓	JR ↑
IM [Jakob et al. 2015]	1.57	11.78	200.52	8.97	0.70
QuadriFlow [Huang et al. 2018]	2.28	13.24	91.58	50.18	0.65
QuadWild [Pietroni et al. 2021]	**1.52**	**11.05**	93.04	10.34	**0.73**
MIQ [Bommes et al. 2009]	5.23	12.89	**82.12**	**8.25**	0.58
NeurCross (Ours)	**1.48**	**9.85**	**85.32**	**8.03**	**0.78**

Table 3. Quantitative comparison on the Thingi10K dataset [Zhou and Jacobson 2016]. The quad mesh generated by all methods comprises an average of 10,000 vertices and 20,000 faces. Within each column, the best scores are emphasized with bold and underlining (**best**), whereas the second-best scores are simply highlighted in bold (**second best**).

	Area ↓	Angle ↓	# of Sings ↓	CD ↓	JR ↑
IM [Jakob et al. 2015]	1.45	10.57	397.18	9.83	0.75
QuadriFlow [Huang et al. 2018]	1.58	12.39	78.32	26.89	0.72
QuadWild [Pietroni et al. 2021]	1.40	10.16	85.11	28.12	**0.77**
MIQ [Bommes et al. 2009]	**1.38**	**9.85**	**66.54**	**8.57**	0.67
NeurCross (Ours)	**1.33**	**9.68**	**68.96**	**8.22**	**0.81**

In Fig. 11, we display the quad meshes generated by our NeurCross alongside four other methods. For this example, although IM [Jakob et al. 2015] produces a regular quadrilateral mesh, the outcome includes some triangular elements. More comparisons between IM and our method will be provided in Sec. 4.2. QuadriFlow [Huang et al. 2018], MIQ [Bommes et al. 2009], and QuadWild [Pietroni et al. 2021] generate some misaligned quadrilateral elements, as seen in the highlighted windows. In contrast, our method yields a better quadrilateral mesh. Tab. 2 shows the quantitative comparison of our method against the four approaches.

Thingi10K Dataset. The Thingi10K dataset [Zhou and Jacobson 2016] features a variety of shapes with intricate geometric details. For our analysis based on Thingi10K, we tested 1,000 randomly selected triangle meshes from the dataset, which were also used as

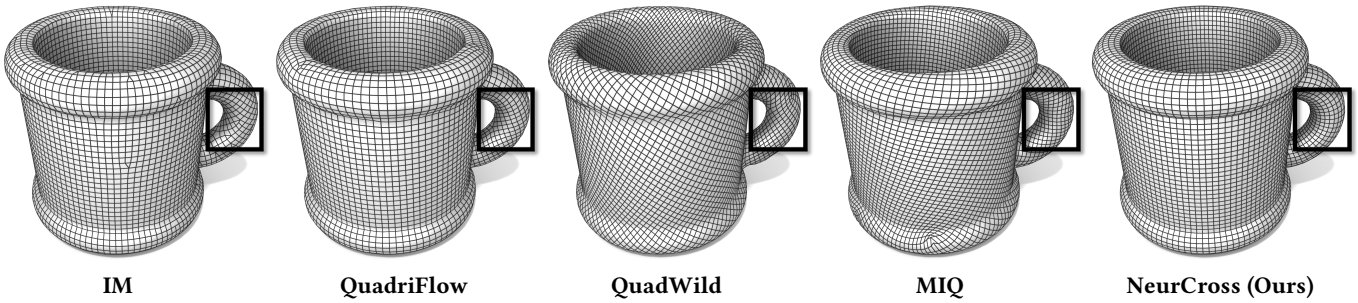


Fig. 12. Quad meshes generated by NeurCross and four other methods on a cup model in the Thingi10K dataset [Zhou and Jacobson 2016].

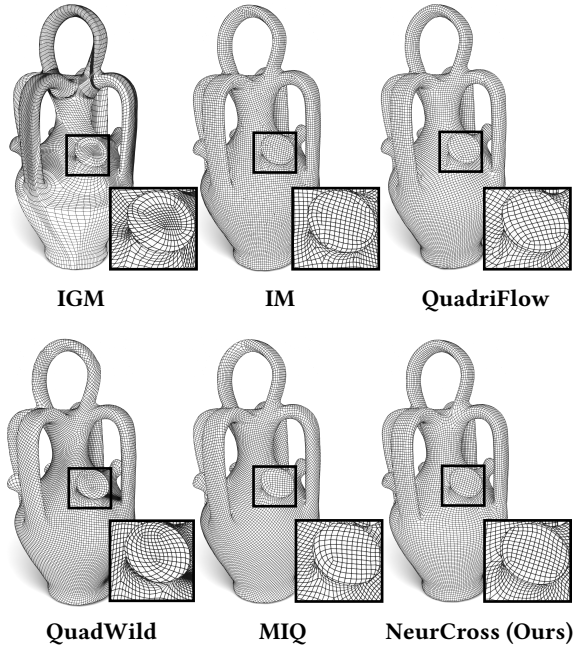


Fig. 13. Comparison with five state-of-the-art methods using data provided in IGM [Bommes et al. 2013a].

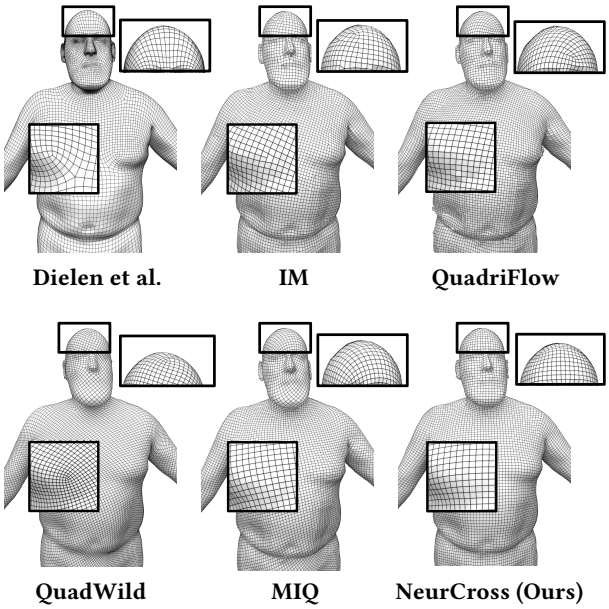


Fig. 14. Comparison with five methods on Human Body data. Dielen et al. [2021] proposed a supervised learning-based approach designed to generate quad meshes on human body data from the FAUST dataset [Bogo et al. 2014]. Due to the absence of available open-source data, the comparison result in the upper left is taken from Dielen et al. [2021]'s paper.

inputs for all comparative methods. The quad meshes generated by all methods contain, on average, 10,000 vertices and 20,000 faces to maintain fidelity to the original models.

