Frontiers of Information Technology & Electronic Engineering
www.jzus.zju.edu.cn; engineering.cae.cn; www.springerlink.com
ISSN 2095-9184 (print); ISSN 2095-9230 (online)
E-mail: jzus@zju.edu.cn

Neural mesh refinement*

Zhiwei ZHU[1,2], Xiang GAO[1,2], Lu YU‡[1,2], Yiyi LIAO‡[1,2]

[1]*College of Information Science and Electronic Engineering, Zhejiang University, Hangzhou 310027, China*
[2]*Zhejiang Provincial Key Laboratory of Information Processing, Communication and Networking (IPCAN), Hangzhou 310027, China*

E-mail: zhuzhiwei21@zju.edu.cn; xiangg@zju.edu.cn; yul@zju.edu.cn; yiyi.liao@zju.edu.cn

Received Apr. 30, 2024; Revision accepted Sept. 18, 2024; Crosschecked Apr. 21, 2025

Abstract: Subdivision is a widely used technique for mesh refinement. Classic methods rely on fixed manually defined weighting rules and struggle to generate a finer mesh with appropriate details, while advanced neural subdivision methods achieve data-driven nonlinear subdivision but lack robustness, suffering from limited subdivision levels and artifacts on novel shapes. To address these issues, this paper introduces a neural mesh refinement (NMR) method that uses the geometric structural priors learned from fine meshes to adaptively refine coarse meshes through subdivision, demonstrating robust generalization. Our key insight is that it is necessary to disentangle the network from non-structural information such as scale, rotation, and translation, enabling the network to focus on learning and applying the structural priors of local patches for adaptive refinement. For this purpose, we introduce an intrinsic structure descriptor and a locally adaptive neural filter. The intrinsic structure descriptor excludes the non-structural information to align local patches, thereby stabilizing the input feature space and enabling the network to robustly extract structural priors. The proposed neural filter, using a graph attention mechanism, extracts local structural features and adapts learned priors to local patches. Additionally, we observe that Charbonnier loss can alleviate over-smoothing compared to L2 loss. By combining these design choices, our method gains robust geometric learning and locally adaptive capabilities, enhancing generalization to various situations such as unseen shapes and arbitrary refinement levels. We evaluate our method on a diverse set of complex three-dimensional (3D) shapes, and experimental results show that it outperforms existing subdivision methods in terms of geometry quality. See https://zhuzhiwei99.github.io/NeuralMeshRefinement for the project page.

Key words: Geometry processing; Mesh refinement; Mesh subdivision; Disentangled representation learning; Neural network; Graph attention
https://doi.org/10.1631/FITEE.2400344 **CLC number:** TP391

1 Introduction

Three-dimensional (3D) meshes are extensively used in computer graphics and computer vision, playing a vital role in applications such as virtual reality and 3D reconstruction. Although detailed 3D meshes can model complex details, they put a strain on storage and computation. Coarse 3D meshes are favored for efficient storage and quick transmission, but they struggle to accurately represent fine geometric details. To tackle this trade-off, a practical solution is to store and transmit a coarse mesh and then use a refinement method at the user end to restore the details. We study the task of mesh refinement, in which the goal is to refine a coarse triangle mesh into a finer geometric structure with appropriate details.

Classic subdivision methods are highly suitable for interactive mesh refinement. They typically

‡ Corresponding authors
* Project supported by the National Natural Science Foundation of China (Nos. 62071427 and U21B2004)
ORCID: Zhiwei ZHU, https://orcid.org/0009-0005-3960-1828; Lu YU, https://orcid.org/0000-0002-0550-7754; Yiyi LIAO, https://orcid.org/0000-0001-6662-3022

involve two steps: initially, the input mesh elements are divided into smaller ones (e.g., splitting a triangle into four), followed by adjusting the positions of mesh vertices using a weighting scheme based solely on local mesh connectivity. However, these methods fall short of adaptively upsampling a low-resolution mesh. Without user guidance, classic methods tend to overly smooth the entire shape, leading to shrinkage, or amplify tessellation artifacts (Fig. 1). Classic methods based on fixed "one-size-fits-all" weighting rules are determined for their general convergence, and they fail to identify the details that need to be preserved or enhanced during upsampling.

Nonlinear subdivision schemes have been introduced to eliminate artifacts, preserve shape, and generate smooth curves on manifolds (Schaefer et al., 2008). Previous nonlinear approaches take advantage of nonlinearity to generate a much wider class of functions than linear subdivision algorithms. Compared to manually designed nonlinear functions, neural networks can be used to significantly expand the space of geometric details created with a subdivision scheme. Liu HTD et al. (2020) first used a neural network to achieve nonlinear geometry-aware subdivision, effectively avoiding over-smoothing and contraction issues. However, its robustness is insufficient. During inference, when encountering unseen shapes or subdivision levels not observed during training, the subdivision results tend to generate bulging artifacts, potentially compromising the structural integrity of the shape (Fig. 1).

We propose a neural mesh refinement (NMR) method that demonstrates robust generalization. Our key insight is that it is necessary to disentangle the network from non-structural information such as scale, rotation, and translation, allowing the network to focus on learning and applying the structural priors of local patches for adaptive refinement. Based on this insight, we design an intrinsic structure descriptor that excludes non-structural information to align local patches, stabilizing the input feature space and ensuring that the network accurately extracts intrinsic structural features. We propose a neural filter that performs nonlinear filtering following linear interpolation, particularly incorporating a graph attention aggregation module that facilitates the fusion of significant structural features and dynamically adapts to the local regions of different

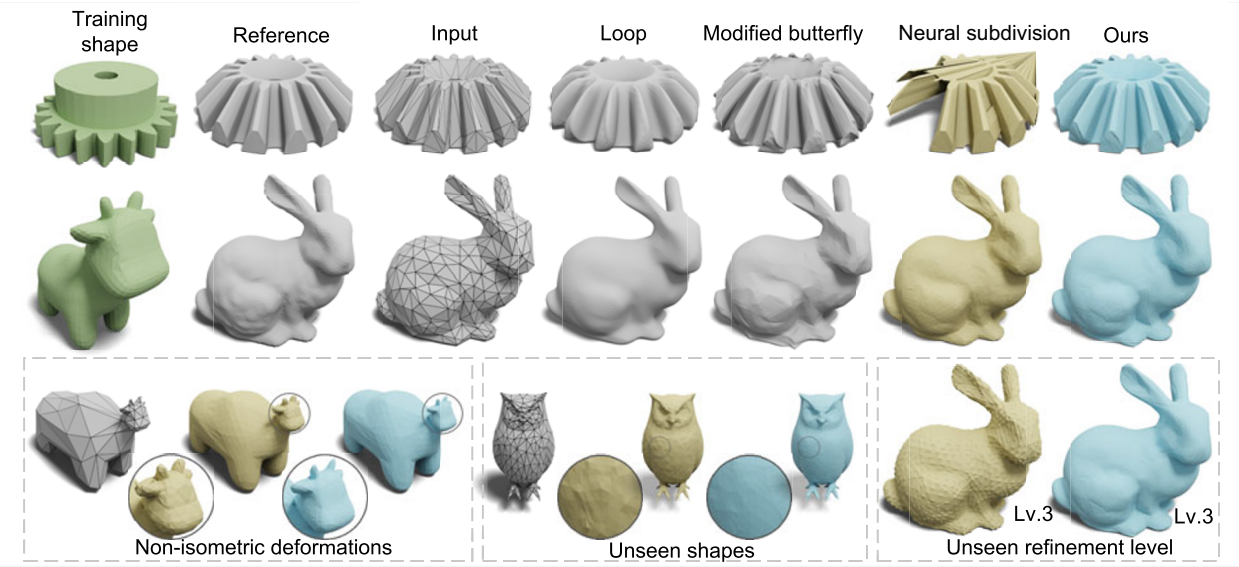

Fig. 1 Neural mesh refinement (NMR) performs data-driven nonlinear refinement and demonstrates robust generalization. It can refine coarse input shapes into finer ones with appropriate geometric details through subdivision, even when trained on a single object. NMR does not suffer from the inherent limitations of existing methods, such as volume shrinkage and over-smoothing (Loop (Loop, 1987)), amplification of tessellation artifacts (modified butterfly (Zorin et al., 1996)), or shape damage (neural subdivision (Liu HTD et al., 2020)). Moreover, it outperforms neural subdivision in generalization across non-isometric deformations, unseen shapes, and unseen refinement levels. Throughout this paper, the refinement results displayed are at level 2 unless specified otherwise

objects. Additionally, we notice that the L2 loss penalizes larger errors more severely but is more tolerant of smaller errors, often leading to overly smooth results. To address the issue of over-smoothing, we propose to use the Charbonnier loss to constrain vertex correspondences. Jointly, these design choices bestow our method with robust geometric learning and locally adaptive capabilities, enhancing its generalization performance.

We evaluate our method through diverse qualitative and quantitative experiments on the TOSCA (Bronstein et al., 2009) and Thingi10K (Zhou and Jacobson, 2016) datasets, showing substantial improvements over baseline approaches. Our method produces refined meshes closer to true high-resolution shapes compared to traditional and data-driven methods. We illustrate common applications of our approach in low-polygon mesh upsampling and 3D modeling. We demonstrate its ability to generalize to new shapes even when trained on just one pair, a capability lacking in other data-driven subdivision methods (Fig. 2).

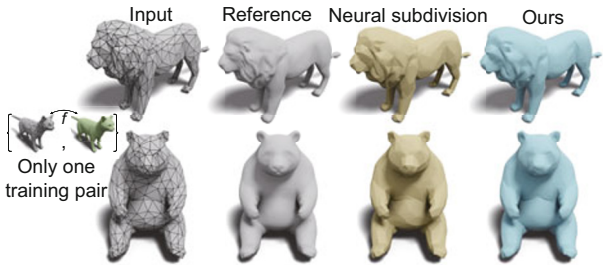

Fig. 2 When the training set consists of only one training pair, neural subdivision (Liu HTD et al., 2020) overfits and cannot learn geometric patterns for generalization. In contrast, NMR can learn geometric patterns to generalize to novel shapes

In summary, our work makes the following contributions:

1. We propose neural mesh refinement (NMR), which combines an intrinsic structure descriptor with graph attention aggregation. This allows NMR to learn rich structural priors during training and then adaptively apply priors during inference to refine coarse shapes into finer ones according to their geometry.

2. Extensive experiments demonstrate that our method exhibits quantitative and qualitative advantages compared to baseline methods. Our method proves to be robust and applicable to various situations, such as unseen shapes, unseen poses, arbitrary refinement levels, and non-isometric transformations. In addition, the geometric structure learning capability allows it to exhibit the potential for geometric style transfer.

2 Related works

This section aims to establish connections with related works. We survey the literature relevant to our NMR. Our work leverages insights from the domains of mesh subdivision, mesh neural networks, and deep learning-based signal upsampling to achieve adaptive mesh refinement.

2.1 Mesh subdivision

Mesh subdivision is characterized by deterministic recursive upsampling of discrete meshes, which is a classical geometric modeling approach. Pioneered by the Doo–Sabin algorithm (Doo and Sabin, 1998), techniques such as the linear approach for triangle meshes (Loop, 1987) and the quad mesh production method (Catmull and Clark, 1998) have been widely used. Control factors were introduced by Hoppe et al. (1994) and DeRose et al. (1998) to preserve sharp edges or semi-sharp creases. Guiding subdivisions were introduced (Levin, 2006; Karčiauskas and Peters, 2007) to achieve specific properties. Subdivision methods have been extensively studied from a theoretical standpoint (Stam, 1998; Zorin, 2007). Linear subdivision methods, as pointed out by Oliensis (1993) and Taubin (1995), essentially involve Gaussian filtering, leading to inevitable volume shrinkage. To address this, nonlinear subdivision schemes have been introduced (Dyn, 2006; Schaefer et al., 2008; Vaxman et al., 2018) to preserve shape, eliminate artifacts, and generate smooth curves on manifolds.

Instead of manually designing nonlinear functions, neural networks have been used to significantly expand the space of geometric details created with a subdivision scheme (Hertz et al., 2020). Liu HTD et al. (2020) proposed neural subdivision, a data-driven nonlinear subdivision method beyond classic linear methods; however, it suffers from scale dependence, which leads to the lack of robustness, manifested as problems such as the tendency to produce artifacts on novel shapes and can work only properly at limited subdivision levels. Our approach, akin

to neural subdivision, learns and applies local geometric priors based on normalized one-ring neighbors, avoiding scale dependency. Furthermore, we integrate the graph attention mechanism to adapt to diverse shapes. Recently, Liu K et al. (2023) introduced the implicit neural distance as the loss function of neural subdivision for better convergence. Chen YC et al. (2023) introduced per-face features encoded from the original high-resolution mesh to assist neural subdivision in restoring details at the decoder side. Their contributions are orthogonal to ours, and we could extend our network in theory to incorporate these methods.

2.2 Neural networks on meshes

The irregularity and non-uniformity of mesh data pose challenges for directly applying neural networks to mesh structures. As a result, most methods transform the mesh data into regular inputs to make use of the learning strategy. Since triangle meshes are structured graphs, various methods introduce the graph neural network to mesh processing (Monti et al., 2018; Veličković et al., 2018; Wang NY et al., 2018; Dai A and Nießner, 2019; Chen Y et al., 2020; Milano et al., 2020). While some methods extend the classical Fourier analysis to the 3D mesh, transforming data from the spatial domain to the spectral domain (Atwood and Towsley, 2016; Levie et al., 2019; Smirnov and Solomon, 2021; Sharp et al., 2022), some methods transform the mesh into the image domain, enabling the use of convolutional neural networks (CNNs) for learning and feature extraction (Maron et al., 2017; Poulenard and Ovsjanikov, 2018; Tatarchenko et al., 2018; Haim et al., 2019; Huang et al., 2019; Yang et al., 2019).

Another category of methods focuses on developing network models that use mesh structure, which can be further categorized into vertex-, edge-, and face-based approaches. Lim I et al. (2019) proposed spiral convolution patterns within the one-ring neighborhood of vertices. MeshCNN (Hanocka et al., 2019) learns edge features and performs pooling based on edge collapsing operations. MeshWalker (Lahav and Tal, 2020) extracts shape features by walking along edges. MeshNet (Feng et al., 2019) learns face features by aggregating spatial and structural features from the one-ring neighbors through two mesh convolution layers. Abutbul et al. (2020) proposed to learn face features from a local patch to denoise mesh normals. Hertz et al. (2020) used three-face convolution and subdivision to achieve geometry texture transfer. Hu SM et al. (2022) introduced a face-based CNN framework for triangle meshes with Loop subdivision sequence connectivity. Perez et al. (2023) proposed a mesh convolutional network that combines face- and vertex-based operations.

Our method is similar to the mesh structure-based geometry learning approach. Leveraging our proposed intrinsic structure descriptor, NMR can efficiently extract edge features within a one-ring neighborhood and then use the graph attention mechanism to aggregate these edge features, ultimately obtaining the central vertex features.

2.3 Upsampling

Mesh subdivision, which can be regarded as mesh upsampling, belongs to a subfield of upsampling. Researching methods in upsampling can inspire the work on subdivision surfaces. In the following, we briefly introduce deep learning-based image upsampling and point cloud upsampling approaches.

2.3.1 Image upsampling

Deep image upsampling can be categorized based on the model framework: pre-upsampling (Dong et al., 2016b), post-upsampling (Dong et al., 2016a), iterative upsampling and downsampling (Haris et al., 2018), and progressive upsampling (Lai et al., 2017). Progressive upsampling offers advantages such as better model training, easier convergence, and the ability to output multi-resolution results in a single inference. In terms of network architecture design, image super-resolution methods include residual learning (Jiao et al., 2017; Lim B et al., 2017), recursive learning (Kim J et al., 2016; Tai et al., 2017), dense connection networks (Wang XT et al., 2018), and attention mechanisms (Dai T et al., 2019). As for training strategies, Lai et al. (2017) demonstrated that the Charbonnier loss (Charbonnier et al., 1994), a smooth approximation of the L1 loss, performs better in detail preservation and convergence compared to the L2 loss. Additionally, the use of multi-supervision (Wang YF et al., 2018) helps avoid the issues of gradient explosion and vanishing, thereby facilitating network convergence.

For a more detailed introduction to image upsampling, it is recommended to read a few previous

articles (Yang et al., 2019; Wang ZH et al., 2021; Chen HG et al., 2022). Our method leverages the framework of progressive upsampling and residual learning, incorporating multi-supervision Charbonnier loss as a training technique to enhance performance in surface upsampling.

2.3.2 Point cloud upsamling

Building on the success of PointNet (Qi et al., 2017), Yu et al. (2018) introduced the first deep learning-based point cloud upsampling method. Wang YF et al. (2019) proposed a multi-step point cloud upsampling network (MPU) that performs progressive upsampling. Li et al. (2019) introduced generative adversarial networks (GANs) to point cloud upsampling. Furthermore, Qian et al. (2021) proposed node shuffle to improve performance. However, the above-mentioned upsampling methods treat each point independently, failing to capture the geometric relationship within the input point cloud. To address this, Wang Y et al. (2019) and Qian et al. (2021) dynamically associated vertices and their neighborhoods, constructing a graph and applying graph convolutional networks (GCNs). Moreover, Han et al. (2022) and Kim D et al. (2023) proposed attention mechanisms to assign different attentional weights dynamically for combining edge features.

In contrast to point clouds, which need manual processes such as ball query and K-nearest neighbors (KNN) for neighbor identification and graph construction, meshes already have a built-in graph structure. This feature makes it easier and more effective to use graph neural networks. Our approach expands on the idea of gathering local information in point cloud upsampling and uses the natural graph structure of meshes to design modules for embedding edge features and aggregating features through graph attention mechanisms.