Quantitative comparison statistics are presented in Tab. 3. Our method consistently outperforms others on average across this dataset. Interestingly, MIQ [Bommes et al. 2009] shows commendable performance on this dataset. However, it is important to note that despite this improvement, the issue of producing distorted quadrilaterals in the resulting quad mesh remains (see Fig. 12 and the JR metric in Tab. 3). Quadwild [Pietroni et al. 2021] requires smoothing of the generated quad mesh, which compromises geometric details and increases the Chamfer Distance (CD). In contrast, our method produces a more intuitive cross field without needing to introduce excessive singular points (see zoom-in windows in Fig. 12).

4.2 Further Comparison

Comparison with IGM. IGM [Bommes et al. 2013a] is characterized as a global approach, primarily focused on the joint optimization of parametrization with integer constraints. Like our method, it also utilizes libQEx [Ebke et al. 2013] for extracting quad meshes from the parametrization. Although IGM provides full control over edge alignment and singularity placement, yielding high-quality quad meshes, its lack of scalability can lead to severely distorted quadrilaterals (see Fig. 13).

Comparison with Learning Methods. Dielen et al. [2021] represents a pioneering effort in quad mesh generation through deep learning methodologies. Their method uses a supervised network architecture to predict the frame field, comprising both a global network and a local network for field prediction. Subsequently, the parametrization-based quadrangulation method proposed in Campen et al. [2015b] is employed to generate the quad meshes.

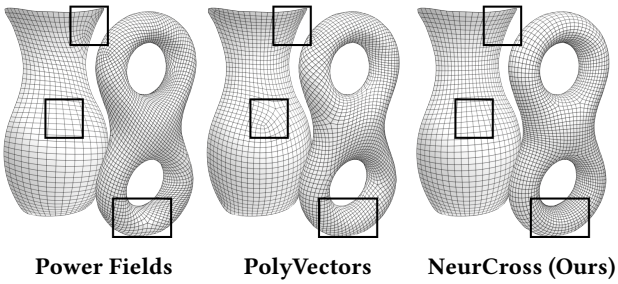


Fig. 15. Comparison of quad meshes generated by Power Fields [Knöppel et al. 2013], PolyVectors [Diamanti et al. 2014], and NeurCross.

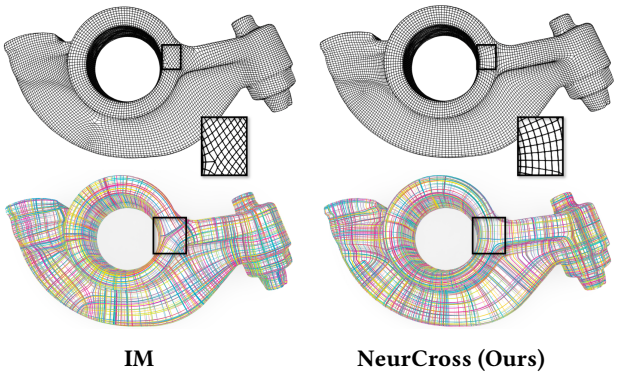


Fig. 16. Comparison with IM. Here, the same approach—applying global seamless parameterization [Jacobson et al. 2017] and libQEx [Ebke et al. 2013]— is used to extract quadrilateral meshes from the respective cross fields of IM and our NeurCross.

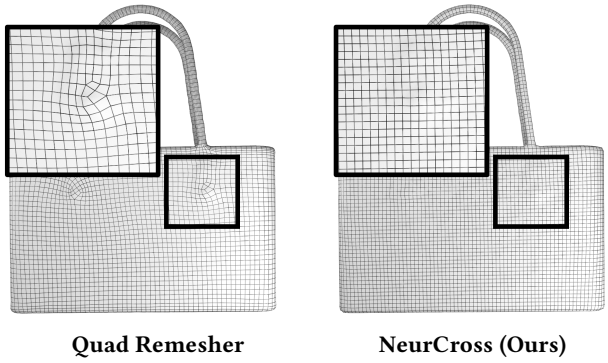


Fig. 17. Comparison of quad meshes generated by Quad Remesher [Remesher 2019] and NeurCross.

However, due to the inherent constraints of supervised learning, this approach shows optimal performance only on the FAUST dataset [Bogo et al. 2014], a limitation not encountered by our self-supervised method. Owing to a lack of required data, our comparison is limited to the model presented in their paper (see Fig. 14).

Comparison with Power Fields and PolyVectors. Power Fields [Knöppel et al. 2013] efficiently constructs smooth n-direction fields on surfaces by solving a sparse eigenvalue problem, ensuring global optimality and high-quality results. PolyVectors [Diamanti et al. 2014] extends N-RoSy fields to N-PolyVector fields by relaxing orthogonality and symmetry constraints, enabling their computation

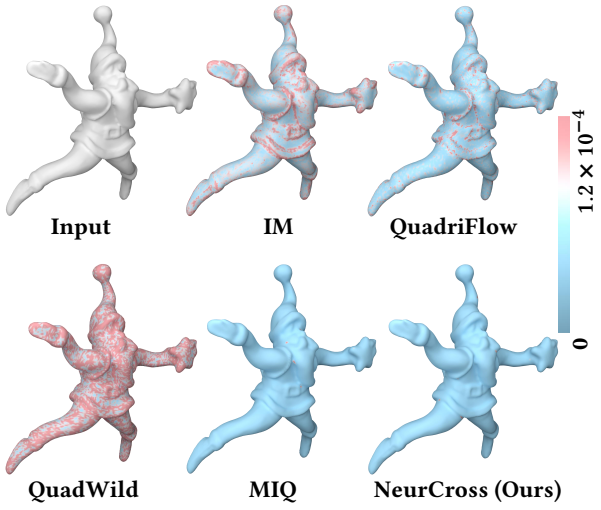


Fig. 18. **Approximation accuracy**. Here we show the approximation errors between the input surface and the final quad meshes generated by different methods. The error is measured from each sampled point on the quad mesh to the input surface.

via a sparse linear system without integer variables. Both methods focus on efficient computation of directional fields, with Power Fields [Knöppel et al. 2013] optimizing smoothness and PolyVectors [Diamanti et al. 2014] generalizing traditional field representations. Fig. 15 compares the quadrilateral meshes generated by our NeurCross and these methods. NeurCross not only aligns with principal curvatures but also preserves overall smoothness.

Comparison with IM. IM [Jakob et al. 2015] is an effective method for generating quad meshes. To facilitate a fair comparison between IM and our approach, we use the same global seamless parameterization and extraction technique (libQEx [Ebke et al. 2013]) to extract the quad mesh. As shown in Fig. 16, our method produces fewer singularities than IM. Additionally, our method outperforms IM [Jakob et al. 2015] in terms of principal direction alignment and structural integrity, as illustrated in the close-up views.