3 Preliminary analysis

In this section, we aim to illustrate the motivation for disentangling non-structural information from the network through theoretical and preliminary experimental analyses.

The geometric characteristics of a mesh are closely tied to the local surface features. These features can be grouped into structural and non-structural aspects. Structural features involve the shapes of faces and the angles between them, while non-structural features include the scale, rotation, and translation of the faces. The influence of non-structural features on geometric details can be simply expressed through affine transformation. In contrast, the relationship between structural features and geometric details exhibits more complex patterns, which is why classic methods with fixed one-size-fits-all rules fail to perform adaptive refinement. The powerful expressive capability of neural networks offers an opportunity to learn the geometric priors hidden within existing 3D models. Consequently, we believe that disentangling the network from non-structural information will help the network focus exclusively on extracting structural features, ultimately improving the subdivision results.

Liu HTD et al. (2020) demonstrated that disentangling rotation and translation information makes neural subdivision invariant to rigid motions, leading to a significant enhancement in quality when compared to methods lacking this invariance. However, neural subdivision lacks scale invariance. Including scale information in the input features requires the network to learn geometric priors across different scales. Limited scale scenarios in the training set can cause overfitting, resulting in poor generalization to patches with new scales during testing. This limitation is the main reason for artifacts in neural subdivision when applied to new shapes or beyond the trained subdivision level. One potential solution is to increase the scale diversity of patches in the training set; however, this can lead to a divergence in the input feature space, making it challenging for the network to learn geometric priors and resulting in mediocre outcomes. A more effective approach involves directly eliminating scale information from the network's input features by scale normalization.

The preliminary experimental results, as shown in Fig. 3, indicate that incorporating scale normalization in neural subdivision allows for convergence to the same accuracy with fewer iterations or parameters. This confirms our key insight that disentangling the network from scale enables it to focus on extracting structural features, alleviating learning pressure and speeding up convergence. While the introduction of scale normalization leads to significant performance enhancements, previous methods still face challenges such as limited receptive field in

feature extraction, non-unified coordinate systems during feature fusion, and reduction of feature response through naive average pooling. Therefore, we propose NMR to address these issues, whose details are presented in the subsequent section.

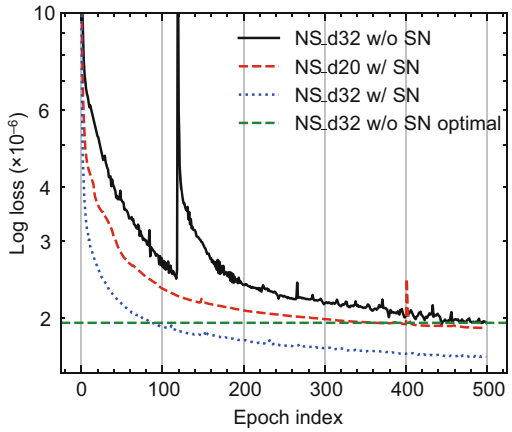

Fig. 3 Disentangling the network from scale can alleviate its learning pressure, allowing it to converge to the same accuracy with fewer iterations or parameters. Here, NS represents neural subdivision, the number following "d" denotes the dimension of the hidden layer, and SN indicates scale normalization

4 Methods

Our method processes a coarse triangle mesh (gray) and produces a series of subdivided meshes (blue) with varying levels of details (LoDs) (Fig. 4). Initially, we apply midpoint subdivision to insert new vertices along each edge. Subsequently, we create an intrinsic structure descriptor by normalizing the one-ring neighborhood of the newly inserted vertices. The neural filter then extracts features based on these descriptors and predicts offsets for the new vertices, updating their positions.

To prepare the training data, we scale each mesh in the dataset to fit a unit cube and follow the successive self-parameterization (Liu HTD et al., 2020) method to generate meshes M^0, M^1, $\cdots$, M^L of different resolutions from the original high-resolution mesh M. The vertex positions of M^0, M^1, $\cdots$, M^L are determined by projecting them onto the original mesh M to obtain the ground-truth location. During training, the network takes the lowest-resolution mesh M^0 as the input, recursively conducting subdivision and vertex repositioning to produce multiple refinement versions $\hat{M}^1$, $\hat{M}^2$, $\cdots$, $\hat{M}^L$. We optimize our network using the Charbonnier loss (Charbonnier et al., 1994), which measures the distance between each predicted vertex position of $\hat{M}^l$ and its corresponding vertex on M^l (Fig. 4).

4.1 Topology update

The irregularity and non-uniformity of mesh data pose challenges for applying neural networks to mesh structures. To tackle this, we update the topology of coarse meshes in two steps to ensure that the subdivided mesh has a regular topology, facilitating subsequent network design. First, we ensure that all input meshes are edge-manifold without boundary edges (each edge is adjacent to two faces) through an optional preprocessing step. If a mesh does not meet this condition, such as a non-watertight, self-intersecting mesh, we preprocess it using the method proposed by Hu YX et al. (2020). Subsequently, our method uses midpoint interpolation to subdivide the mesh, adding new vertices at the midpoint of each edge, connecting the new vertices through new edges, and dividing the original face into four faces. These steps guarantee that each newly inserted vertex has

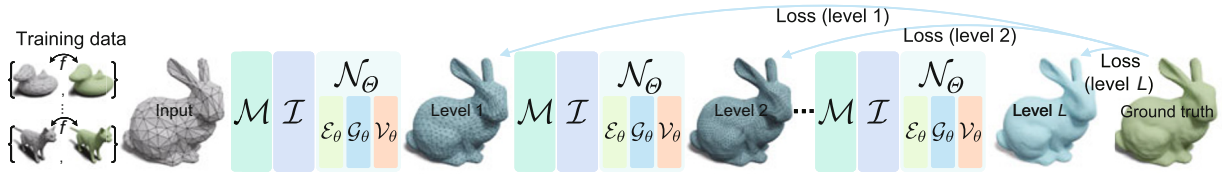

Fig. 4 NMR processes a coarse triangle mesh (gray) to produce a sequence of subdivided meshes (blue) with varying levels of details. During training, we minimize the Charbonnier loss between the ground truth (green) and the output meshes (blue) across levels. Our training data comprise pairs of coarse and fine meshes (left) with a bijective map f between each pair. Each refinement involves three steps: $\mathcal{M}$ for midpoint subdivision, $\mathcal{I}$ for intrinsic structure descriptor construction, and $\mathcal{N}_\Theta$ for the neural filter. $\mathcal{N}_\Theta$ consists of $\mathcal{E}_\theta$ (edge feature embedding module), $\mathcal{G}_\theta$ (graph attention aggregation module), and $\mathcal{V}_\theta$ (vertex repositioning module). References to color refer to the online version of this figure

six neighbors, as depicted in Fig. 5. The uniform neighbor count of the new vertices provides a stable graph structure, inspiring the design of our intrinsic structure descriptor.

4.2 Intrinsic structure descriptor

The intrinsic structure descriptor is a crucial part of our framework, aligning local patches and allowing the network to concentrate on extracting intrinsic structural features agnostic to non-structural information. It extracts vertex features of the one-ring neighborhood of each newly inserted vertex by normalizing their coordinates to the unit sphere of the one-ring custom local coordinate system (Fig. 5).

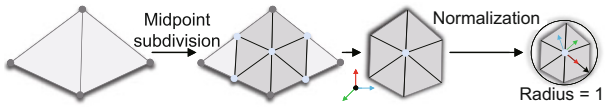

Fig. 5 Topology update rule and intrinsic structure descriptor construction

To provide sufficient local information for reflecting surface shape trends and broadening the receptive field, our method uses three sets of coordinates to represent the geometric information of each vertex $\boldsymbol{v}^i$ within the one-ring neighborhood, denoted as $\boldsymbol{f}_{\mathrm{v}}^i = (\boldsymbol{p}_{\mathrm{v}}^i, \boldsymbol{l}_{\mathrm{v}}^{i(1)}, \boldsymbol{l}_{\mathrm{v}}^{i(2)}) \in \mathbb{R}^9$. Here, $\boldsymbol{p}_{\mathrm{v}}^i \in \mathbb{R}^3$ represents the coordinates of each vertex $\boldsymbol{v}^i$, $\boldsymbol{l}_{\mathrm{v}}^{i(1)} \in \mathbb{R}^3$ represents the first-order Laplacian coordinates (Sorkine et al., 2004) indicating the curvature of the vertex, and $\boldsymbol{l}_{\mathrm{v}}^{i(2)} \in \mathbb{R}^3$ represents the second-order Laplacian coordinates indicating the rate of change of the vertex's curvature. For computational convenience and to let Laplacian coordinates scale identically across different orders, we use uniform Laplacian coordinates, which can be expressed as follows:

$$\boldsymbol{l}_{\mathrm{v}}^{i(n+1)} = \boldsymbol{l}_{\mathrm{v}}^{i(n)} - \frac{1}{d_i} \sum_{j \in \mathcal{N}_i} \boldsymbol{l}_{\mathrm{v}}^{j(n)}, \tag{1}$$

where d_i is the valency of vertex $\boldsymbol{v}^i$ and $\mathcal{N}_i$ is its one-ring neighborhood. When calculating the first-order Laplacian coordinates, $\boldsymbol{l}_{\mathrm{v}}^{i(0)}$ equals $\boldsymbol{p}_{\mathrm{v}}^i$.