Comparison with Quad Remesher. Quad Remesher [Remesher 2019] excels at generating quadrilateral meshes and is available as a plugin for software like Blender. It is stable, efficient, and effective at preserving model features while maintaining topological uniformity, with our method achieving comparable results. However, its performance depends heavily on the quality of the input mesh, producing low-quality quadrilateral meshes when the input polygonal mesh is suboptimal (see Fig. 17).

Fidelity. In practical applications, when converting a shape from a triangular mesh to a quadrilateral mesh representation, the goals extend beyond minimizing area distortion, angle distortion, and the number of singular points; maintaining fidelity to the original shape is also crucial. Recognizing that a low-resolution quad mesh may naturally lose some details, we use various methods to generate a quad mesh containing 25,000 vertices and 50,000 faces for a more detailed comparison.

In Fig. 18, we present the approximation errors between the quad meshes generated by five methods and the input triangle mesh. The

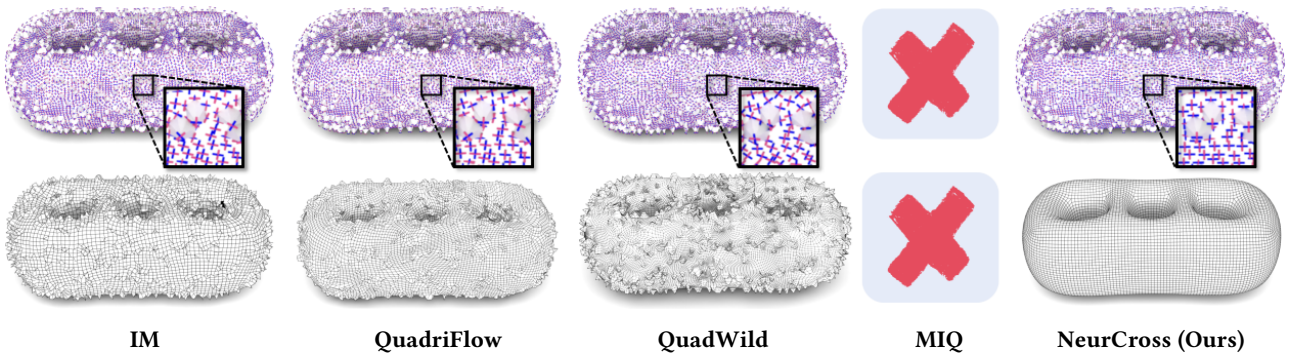


Fig. 19. The top row shows the cross field generated by our method and four other methods on a noisy input mesh. The bottom row shows the resulting quad meshes produced by each approach. Note that MIQ fails to produce a valid result for this input surface with noise.

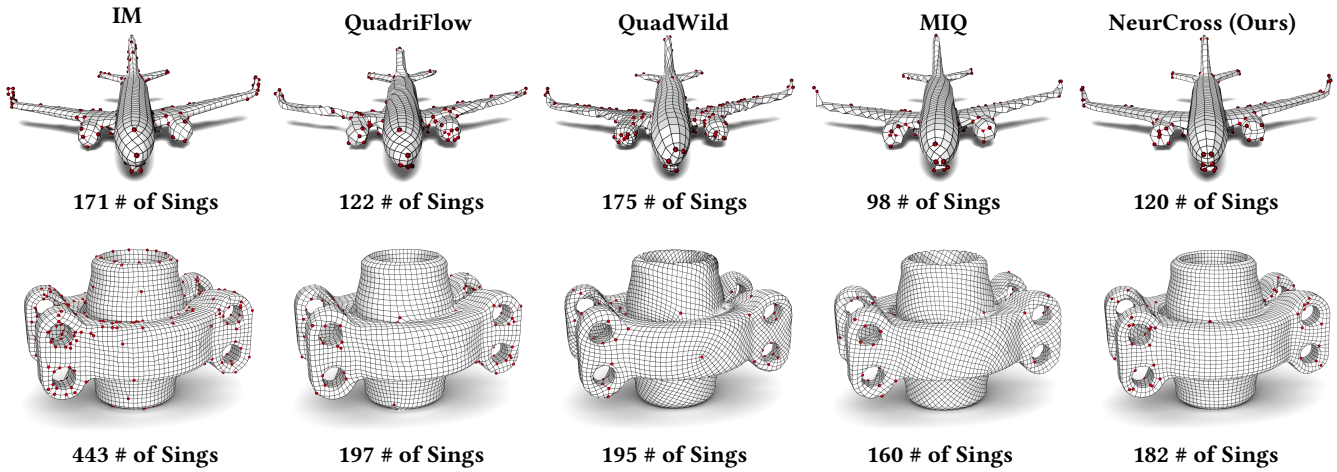


Fig. 20. Quad meshes generated by all the methods on two models from ShapeNet [Chang et al. 2015] (the airplane model) and Thingi10K [Zhou and Jacobson 2016] (the grayloc model). We also show the locations of singular points, where "# of Sings" denotes the number of singular points on each quad mesh.

quad meshes generated by our NeurCross and MIQ [Bommes et al. 2009] faithfully represent the original input. IM [Jakob et al. 2015] and QuadriFlow [Huang et al. 2018] exhibit minor shape distortions, whereas QuadWild [Pietroni et al. 2021] produces a smoother result, leading to a loss of detail.

Resistance to Noise. As noted in Wang et al. [2023], Wang et al. [2024], and Dong et al. [2024], the Hessian matrix possesses intrinsic smoothing properties. Benefiting from this characteristic, our method demonstrates inherent resistance to noise in cross field prediction. We used a baseline mesh with 15,000 vertices and introduced Gaussian noise (i.e., 2% relative to the normal direction of each model) to test the noise immunity of our NeurCross. For a comprehensive comparison, we evaluated the four other methods under the same noise conditions.

In Fig. 19, we present the results of different methods under noisy input. Notably, our approach optimizes the SDF and the cross field simultaneously. As a result, during optimization, the underlying SDF naturally smooths out noise, leading to a more intuitive cross field. In summary, our method demonstrates stronger noise resistance compared to four other methods.

Singular Points. It's well acknowledged that a trade-off must be achieved between reducing singular points and aligning with principal directions. Thus, it's preferable to position singular points in regions with high curvature variation rather than in flatter areas. As observed in Tab. 2, Tab. 3, and Fig. 20, our method produces a slightly higher number of singular points compared to MIQ [Bommes et al. 2009]. This occurrence can be attributed to MIQ's tendency to produce distorted quadrilaterals, which consequently reduces the occurrence of singular points as well as area and angular distortions. However, MIQ's quad mesh lacks overall consistency and tends to oversmooth areas with significant changes in the direction of the cross field.

In Fig. 20, we visualize the locations of singular points in the quad meshes generated by all methods on two models. The placement of singular points in the quad mesh generated by our method is more reasonable, and the resulting quadrilateral mesh exhibits high overall consistency.

Geometrically Complex Models. Various complex geometric models, such as triangular meshes with high genus, thin shells, or non-orientable surfaces, are common in many fields. In Fig. 21, we display

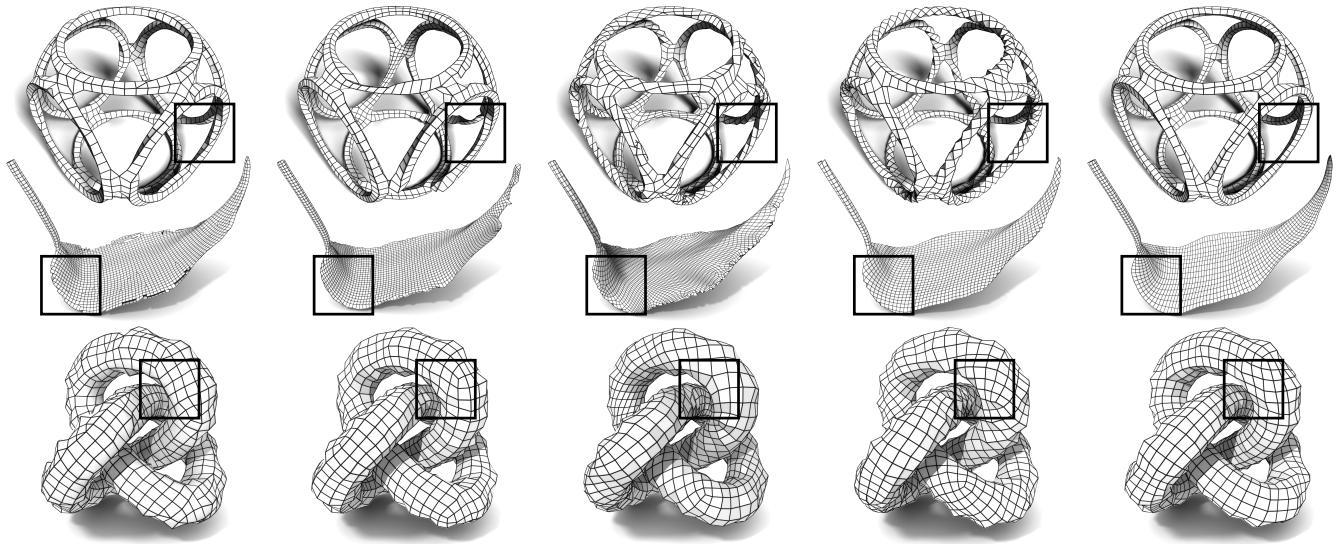


Fig. 21. Comparison of quad meshes generated by various methods for some challenging models, i.e. with high genus, thin shells, and non-orientable rings. Across all tests, the quad meshes generated by NeurCross consistently exhibit higher quality compared to those produced by other methods.

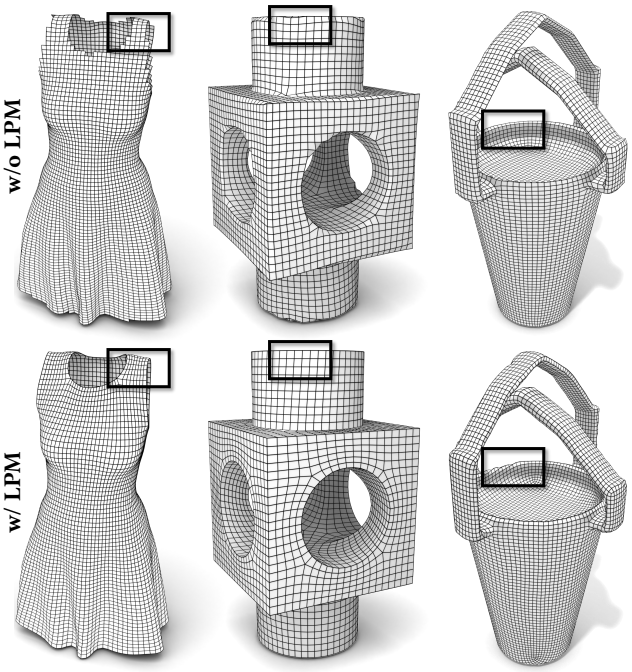


Fig. 22. Quad meshes extracted by NeurCross using different mesh extraction methods for various models: (a) A garment with open boundaries; (b) A CAD model with feature lines; and (c) A free-form model. The results are presented for each extraction method with or without the localized patching mechanism (LPM).

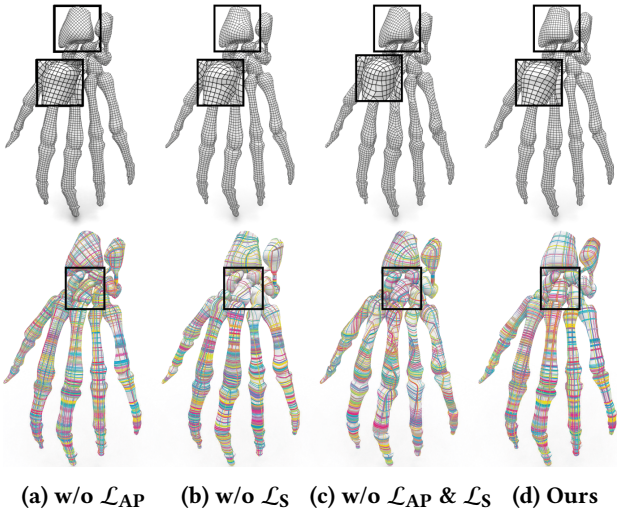


Fig. 23. The quad meshes and cross fields generated by our NeurCross using various loss term combinations: (a) without the alignment with principal directions loss term (w/o $\mathcal{L}_{AP}$); (b) without the smoothness loss term (w/o $\mathcal{L}_{S}$); (c) without both (w/o $\mathcal{L}_{AP}$ & $\mathcal{L}_{S}$); and (d) with both (Ours).

the quad meshes generated by our method and other methods on several geometrically complex models. The visualization results show that our method's performance on the unoriented ring model is comparable to that of IM [Jakob et al. 2015] and QuadriFlow [Huang et al. 2018]. However, for the other two models, only our method consistently produces high-quality quad meshes. Specifically, on the model with a thin shell (the leaf model), only our method and MIQ [Bommes et al. 2009] managed to avoid surface damage. While MIQ produced distorted quadrilaterals at the boundary of the thin shell, our method maintained good overall consistency in the quadrilateral meshes. Fig. 26 shows more results generated by our NeurCros on challenging models.