To ensure that vertex features remain invariant to affine transformation, we transform the features of each vertex within the one-ring neighborhood to the unit sphere of the one-ring custom local coordinate system. Specifically, we set the central vertex $\boldsymbol{v}^i$ as the origin of the local coordinate system. The x axis is determined by the longest edge, with its length as the scaling factor s^i. The longest edge normal, computed by averaging the normals of its two adjacent faces, serves as the z axis. The y axis is determined by the cross-product of the previously mentioned axes. The transformation can be formulated as follows:

$$\tilde{\boldsymbol{p}}_{\mathrm{v}}^{ij} = (s^i)^{-1} \boldsymbol{R}^i \times (\boldsymbol{p}_{\mathrm{v}}^j - \boldsymbol{p}_{\mathrm{v}}^i), \tag{2}$$

$$\tilde{\boldsymbol{l}}_{\mathrm{v}}^{ij(1)} = (s^i)^{-1} \boldsymbol{R}^i \times \boldsymbol{l}_{\mathrm{v}}^{j(1)}, \tag{3}$$

$$\tilde{\boldsymbol{l}}_{\mathrm{v}}^{ij(2)} = (s^i)^{-1} \boldsymbol{R}^i \times \boldsymbol{l}_{\mathrm{v}}^{j(2)}, \tag{4}$$

where $j \in \{i\} \cup \mathcal{N}_i$, and $\boldsymbol{R}^i$ is the rotation matrix, each row of which represents the unit vector in the direction of each coordinate axis. Finally, we obtain the normalized vertex feature $\tilde{\boldsymbol{f}}_{\mathrm{v}}^{ij} = \{\tilde{\boldsymbol{p}}_{\mathrm{v}}^{ij}, \tilde{\boldsymbol{l}}_{\mathrm{v}}^{ij(1)}, \tilde{\boldsymbol{l}}_{\mathrm{v}}^{ij(2)}\}$ of $\boldsymbol{v}^j$ within the one-ring neighborhood whose central vertex is $\boldsymbol{v}^i$.

4.3 Neural filter

The proposed neural filter ($\mathcal{N}_\Theta$) extracts structural features based on the intrinsic structure descriptor and combines the learned local pattern priors to restore local geometric details. To alleviate learning pressure, our approach first learns the structural features on edges and then aggregates these features to the central vertex. To adaptively emphasize the most relevant edge features based on the local structure, we use a graph attention mechanism. Specifically, $\mathcal{N}_\Theta$ comprises three main modules (Fig. 6): edge feature embedding ($\mathcal{E}_\theta$), graph attention aggregation ($\mathcal{G}_\theta$), and vertex repositioning ($\mathcal{V}_\theta$). $\mathcal{E}_\theta$ captures edge features from the outgoing edges of the central vertex, $\mathcal{G}_\theta$ aggregates edge features to the central vertex using graph attention, and $\mathcal{V}_\theta$ predicts the residual from the central vertex feature to update its position.

4.3.1 Edge feature embedding

The edge feature embedding ($\mathcal{E}_\theta$) module employs a half-flap operator. This operator is applied to each half-edge adjacent to the central vertex $\boldsymbol{v}_i$ to extract edge features $\hat{\boldsymbol{f}}_{\mathrm{e}}^{ij} \in \mathbb{R}^C$, where C denotes the dimensionality of these edge features. Half-flap, a two-face flap adjacent to a half-edge, is commonly used in the mesh processing method (Hanocka et al., 2019; Liu HTD et al., 2020). Since a half-edge is a directed edge, it provides a unique canonical orientation for the four vertices at the corners of adjacent

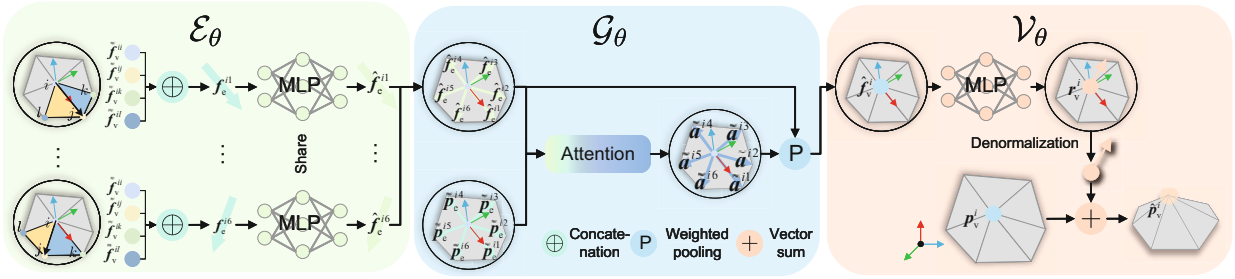

Fig. 6 Neural filter ($\mathcal{N}_\Theta$) comprises three main modules: edge feature embedding ($\mathcal{E}_\theta$), graph attention aggregation ($\mathcal{G}_\theta$), and vertex repositioning ($\mathcal{V}_\theta$). $\mathcal{E}_\theta$ captures edge features from the outgoing edges of the central vertex, $\mathcal{G}_\theta$ aggregates edge features to the central vertex using graph attention, and finally $\mathcal{V}_\theta$ predicts the residual from the central vertex feature to update its position

faces, denoted as $(\boldsymbol{v}^i, \boldsymbol{v}^j, \boldsymbol{v}^k, \boldsymbol{v}^l)$ (Fig. 6). We first concatenate the four vertex features of each half-flap in sequence to obtain the input edge features $\boldsymbol{f}_{\mathrm{e}}^{ij}$. Then we feed $\boldsymbol{f}_{\mathrm{e}}^{ij}$ into a shallow multi-layer perceptron (MLP), denoted as M_{e}, to extract the latent edge features $\hat{\boldsymbol{f}}_{\mathrm{e}}^{ij}$:

$$\hat{\boldsymbol{f}}_{\mathrm{e}}^{ij} = M_{\mathrm{e}}(\boldsymbol{f}_{\mathrm{e}}^{ij}) = M_{\mathrm{e}}(\tilde{\boldsymbol{f}}_{\mathrm{v}}^{ii} \oplus \tilde{\boldsymbol{f}}_{\mathrm{v}}^{ij} \oplus \tilde{\boldsymbol{f}}_{\mathrm{v}}^{ik} \oplus \tilde{\boldsymbol{f}}_{\mathrm{v}}^{il}), \quad (5)$$

where $\oplus$ means concatenation. Here, we remove the first three components of $\tilde{\boldsymbol{f}}_{\mathrm{v}}^{ii}$ as they are always (0, 0, 0).

4.3.2 Graph attention aggregation

The uniform neighbor count of the newly inserted vertices inherently forms a directed graph $G(V, E)$ with seven vertices and six directed edges. The $\mathcal{G}_\theta$ module uses the graph attention mechanism to aggregate the six edge features, generating the features of the newly inserted vertices, denoted as $\hat{\boldsymbol{f}}_{\mathrm{v}}^{i} \in \mathbb{R}^C$ (Fig. 6). This mechanism allows $\mathcal{G}_\theta$ to adapt to various local structures, emphasizing the most relevant neighbors while diminishing the influence of the less relevant ones, thereby enhancing feature aggregation.

$\mathcal{G}_\theta$ considers both spatial and feature relevance. It combines the six edge spatial vectors $\tilde{\boldsymbol{p}}_{\mathrm{e}}^{ij} = \tilde{\boldsymbol{p}}_{\mathrm{v}}^{ij}$ and edge feature vectors $\hat{\boldsymbol{f}}_{\mathrm{e}}^{ij}$ adjacent to the central vertex $\boldsymbol{v}^i$, and learns attention weight vectors $\tilde{\boldsymbol{a}}^{ij} \in \mathbb{R}^C$ for the weighted pooling of edge feature vectors.

Specifically, we first perform channel normalization on the six edge features to establish connections between them and enhance the generalization ability of the $\mathcal{G}_\theta$ module.

$$\tilde{\boldsymbol{f}}_{\mathrm{e}}^{ij} = \left\{ \frac{\hat{f}_{\mathrm{e}}^{ij,c} - \mu_{\mathrm{e}}^{i,c}}{\sigma_{\mathrm{e}}^{i,c}} \mid c = 1, 2, ..., C \right\}, j \in \mathcal{N}_i, \quad (6)$$

where $\hat{f}_{\mathrm{e}}^{ij,c}$ is the c^{th} channel component of edge feature $\hat{f}_{\mathrm{e}}^{ij}$, $\mu_{\mathrm{e}}^{i,c}$ and $\sigma_{\mathrm{e}}^{i,c}$ are the mean and standard deviation of the six edge features of the channel with index c, respectively, and $\tilde{\boldsymbol{f}}_{\mathrm{e}}^{ij}$ is the normalized edge feature.