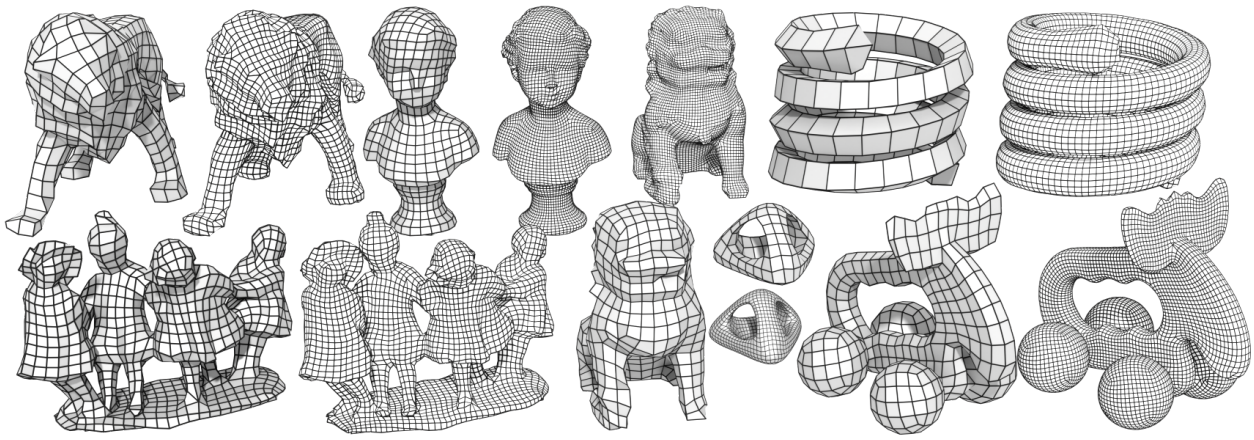


Fig. 24. Quad meshes generated by our NeurCross at different resolutions. The low-resolution models contain fewer than 1,000 vertices, while the high-resolution models consist of over 5,000 vertices.

Table 4. Ablation studies on the alignment with principal directions loss term $\mathcal{L}_{\text{AP}}$ and the smoothness loss term $\mathcal{L}_{\text{S}}$.

		Area ↓	Angle ↓	# of Sings ↓	CD ↓	JR ↑
ShapeNet [Chang et al. 2015]	w/o $\mathcal{L}_{\text{AP}}$	1.59	11.96	89.96	8.05	0.75
	w/o $\mathcal{L}_{\text{S}}$	1.96	15.12	113.28	8.09	0.71
	w/o $\mathcal{L}_{\text{AP}}$ & $\mathcal{L}_{\text{S}}$	2.25	20.73	238.71	8.15	0.55
	NeurCross (Ours)	**1.48**	**9.85**	**85.32**	**8.03**	**0.78**
Thingi10K [Zhou and Jacobson 2016]	w/o $\mathcal{L}_{\text{AP}}$	1.48	11.89	73.79	8.25	0.79
	w/o $\mathcal{L}_{\text{S}}$	1.87	15.03	105.37	8.29	0.73
	w/o $\mathcal{L}_{\text{AP}}$ & $\mathcal{L}_{\text{S}}$	2.21	20.67	225.18	8.31	0.58
	NeurCross (Ours)	**1.33**	**9.68**	**68.96**	**8.22**	**0.81**

5 ABLATION STUDIES

5.1 Extraction Methods

As discussed in Section 4.1, the global parameterization techniques in libigl [Jacobson et al. 2017] fail to align parameterized lines with sharp feature lines. To address this, we adopt QuadWild [Pietroni et al. 2021], leveraging the marked sharp features from Sec. 3.3 to divide the surface into patches using the localized patching mechanism (LPM) [Pietroni et al. 2021], and using our cross field to guide the patch tessellation process.

As illustrated in the bottom row of Fig. 22, NeurCross can successfully generate feature-aligned quadrilateral meshes, which is particularly beneficial for CAD models. For free-form models, the localized patching mechanism (LPM) [Pietroni et al. 2021] introduces singularities at the junctions of adjacent patches and even produces malformed quadrilaterals. Therefore, we generally rely on the global parameterization methods from libigl [Jacobson et al. 2017], unless the user explicitly requires the alignment of parameterized lines with sharp feature lines, in which case we employ the localized patching mechanism (LPM) [Pietroni et al. 2021].

5.2 Cross Field Loss Terms

To further highlight the efficacy of our cross field loss terms in quad mesh generation, we conducted a comparative analysis by disabling these loss terms. We used the ShapeNet [Chang et al. 2015] and Thingi10K [Zhou and Jacobson 2016] datasets for testing and comparison, setting the weight λ_{AP} of the alignment with principal directions loss term, the weight λ_{S} of the smoothness loss term, or both, to zero, while keeping other settings unchanged.

Fig. 23 illustrates the quadrilateral meshes and cross field generated by our method under various loss term combinations. The results show that our method produces the highest quality quadrilateral meshes. Disabling the alignment with principal directions term $\mathcal{L}_{\text{AP}}$ maintains only local correlation and lacks overall consistency. Although the mesh generated without the smoothness term $\mathcal{L}_{\text{S}}$ shows some degree of overall consistency, it is prone to producing singular points due to the absence of constraints on the local cross field. Without constraints from neither $\mathcal{L}_{\text{AP}}$ nor $\mathcal{L}_{\text{S}}$, the resulting quadrilateral mesh exhibits both aforementioned defects. The quantitative results presented in Tab. 4 align with the qualitative findings in Fig. 23, further demonstrating the superiority of our method in generating quadrilateral meshes.

5.3 Resolution of Quad Mesh

In real-world applications, selecting the appropriate resolution for quad mesh extraction depends on the specific requirements of different tasks. In Fig. 24, we use the same cross field for both low- and high-resolution quad meshes, ensuring consistent placement of singular points. Interestingly, the low-resolution mesh better highlights the positioning of these singular points. Fig. 24 demonstrates that, in our approach, most singular points are strategically located in regions with high curvature rather than in flat areas.

6 LIMITATION

A significant limitation of the self-supervised optimization is its substantial time requirement. For a triangular mesh input with 50,000 faces, each iteration takes 68.34 ms, with a default setting of 10,000 iterations. However, for geometrically simple and regular shapes, NeurCross typically converges in fewer iterations to produce high-quality quadrilateral meshes (see the top row of Fig. 25), whereas

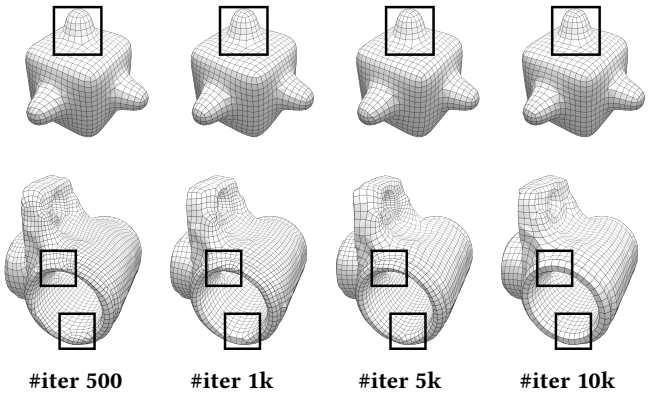


Fig. 25. Trend of convergence. Quad meshes generated by our NeurCross with different numbers of iterations (#iter).

complex shapes may require additional iterations to achieve comparable results (see the bottom row of Fig. 25).

A promising future direction is to leverage this approach to generate ample training data for feeding generative models, such as MeshGPT [Siddiqui et al. 2024]. This would enable users to obtain high-quality quad meshing outcomes instantly.

7 CONCLUSION

In this paper, we propose a self-supervised neural representation of the cross field for quadrilateral mesh generation. To the best of our knowledge, this is the first self-supervised approach for this task. Our network, named NeurCross, consists of two modules: one to fit the SDF and another to predict the cross field. The design of our loss function addresses three key aspects: surface approximation quality, alignment with principal directions, and the spatial smoothness of the cross field. Leveraging our network, the SDF and cross field are optimized simultaneously, achieving a desirable balance between approximation accuracy and cross field smoothness. Experimental results consistently validate improvements in singular point placement and in the approximation accuracy between the input triangular surface and the output quad mesh.

ACKNOWLEDGMENTS

The authors would like to thank the anonymous reviewers for their valuable comments and suggestions. This work was supported by the National Key R&D Program of China (2022YFB3303200), the National Natural Science Foundation of China (U23A20312, 62272277, 62102380), the Shandong Provincial Natural Science Foundation (ZR2024MF083), the Innovation and Technology Commission of the HKSAR Government under the InnoHK initiative (TransGP project) and the ITSP-Platform grant (Ref: ITS/335/23FP), and the Research Grants Council of Hong Kong (Ref: 17210222).

REFERENCES

Yizhak Ben-Shabat, Chamin Hewa Koneputugodage, and Stephen Gould. 2022. DiGS: Divergence guided shape implicit neural representation for unoriented point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR).*

Federica Bogo, Javier Romero, Matthew Loper, and Michael J. Black. 2014. FAUST: Dataset and Evaluation for 3D Mesh Registration. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*. 3794–3801.

David Bommes, Marcel Campen, Hans-Christian Ebke, et al. 2013a. Integer-Grid Maps for Reliable Quad Meshing. *ACM Transactions on Graphics* 32, 4 (2013), 1–12.

David Bommes, Bruno Lévy, Nico Pietroni, et al. 2013b. Quad-Mesh Generation and Processing: A Survey. *Computer Graphics Forum* 32, 6 (2013), 51–76.

David Bommes, Henrik Zimmer, and Leif Kobbelt. 2009. Mixed-Integer Quadrangulation. *ACM Transactions on Graphics* 28, 3 (2009), 1–10.

Alexandre Boulch and Renaud Marlet. 2022. POCO: Point Convolution for Surface Reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR).*

Marcel Campen, David Bommes, and Leif Kobbelt. 2015a. Quantized Global Parametrization. *ACM Transactions on Graphics* 34, 6 (2015), 1–12.

Marcel Campen, David Bommes, and Leif Kobbelt. 2015b. Quantized global parametrization. *ACM Trans. Graph.* 34, 6 (2015), Article 192.

Jonathan C Carr, Richard K Beatson, Jon B Cherrie, Tim J Mitchell, W Richard Fright, Bruce C McCallum, and Tim R Evans. 2001. Reconstruction and representation of 3D objects with radial basis functions. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques.*

Angel X. Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. 2015. *ShapeNet: An Information-Rich 3D Model Repository.* Technical Report arXiv:1512.03012 [cs.GR].

E. Chien, Z. Levi, and O. Weber. 2016. Bounded distortion parametrization in the space of metrics. *ACM Transactions on Graphics (TOG)* 35, 6 (2016), 215.

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

Alexander Dielen, Isaak Lim, Max Lyon, and Leif Kobbelt. 2021. Learning direction fields for quad mesh generation. In *Computer Graphics Forum*, Vol. 40. Wiley Online Library, 181–191.

Qiujie Dong, Rui Xu, Pengfei Wang, Shuangmin Chen, Shiqing Xin, Xiaohong Jia, Wenping Wang, and Changhe Tu. 2024. NeurCADRecon: Neural Representation for Reconstructing CAD Surfaces by Enforcing Zero Gaussian Curvature. *ACM Transactions on Graphics* (2024).

S. Dong, P.-T. Bremer, M. Garland, V. Pascucci, and J. C. Hart. 2006. Spectral surface quadrangulation. *ACM Transactions on Graphics (TOG)* 25 (2006), 1057–1066.

H.-C. Ebke, D. Bommes, M. Campen, and L. Kobbelt. 2013. QEx: Robust quad mesh extraction. *ACM Transactions on Graphics (TOG)* 32, 6 (2013), 168.

Hans-Christian Ebke, Patrick Schmidt, Marcel Campen, and Leif Kobbelt. 2016. Interactively Controlled Quad Remeshing of High Resolution 3D Models. *ACM Transactions on Graphics* 35, 6 (Nov. 2016), 218:1–218:13.

Philipp Erler, Paul Guerrero, Stefan Ohrhallinger, Niloy J Mitra, and Michael Wimmer. 2020. Points2Surf: learning implicit surfaces from point clouds. In *Proceedings of the European Conference on Computer Vision (ECCV).*

Xianzhong Fang, Hujun Bao, Yiyin Tong, et al. 2018. Quadrangulation through Morse-Parameterization Hybridization. *ACM Transactions on Graphics* 37, 4 (2018).

Amos Gropp, Lior Yariv, Niv Haim, Matan Atzmon, and Yaron Lipman. 2020. Implicit Geometric Regularization for Learning Shapes. In *Proceedings of the International Conference on Machine Learning (ICML).*

T. Gurung, D. Laney, P. Lindstrom, and J. Rossignac. 2011. SQuad: Compact representation for triangle meshes. In *Computer Graphics Forum*, Vol. 30. 355–364.

Kaiming He, X. Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2016), 770–778.