Then we use a two-layer MLP with leaky rectified linear unit (LeakyReLU) (with negative input slope $\alpha = 0.2$), denoted as M_η, to obtain the initial attention weight $\boldsymbol{a}^{ij} \in \mathbb{R}^C$:

$$\boldsymbol{a}^{ij} = M_\eta(\tilde{\boldsymbol{p}}_{\mathrm{e}}^{ij} \oplus \tilde{\boldsymbol{f}}_{\mathrm{e}}^{ij}). \quad (7)$$

To make attention coefficients easily comparable across different nodes (Veličković et al., 2018), the above initial attention weight $\boldsymbol{a}^{ij}$ is normalized via softmax as follows:

$$\tilde{\boldsymbol{a}}^{ij} = \left\{ \frac{\exp(a^{ij,c})}{\sum_{l \in \mathcal{N}_i} \exp(a^{il,c})} \mid c = 1, 2, ..., C \right\}, \ j \in \mathcal{N}_i. \quad (8)$$

Finally, the output features of vertex $\hat{\boldsymbol{f}}_{\mathrm{v}}^{i}$ via the $\mathcal{G}_\theta$ module can be formulated as follows:

$$\hat{\boldsymbol{f}}_{\mathrm{v}}^{i} = \sum_{j \in \mathcal{N}_i} \tilde{\boldsymbol{a}}^{ij} \cdot \hat{\boldsymbol{f}}_{\mathrm{e}}^{ij}. \quad (9)$$

4.3.3 Vertex repositioning

The $\mathcal{V}_\theta$ module is a vertex operator that inputs the vertex feature $\hat{\boldsymbol{f}}_{\mathrm{v}}^{i}$ and simply applies a shared MLP to output the vertex residual $\boldsymbol{r}_{\mathrm{v}}^{i}$ to update the vertex position. The residual would be transformed from the normalized local coordinate frame to the world coordinate frame, and then added to the original coordinates of the newly inserted vertices. It is worth pointing out that our network is scale-invariant, while our entire method is scale-dependent. The updated vertex position $\hat{\boldsymbol{p}}_{\mathrm{v}}^{i}$ can be

formulated as follows:

$$\hat{\boldsymbol{p}}_{\mathrm{v}}^{i}=\boldsymbol{p}_{\mathrm{v}}^{i}+s^{i}(\boldsymbol{R}^{i})^{-1}\times\boldsymbol{r}_{\mathrm{v}}^{i}. \tag{10}$$

4.4 Loss function

The training data generated by successive self-parameterization (Liu HTD et al., 2020) provide us with target shapes with subdivision connectivity and one-to-one correspondence with coarse mesh vertices. Consequently, we can directly use the bijective map to compute the vertex-to-vertex loss for training. Compared to vertex-to-surface metrics (e.g., chamfer distance), it not only accelerates the calculation but also mitigates issues related to incorrect matches and self-overlapping between shapes. Additionally, drawing from experiences in image upsampling (Lai et al., 2017; Wang YF et al., 2018), we suggest using the Charbonnier loss to replace the L2 loss used by Liu HTD et al. (2020), which is sensitive to outliers and can result in over-smoothing. The Charbonnier loss better preserves detailed information owing to its reduced sensitivity to outliers.

In our setup, M^0 represents the input coarse mesh and Θ denotes the set of network parameters to be optimized. Our goal is to learn a mapping function for generating a finer mesh $\hat{M}^L$ that is close to the ground-truth finer mesh M^L. We denote the vertices of the ground-truth mesh M^l by $\boldsymbol{V}^l$ and the vertices of the output finer mesh $\hat{M}^l$ by $\hat{\boldsymbol{V}}^l$; the loss function is expressed as follows:

$$\mathcal{L}(\hat{\boldsymbol{V}},\boldsymbol{V};\Theta)=\frac{1}{N}\sum_{n=1}^{N}\sum_{l=1}^{L}\rho\left(\hat{\boldsymbol{V}}_n^l-\boldsymbol{V}_n^l\right), \tag{11}$$

where $\rho(\boldsymbol{x})=\sqrt{\|\boldsymbol{x}\|_1^2+\epsilon^2}$ is the Charbonnier penalty function, N is the number of training samples in each batch, and L is the total number of refinement levels. We empirically set ϵ to 0.001.

This multi-loss approach mitigates gradient explosion and vanishing issues, promoting network convergence. Through deep supervision, the network is guided to predict vertex position residuals at different levels, resulting in the production of multi-level output meshes.

5 Experiments and analysis

We first design a series of experiments to evaluate the quantitative and qualitative performance of NMR. We start by demonstrating the robustness and generalization of our method when trained and evaluated on large datasets. Next, we show interesting applications of NMR with a range of experiments to demonstrate the qualitative property of NMR.

5.1 Experimental setup

5.1.1 Datasets

We evaluate our method on the Thingi10K (Zhou and Jacobson, 2016) and TOSCA (Bronstein et al., 2009) datasets. The Thingi10K dataset offers diverse models with interesting geometric and topological features. The TOSCA dataset comprises 80 shapes across nine categories, with each category containing one type of surface and various poses. For the Thingi10K dataset, we follow Chen YC et al. (2023) to preprocess data, sampling 6000 meshes and splitting them into training (80%), validation (10%), and test (10%) sets for experiments. We decimate each source mesh with the QSlim algorithm (Garland and Heckbert, 1997) down to 0.06 times the original face counts to obtain the coarse mesh input. Concerning the TOSCA dataset, we divide the 80 shapes into training (62), validation (9), and test (9) sets. Each source mesh is decimated to produce a coarse mesh with 450–550 faces. Due to the limited shape variety in TOSCA, we randomly decimate each mesh into 10 different coarse meshes for data augmentation. For both datasets, the highest subdivision level L is set to 3.

5.1.2 Evaluation metrics

We measure the similarity of the third-level subdivided mesh and ground truth using the Hausdorff distance (HD), chamfer distance (CD), point-to-point (P2P) distance, and point-to-surface (P2F) distance. We also adopt distance-based peak signal-to-noise ratio (PSNR) evaluation metrics, which are used in the video-based dynamic mesh coding (V-DMC) (Choi et al., 2022) drafted by the Moving Picture Experts Group (MPEG). To evaluate the objective performance of our method, we compare it with a large number of subdivision methods, including midpoint subdivision, the classic Loop (Loop, 1987), butterfly subdivision (Beatty et al., 1990), modified butterfly subdivision (Zorin et al., 1996), and neural subdivision (Liu HTD et al., 2020).

5.1.3 Implementation details

Our network is implemented in PyTorch (Paszke et al., 2019), using ReLU activation (Nair and Hinton, 2010) and the Adam optimizer (Kingma and Ba, 2015). Training spans 500 epochs with an initial learning rate of 0.002, which decays to half every 100 epochs. The intrinsic structure descriptor aligning local patches allows us to use shallow two-layer MLPs for each sub-network (Table 1). As a result, our method boasts relatively low storage and computational costs. To adapt neural subdivision for large-scale datasets, we increase the hidden layer dimension f_{h} from the original 32 to 64, enhancing the expressive power of their network.

Table 1 Hyperparameters of our sub-networks

Network layer	$\mathcal{E}_\theta$	$\mathcal{G}_\theta$	$\mathcal{V}_\theta$
f_{in}	$4\times 9-3$	$3+64$	64
f_{h1}	64	64	32
f_{h2}	64	64	16
f_{out}	64	64	3

5.2 Quantitative analysis

Our approach demonstrates robust generalization, owing to the intrinsic structure descriptor and the graph attention mechanisms. This allows our method to better accommodate diverse categories, geometric textures, poses, and local patch scales on large datasets compared to the baseline methods. To quantitatively analyze how our network generalizes to unseen shapes, we conduct quantitative analyses on the TOSCA (Bronstein et al., 2009) and Thingi10K (Zhou and Jacobson, 2016) datasets.

The results in Table 2 demonstrate that our method outperforms all baseline methods on the TOSCA dataset and exhibits significant advantages across all evaluation metrics. Fig. 7 reveals that our approach produces finer meshes closer to the ground truth, successfully addressing issues encountered in other methods, such as over-smoothing, volume shrinkage, and artifacts.