Aaron Hertzmann and Denis Zorin. 2000. Illustrating smooth surfaces. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '00).* ACM Press/Addison-Wesley Publishing Co., USA, 517–526.

Fei Hou, Chiyu Wang, Wencheng Wang, Hong Qin, Chen Qian, and Ying He. 2022. Iterative Poisson Surface Reconstruction (IPSR) for Unoriented Points. *ACM Trans. Graph.* 41, 4 (2022), 13 pages.

Shimin Hu, Zheng-Ning Liu, Meng-Hao Guo, Junxiong Cai, Jiahui Huang, Tai-Jiang Mu, and Ralph Robert Martin. 2021. Subdivision-based Mesh Convolution Networks. *ACM Transactions on Graphics (TOG)* 41 (2021), 1 – 16.

Jiahui Huang, Hao-Xiang Chen, and Shi-Min Hu. 2022. A Neural Galerkin Solver for Accurate Surface Reconstruction. *ACM Trans. Graph.* 41, 6 (2022), 16 pages.

Jingwei Huang, Yichao Zhou, Matthias Niessner, et al. 2018. QuadriFlow: A Scalable and Robust Method for Quadrangulation. *Computer Graphics Forum* (2018).

A. Jacobson, D. Panozzo, and et al. 2017. libigl: A simple C++ geometry processing library. http://libigl.github.io/libigl/.

Wenzel Jakob, Marco Tarini, Daniele Panozzo, and Olga Sorkine-Hornung. 2015. Instant field-aligned meshes. *ACM Transactions on Graphics* 34, 6 (2015), 189.

Michael Kazhdan and Hugues Hoppe. 2013. Screened Poisson surface reconstruction. *ACM Transactions on Graphics (ToG)* 32, 3 (2013), 1–13.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization.

Felix Knöppel, Keenan Crane, Ulrich Pinkall, and Peter Schröder. 2013. Globally optimal direction fields. *ACM Trans. Graph.* 32, 4 (2013), Article 59.

Felix Kälberer, Matthias Niesser, and Konrad Polthier. 2007. QuadCover – Surface Parameterization using Branched Coverings. *Computer Graphics Forum* 26, 3 (2007).

Y.-K. Lai, M. Jin, X. Xie, Y. He, J. Palacios, E. Zhang, S.-M. Hu, and X. Gu. 2010. Metric-driven RoSy field design and remeshing. *IEEE Transactions on Visualization and Computer Graphics* 16, 1 (2010), 95–108.

Y.-K. Lai, L. Kobbelt, and S.-M. Hu. 2008. An incremental approach to feature aligned quad dominant remeshing. In *Proceedings of the 2008 ACM Symposium on Solid and Physical Modeling*. ACM, 137–145.

Z. Levi and D. Zorin. 2014. Strict minimizers for geometric optimization. *ACM Transactions on Graphics (TOG)* 33, 6 (2014), 185.

R. Ling, J. Huang, B. Jüttler, F. Sun, H. Bao, and W. Wang. 2014. Spectral quadrangulation with feature curve alignment and element size control. *ACM Transactions on Graphics (TOG)* 34, 1 (2014), 11.

Yaron Lipman. 2021. Phase Transitions, Distance Functions, and Implicit Neural Representations. In *Proceedings of the International Conference on Machine Learning (ICML)*.

Max Lyon, Marcel Campen, David Bommes, and Leif Kobbelt. 2019. Parametrization Quantization with Free Boundaries for Trimmed Quad Meshing. *ACM Transactions on Graphics* 38, 4 (2019), 1–14.

Baorui Ma, Zhizhong Han, Yu-Shen Liu, and Matthias Zwicker. 2021. Neural-Pull: Learning Signed Distance Functions from Point Clouds by Learning to Pull Space onto Surfaces. In *Proceedings of the International Conference on Machine Learning (ICML)*.

Baorui Ma, Liu Yu-Shen, Zwicker Matthias, and Han Zhizhong. 2022. Surface Reconstruction from Point Clouds by Learning Predictive Context Priors. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Francesco Milano, Antonio Loquercio, Antoni Rosinol, Davide Scaramuzza, and Luca Carlone. 2020. Primal-dual mesh convolutional neural networks. *Advances in Neural Information Processing Systems* 33 (2020), 952–963.

Tai-Jiang Mu, Hao-Xiang Chen, Jun-Xiong Cai, and Ning Guo. 2023. Neural 3D reconstruction from sparse views using geometric priors. *Computational Visual Media* 9, 4 (2023), 687–697.

A. Myles, N. Pietroni, and D. Zorin. 2014. Robust field-aligned global parametrization. *ACM Transactions on Graphics (TOG)* 33, 4 (2014), 135.

A. Myles and D. Zorin. 2013. Controlled-distortion constrained global parametrization. *ACM Transactions on Graphics (TOG)* 32, 4 (2013), 105.

G.M. Nielson. 2004. Dual marching cubes. In *IEEE Visualization 2004.* 489–496.

S. J. Owen, M. L. Staten, S. A. Canann, and S. Saigal. 1999. Q-Morph: An indirect approach to advancing front quad meshing. *Internat. J. Numer. Methods Engrg.* 44, 9 (1999), 1317–1340.

David R Palmer, Oded Stein, and Justin M. Solomon. 2021. Frame Field Operators. *Computer Graphics Forum* 40 (2021). https://api.semanticscholar.org/CorpusID:235658269

Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. 2019. DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Nico Pietroni, Stefano Nuvoli, Thomas Alderighi, Paolo Cignoni, and Marco Tarini. 2021. Reliable feature-line driven quad-remeshing. *ACM Trans. Graph.* 40, 4 (2021), Article 155.

N. Ray, B. Vallet, W. C. Li, and B. Lévy. 2008. n-symmetry direction field design. *ACM Transactions on Graphics (TOG)* 27, 2 (2008), 10.

J.-F. Remacle, J. Lambrechts, B. Seny, E. Marchandise, A. Johnen, and C. Geuzainet. 2012. Blossom-Quad: A non-uniform quadrilateral mesh generator using a minimum-cost perfect-matching algorithm. *Internat. J. Numer. Methods Engrg.* 89, 9 (2012), 1102–1119.

Quad Remesher. 2019. Quad Remesher. https://exoside.com/

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. U-net: Convolutional networks for biomedical image segmentation. In *Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18*. Springer, 234–241.

Yawar Siddiqui, Antonio Alliegro, Alexey Artemov, Tatiana Tommasi, Daniele Sirigatti, Vladislav Rosov, Angela Dai, and Matthias Nießner. 2024. Meshgpt: Generating triangle meshes with decoder-only transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 19615–19625.