Additionally, we find that neural subdivision performs extremely poorly on the Thingi10K dataset (Table 3). This is because the Thingi10K dataset contains meshes with greater diversity compared to the TOSCA dataset. Neural subdivision struggles to converge among patches with diverse geometric characteristics. When processing meshes with closely adjacent surfaces, neural subdivision may develop self-intersection after a single subdivision step, thereby disrupting the manifold and causing subsequent

Table 2 Comparison between NMR and baselines on the TOSCA dataset

Method	HD ($\times 10^{-2}$)↓	CD ($\times 10^{-6}$)↓	P2P ($\times 10^{-6}$)↓	P2F ($\times 10^{-6}$)↓	P2P-PSNR (dB)↑	P2F-PSNR (dB)↑
Midpoint	1.407	11.212	11.743	9.698	54.412	55.292
Loop	2.441	29.645	33.211	30.744	49.912	50.264
Butterfly	1.678	6.634	6.808	4.893	56.780	58.345
Modified butterfly	1.651	6.799	7.034	5.123	56.657	58.181
Neural subdivision	1.359	5.838	6.158	4.201	57.282	59.134
Ours	**1.294**	**4.447**	**4.698**	**2.827**	**58.480**	**61.031**

The best results are in bold

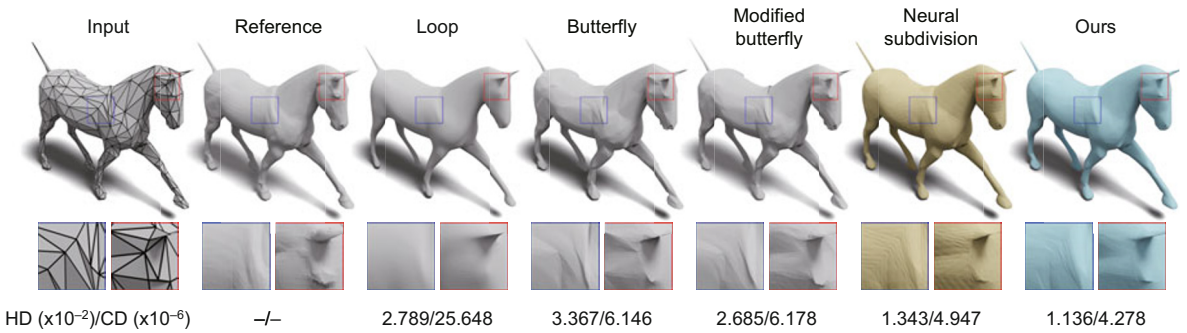

Fig. 7 Quantitative comparison between NMR and baselines on horse14 of the TOSCA dataset (Bronstein et al., 2009): our method produces finer meshes closer to the ground truth, without encountering issues faced by other methods, such as over-smoothing, volume shrinkage, and artifacts

Table 3 Comparison between NMR and baselines on the Thingi10K dataset

Method	HD ($\times10^{-2}$)↓	CD ($\times10^{-6}$)↓	P2P ($\times10^{-6}$)↓	P2F ($\times10^{-6}$)↓	P2P-PSNR (dB)↑	P2F-PSNR (dB)↑
Midpoint	3.340	5.904	7.024	6.076	50.299	51.801
Loop	6.883	31.185	44.794	41.951	42.958	43.494
Butterfly	5.824	10.184	13.226	12.278	47.188	48.067
Modified butterfly	6.227	10.946	13.963	13.067	46.693	47.412
Neural subdivision	329.582	7.717	642.590	211.493	48.672	51.355
Neural subdivision*	2.950	4.943	6.420	5.263	51.448	53.435
Ours	**2.662**	**3.605**	**4.548**	**3.870**	**53.084**	**56.347**

For a fairer comparison, the results marked as Neural subdivision* represent the statistical averages after excluding objects for which neural subdivision fails to preserve the shape of the input mesh. The best results are in bold

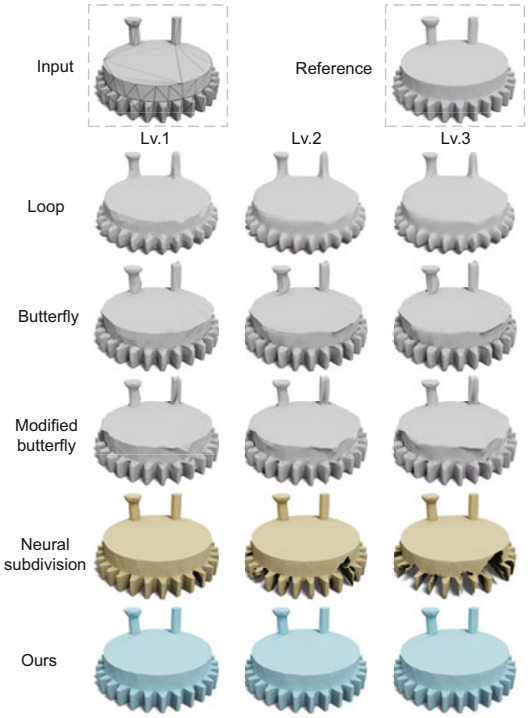

Fig. 8 Failure cases of neural subdivision when testing on the Thingi10K dataset (Zhou and Jacobson, 2016). Neural subdivision distorts and even disrupts the mesh shape. Other methods also encounter issues with volume shrinkage and amplification of tessellation artifacts. In contrast, our method robustly preserves the original shape

subdivision steps to fail. In some cases, this ultimately leads to the inability to generate reasonable results, resulting in disrupted mesh shapes (Fig. 8). In contrast, our method demonstrates robustness and does not encounter this issue. Our method maintains a significant advantage.

5.3 Qualitative analysis

We evaluate NMR using a range of experiments containing interesting applications. We demonstrate that our method can generalize to isometric deformations, non-isometric deformations, shapes from different classes, and arbitrary refinement levels.

5.3.1 Generalization to isometric deformations

In practice, modelers often adjust the coarse cage of a character into various poses and then apply the refinement operator. This situation highlights the importance of being able to train NMR on a single pose and generalize to unseen poses for character animation. In Fig. 9, we train the network on a single pose (green) and show that NMR can generalize to unseen poses under (approximately) isometric deformations.

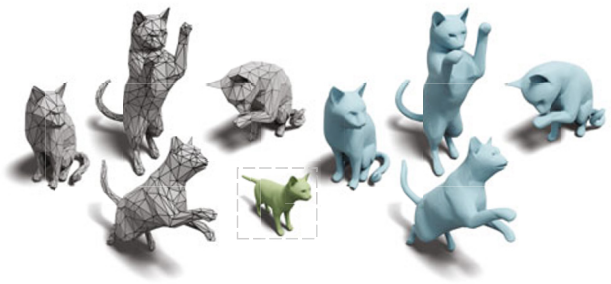

Fig. 9 Even when trained on a single pose (green), NMR can generalize to refining unseen poses (blue). References to color refer to the online version of this figure

5.3.2 Generalization to non-isometric deformations

In addition to adapting to pose changes, we manually simulate real-world scenarios by modifying the coarse cage (Fig. 10). We demonstrate that even when faced with highly exaggerated non-isometric deformations, NMR can generalize well to non-isometric deformations. However, neural subdivision produces artifacts in small regions.

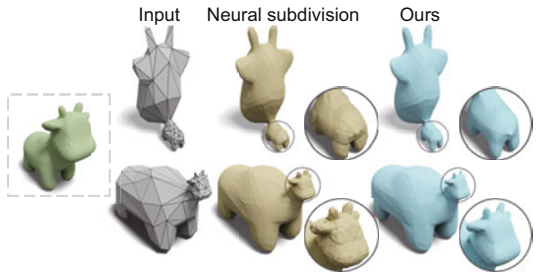

Fig. 10 We mimic the modeling scenario by applying non-isometric deformations to the coarse cage (gray). NMR generalizes well to unseen non-isometric deformations compared to neural subdivision

5.3.3 Generalization to different shapes

Refinement operators are frequently used to generate new 3D content, implying the need to generalize to various shapes. In Fig. 11, we demonstrate that when trained on a single shape, our network effectively generalizes to unseen shapes, while neural subdivision produces unexpected pseudo-artifacts.

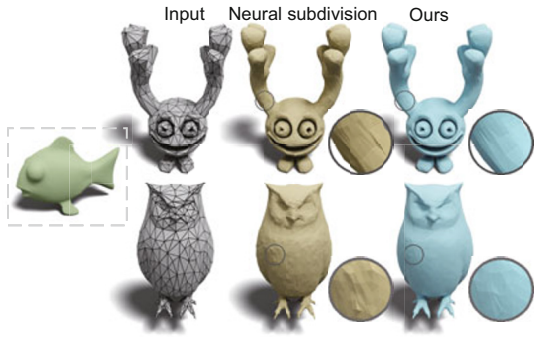

Fig. 11 Even when trained on a single shape (green fish), NMR can generalize to refining unseen shapes and outperform neural subdivision. References to color refer to the online version of this figure

5.3.4 Generalization to arbitrary refinement levels

In real-world applications, the refinement level adopted by modelers is unknown, which requires the methods used to be robust at arbitrary levels. Fig. 12 demonstrates that even when the test refinement level surpasses the training level, NMR still produces a reasonably smooth surface, while neural subdivision cannot. This is due to the tendency of neural subdivision to overfit the training scale. With the increase in the refinement level during testing, the scale of local patches decreases, posing a challenge for neural subdivision to adapt effectively. Owing to the scale-invariant intrinsic structure descriptor, our method can generalize to arbitrary refinement levels.

5.4 Analysis of refinement characteristics

As a data-driven approach, the refinement characteristics of our method are inherently linked to the composition of the training dataset. When trained on a constrained dataset with limited diversity, NMR tends to overfit and exhibit style transfer characteristics (see Fig. 13 and the top two rows of Fig. 14). As Fig. 13 illustrates, NMR can recognize and replicate the "style" of the training shape, enabling it to refine an initial coarse mesh into various distinct styles. When trained on extensive datasets with diverse geometries, NMR demonstrates an adaptive refinement capability (see the bottom row of Fig. 14), discerning where to apply smoothing and where to preserve sharp features. This capability is attributed to the intrinsic structure descriptor and graph attention aggregation, which also contribute to our approach's superior performance on large-scale datasets such as Thingi10K (Table 3), compared to existing methods.

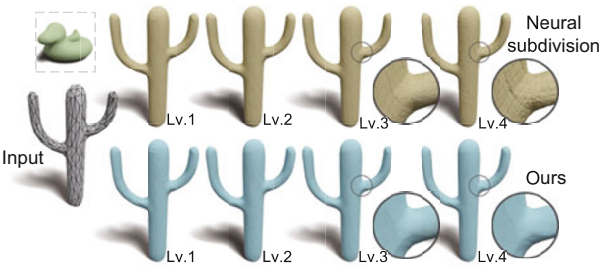

Fig. 12 Even when the test refinement level exceeds the training level, NMR can still output a reasonably smooth surface, while neural subdivision cannot

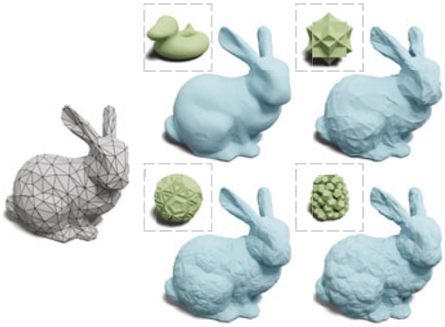

Fig. 13 When trained on a single shape, NMR exhibits style transfer characteristics. Using different shapes in training leads to stylized refinement results (blue) biased towards the training shapes (green). References to color refer to the online version of this figure

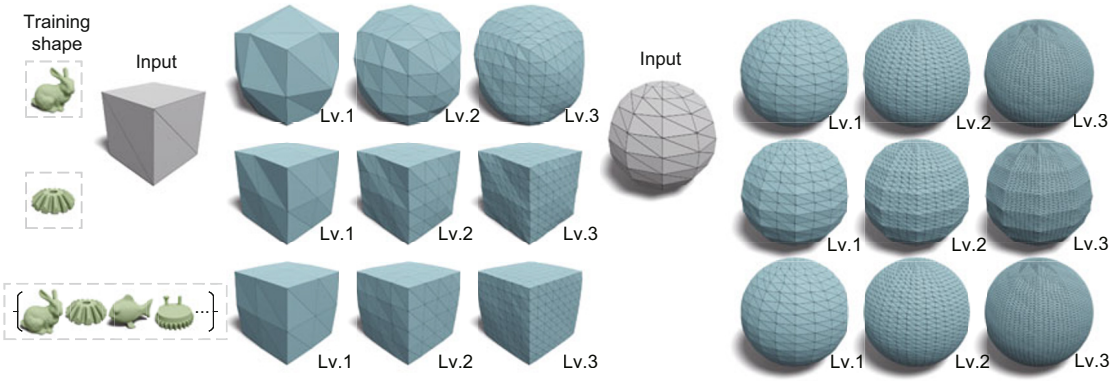

Fig. 14 Refinement characteristics of NMR depend on the composition of the training dataset. When trained on a constrained dataset with limited diversity, NMR exhibits stylized refinement ability (top two rows). When trained on extensive datasets with diverse geometries, NMR exhibits adaptive refinement ability (bottom)

5.5 Complexity analysis

Given that our network is composed solely of shallow MLPs, NMR exhibits low computational complexity. When the hyperparameters are set as shown in Table 1, the total number of network parameters is only 25 827, and the number of floating point operations (FLOPs) required for a single network inference is 20 898 150. The computational expense of NMR is determined by the count of input triangles, with the majority of time spent on the computation of the half-flap data structure. In contrast, network inference consumes much less time. Table 4 illustrates the application time for each NMR module during a single refinement process as the number of input triangles increases. Both the computation time for the half-flap structure and the data input/output (I/O) time are almost proportional to the number of triangles. The network inference time exhibits a nonlinear increase with the increase of the number of triangles, because it is affected by the parallel processing capabilities of the graphics processing unit (GPU) and the size of GPU memory. Training time is similar to application time due to the swiftness of the backpropagation process. The overall training time depends mainly on the total number of triangles across all refinement levels in the training set.

6 Ablation study

6.1 Input features

The design of input features significantly affects the performance of the network. To ensure that our network is disentangled from translation, rotation, and scale, we design vertex features $\boldsymbol{f}_{\mathrm{v}}^{i} \in \mathbb{R}^{9}$ based on intrinsic structure descriptors. These vertex features are then normalized and concatenated to obtain the input edge features $\boldsymbol{f}_{\mathrm{e}}^{ij}$. To validate the effectiveness of our proposed features, we experiment with other types of input features that possess similar transformation invariance.

1. MeshCNN: Hanocka et al. (2019) proposed the 5D features of a half-edge (two internal angles, one dihedral angle, and two edge lengths), which are widely used for mesh intelligence analysis due to their similarity transformation invariance. Here, we directly use the 5D edge features as input edge features $\boldsymbol{f}_{\mathrm{e}}^{ij}$.

2. PointNet ($\boldsymbol{p}_{\mathrm{v}}$): Qi et al. (2017) proposed to extract vertex features from vertex local coordinates. This input feature is widely used in subsequent geometry learning. In the ablation experiment, we sequentially concatenate the local coordinates of the four vertices within the half-flap to obtain the input edge features.

3. Neural subdivision ($\boldsymbol{p}_{\mathrm{v}}$ & $\boldsymbol{l}_{\mathrm{v}}^{(1)}$): Liu HTD et al. (2020) proposed that using local coordinates combined with first-order Laplacian coordinates ($\boldsymbol{l}_{\mathrm{v}}^{(1)}$) can improve convergence performance compared to using only local vertex coordinates ($\boldsymbol{p}_{\mathrm{v}}$).

4. Ours ($\boldsymbol{p}_{\mathrm{v}}$ & $\boldsymbol{l}_{\mathrm{v}}^{(1)}$ & $\boldsymbol{l}_{\mathrm{v}}^{(2)}$): Building on neural subdivision, we integrate the second-order Laplacian coordinates ($\boldsymbol{l}_{\mathrm{v}}^{(2)}$) into the vertex features. The second-order Laplacian coordinates introduce information from the two-ring neighborhood and can reflect the curvature change rate.

The experimental results in Table 5 indicate that our proposed 9D vertex features

Table 4 Application time of each module of NMR

Number of input triangles	Half-flap processing time (s)	Network inference time (s)	I/O time (s)	Total time (s)
100	0.219	0.003	0.006	0.228
400	0.939	0.004	0.009	0.952
1600	5.134	0.005	0.022	5.161
6400	30.786	0.007	0.089	30.882
25 600	141.277	0.009	0.305	141.591

Table 5 Ablation on different input features

Method	HD ($\times10^{-2}$)↓	CD ($\times10^{-6}$)↓	P2P ($\times10^{-6}$)↓	P2F ($\times10^{-6}$)↓	P2P-PSNR (dB)↑	P2F-PSNR (dB)↑
MeshCNN	1.469	11.153	11.682	9.640	54.433	55.318
PointNet ($\boldsymbol{p}_\mathrm{v}$)	1.406	7.223	7.620	5.644	56.352	57.754
Neural subdivision ($\boldsymbol{p}_\mathrm{v}$ & $\boldsymbol{l}_\mathrm{v}^{(1)}$)	1.351	4.556	4.813	2.941	58.344	60.821
Ours ($\boldsymbol{p}_\mathrm{v}$ & $\boldsymbol{l}_\mathrm{v}^{(1)}$ & $\boldsymbol{l}_\mathrm{v}^{(2)}$)	**1.294**	**4.447**	**4.698**	**2.827**	**58.480**	**61.031**

The best results are in bold

($\boldsymbol{p}_\mathrm{v}$ & $\boldsymbol{l}_\mathrm{v}^{(1)}$ & $\boldsymbol{l}_\mathrm{v}^{(2)}$) exhibit significant advantages compared to others. The poor performance of MeshCNN in our task may be attributed to manually defined edge features, leading to the loss of crucial structural information inherent in the original vertex coordinates and connectivity. Conversely, using solely local vertex coordinates disregards connectivity information. Given that the receptive field of our subsequent network comprises only the one-ring neighborhood, it becomes necessary to expand the receptive field at an early stage. Therefore, our proposed vertex features, incorporating first- and second-order Laplacian coordinates, expand the receptive field, aiding in the extraction of structural information.

6.2 Feature aggregation method

The aggregation of edge features onto vertices is equally crucial. In this subsection, we investigate various feature fusion methods and substantiate the effectiveness of our proposed graph attention aggregation module.

1. Average pooling: Average pooling, which simply calculates the average value of the features in each channel, may be the most common aggregation method.