Vincent Sitzmann, Julien N.P. Martel, Alexander W. Bergman, David B. Lindell, and Gordon Wetzstein. 2020. Implicit Neural Representations with Periodic Activation Functions. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Amir Vaxman, Marcel Campen, Olga Diamanti, et al. 2016. Directional Field Synthesis, Design, and Processing. *Computer Graphics Forum* 35, 2 (2016), 545–572.

L. Velho and D. Zorin. 2001. 4–8 subdivision. *Computer Aided Geometric Design* 18, 5 (2001), 397–427.

Peng-Shuai Wang, Yang Liu, and Xin Tong. 2022a. Dual octree graph networks for learning adaptive volumetric shape representations. *ACM Trans. Graph.* 41, 4, Article 103 (jul 2022), 15 pages.

Ruian Wang, Zixiong Wang, Yunxiao Zhang, Shuangmin Chen, Shiqing Xin, Changhe Tu, and Wenping Wang. 2024. Aligning Gradient and Hessian for Neural Signed Distance Function. *Advances in Neural Information Processing Systems* 36 (2024).

Yifan Wang, Lukas Rahmann, and Olga Sorkine-Hornung. 2022b. Geometry-consistent neural shape representation with implicit displacement fields. In *The Tenth International Conference on Learning Representations*. OpenReview.

Yifan Wang, Shihao Wu, Cengiz Oztireli, and Olga Sorkine-Hornung. 2021. Iso-points: Optimizing neural implicit surfaces with hybrid representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 374–383.

Zixiong Wang, Yunxiao Zhang, Rui Xu, Fan Zhang, Peng-Shuai Wang, Shuangmin Chen, Shiqing Xin, Wenping Wang, and Changhe Tu. 2023. Neural-Singular-Hessian: Implicit Neural Representation of Unoriented Point Clouds by Enforcing Singular Hessian. *ACM Trans. Graph.* 42, 6 (dec 2023).

Rui Xu, Zixiong Wang, Zhiyang Dou, Chen Zong, Shiqing Xin, Mingyan Jiang, Tao Ju, and Changhe Tu. 2022. RFEPS: Reconstructing Feature-line Equipped Polygonal Surface. *ACM Transactions on Graphics (TOG)* (2022), 15 pages.

M. Zhang, J. Huang, X. Liu, and H. Bao. 2010. A wave-based anisotropic quadrangulation method. *ACM Transactions on Graphics (TOG)* 29, 4 (2010), 118.

Paul Zhang, Josh Vekhter, Edward Chien, David Bommes, Etienne Vouga, and Justin Solomon. 2020. Octahedral Frames for Feature-Aligned Cross Fields. *ACM Trans. Graph.* 39, 3, Article 25 (April 2020), 13 pages.

Qingnan Zhou and Alec Jacobson. 2016. Thingi10K: A Dataset of 10,000 3D-Printing Models. *arXiv preprint arXiv:1605.04797* (2016).

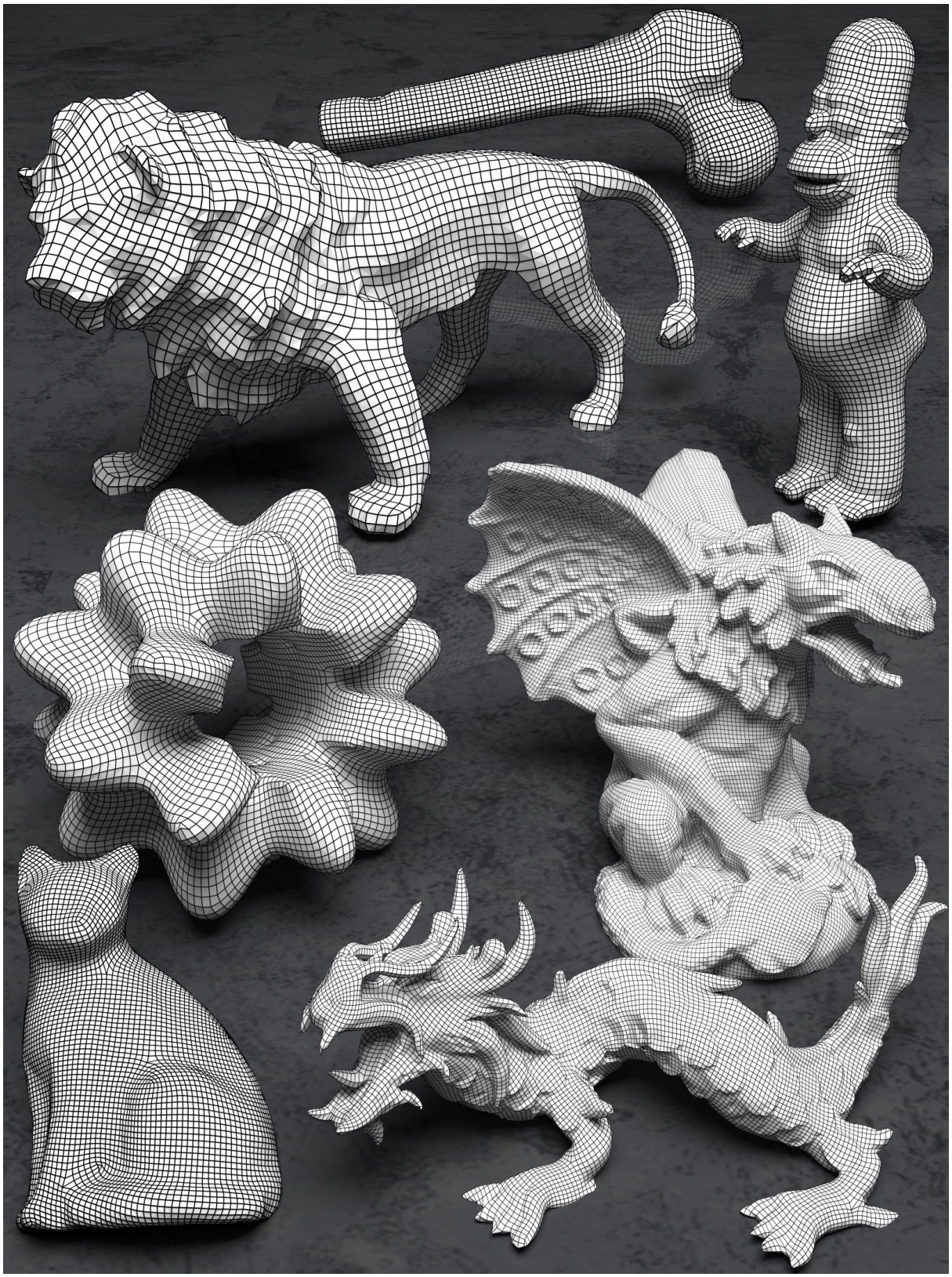


Fig. 26. The quad meshes produced by our NeurCross method on challenging models.