2. Max pooling: Max pooling calculates the maximum value of features in each channel as the output. Due to its ability to preserve significant features and maintain strong neuron activation, it is widely adopted in two-dimensional (2D) and 3D deep learning frameworks.

3. Concatenation&MLP: Concatenation&MLP is an advanced aggregation method that preserves the original activation while allowing the network to combine neighboring features. In the ablation experiment, concatenation starts with the longest edge and combines the features of the other edges in a counterclockwise manner. These concatenated features are then fused through MLP to obtain the output features.

4. Graph attention aggregation: The approach we propose uses graph structures to extract channel-wise attention between different edges, achieving a vector attention map that assigns a weight to each feature element.

From the results in Table 6, we observe that max pooling has the poorest performance. We speculate that preserving only the strongest effects results in the loss of a significant amount of subtle structural information. Furthermore, directly concatenating features causes a rapid increase in the feature dimension, leading to a high number of network parameters and complicating the training process. Additionally, average pooling may weaken the strongly activated areas of the original features and reduce their distinctiveness. Our proposed graph attention aggregation module maintains or accentuates significant structural features, enabling the fusion of significant structural features and dynamically adapting to the local characteristics of different objects. It requires fewer parameters compared to the concatenation&MLP approach, but achieves better convergence performance.

Table 6 Ablation on different feature aggregation methods

Method	HD ($\times10^{-2}$)↓	CD ($\times10^{-6}$)↓	P2P ($\times10^{-6}$)↓	P2F ($\times10^{-6}$)↓	P2P-PSNR (dB)↑	P2F-PSNR (dB)↑
Average pooling	1.341	4.558	4.808	2.938	58.377	60.832
Max pooling	1.370	4.624	4.867	2.992	58.339	60.763
Concatenation&MLP	1.358	4.613	4.829	2.971	58.352	60.784
Ours	**1.294**	**4.447**	**4.698**	**2.827**	**58.480**	**61.031**

The best results are in bold

Table 7 Ablation on different loss functions

Loss function	HD ($\times10^{-2}$)↓	CD ($\times10^{-6}$)↓	P2P ($\times10^{-6}$)↓	P2F ($\times10^{-6}$)↓	P2P-PSNR (dB)↑	P2F-PSNR (dB)↑
L2	1.339	4.977	5.248	3.356	57.981	60.095
Charbonnier	**1.294**	**4.447**	**4.698**	**2.827**	**58.480**	**61.031**

The better results are in bold

6.3 Loss function

We compare the different performances resulting from L2 loss and Charbonnier loss. Experimental results in Table 7 indicate that using Charbonnier loss significantly enhances the objective quality.

7 Conclusions and future work

We propose a neural mesh refinement (NMR) method that performs adaptive refinement and exhibits robust generalization. NMR learns geometric structural priors from fine meshes during training and adaptively applies these priors based on the geometric structural features of coarse meshes during inference, resulting in refined meshes with more reasonable details. Our main contribution is that we propose and demonstrate that disentangling the network from non-structural information (e.g., scale, rotation, and translation) allows the network to focus on learning and applying structural priors for adaptive refinement. To this end, we design an intrinsic structure descriptor that aligns local patches by excluding the non-structural information and propose a graph attention aggregation module that adaptively fuses significant structural features. In addition, we use Charbonnier loss to alleviate oversmoothing. Extensive experiments demonstrate that the proposed method outperforms state-of-the-art mesh subdivision methods.

Despite demonstrating robust refinement performance across various shapes and refinement levels, our method struggles to hallucinate complex topological details missing in the input coarse mesh (Fig. 15). For finer details, global semantic features are helpful for the network to reconstruct details in local areas. As the refinement levels deepen, the number of mesh elements grows exponentially, leading to a significant increase in computational cost, while the level of detail recovery slows down. Adaptively setting the refinement level for each local region will help mitigate the gap between the level of details and the number of faces. Furthermore, extending our method to quadrilateral meshes and surfaces with boundaries will cater to a broader range of practical needs. These aspects present intriguing avenues for future research on NMR.

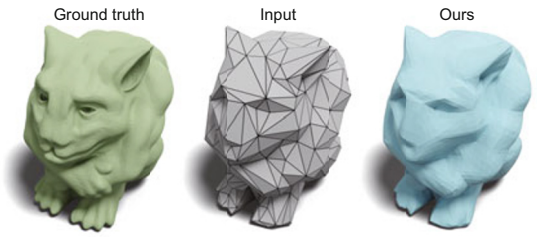

Fig. 15 Our approach is based on local geometry and thus fails to hallucinate semantic features

Contributors

Zhiwei ZHU designed the project, implemented the method, performed the experiments, and drafted the paper. Xiang GAO helped analyze the results and improved the design. Lu YU supervised the project and proposed some innovative idea. Yiyi LIAO provided the outline and revised the paper.

Conflict of interest

All the authors declare that they have no conflict of

interest.

Data availability

The evaluation code and some test data are available at https://zhuzhiwei99.github.io/NeuralMeshRefinement. The other data that support the findings of this study are available from the corresponding authors upon reasonable request.

References

Abutbul A, Elidan G, Katzir L, et al., 2020. DNF-Net: a neural architecture for tabular data. https://doi.org/10.48550/arXiv.2006.06465

Atwood J, Towsley D, 2016. Diffusion-convolutional neural networks. Proc 30th Int Conf on Neural Information Processing Systems, p.2001-2009.

Beatty JC, Dyn N, Levine D, et al., 1990. A butterfly subdivision scheme for surface interpolation with tension control. *ACM Trans Graph*, 9(2):160-169. https://doi.org/10.1145/78956.78958

Bronstein AM, Bronstein MM, Kimmel R, 2009. Numerical Geometry of Non-rigid Shapes. Springer, New York, USA. https://doi.org/10.1007/978-0-387-73301-2

Catmull E, Clark J, 1998. Recursively generated B-spline surfaces on arbitrary topological meshes. Proc Seminal Graphics: Pioneering Efforts That Shaped the Field, p.183-188. https://doi.org/10.1145/280811.280992

Charbonnier P, Blanc-Féraud L, Aubert G, et al., 1994. Two deterministic half-quadratic regularization algorithms for computed imaging. Proc 1st Int Conf on Image Processing, p.168-172. https://doi.org/10.1109/ICIP.1994.413553

Chen HG, He XH, Qing LB, et al., 2022. Real-world single image super-resolution: a brief review. *Inform Fus*, 79:124-145. https://doi.org/10.1016/j.inffus.2021.09.005

Chen Y, Zhao JY, Shi CW, et al., 2020. Mesh convolution: a novel feature extraction method for 3D nonrigid object classification. *IEEE Trans Multim*, 23:3098-3111. https://doi.org/10.1109/TMM.2020.3020693

Chen YC, Kim V, Aigerman N, et al., 2023. Neural progressive meshes. Proc ACM SIGGRAPH, Article 84. https://doi.org/10.1145/3588432.3591531

Choi Y, Jeong JB, Lee S, et al., 2022. Overview of the video-based dynamic mesh coding (V-DMC) standard work. Proc 13th Int Conf on Information and Communication Technology Convergence, p.578-581. https://doi.org/10.1109/ICTC55196.2022.9952734

Dai A, Nießner M, 2019. Scan2Mesh: from unstructured range scans to 3D meshes. Proc IEEE/CVF Conf on Computer Vision and Pattern Recognition, p.5569-5578. https://doi.org/10.1109/CVPR.2019.00572

Dai T, Cai JR, Zhang YB, et al., 2019. Second-order attention network for single image super-resolution. Proc IEEE/CVF Conf on Computer Vision and Pattern Recognition, p.11057-11066. https://doi.org/10.1109/CVPR.2019.01132

DeRose T, Kass M, Truong T, 1998. Subdivision surfaces in character animation. Proc 25th Annual Conf on Computer Graphics and Interactive Techniques, p.85-94. https://doi.org/10.1145/280814.280826

Dong C, Loy CC, Tang XO, 2016a. Accelerating the super-resolution convolutional neural network. Proc 14th European Conf on Computer Vision, p.391-407. https://doi.org/10.1007/978-3-319-46475-6_25

Dong C, Loy CC, He KM, et al., 2016b. Image super-resolution using deep convolutional networks. *IEEE Trans Patt Anal Mach Intell*, 38(2):295-307. https://doi.org/10.1109/TPAMI.2015.2439281

Doo D, Sabin M, 1998. Behaviour of recursive division surfaces near extraordinary points. Proc Seminal Graphics: Pioneering Efforts That Shaped the Field, p.177-181. https://doi.org/10.1145/280811.280991

Dyn N, 2006. Three families of nonlinear subdivision schemes. *Stud Comput Math*, 12:23-38. https://doi.org/10.1016/S1570-579X(06)80003-0

Feng YT, Feng YF, You HX, et al., 2019. MeshNet: mesh neural network for 3D shape representation. Proc 33rd AAAI Conf on Artificial Intelligence, p.8279-8286. https://doi.org/10.1609/aaai.v33i01.33018279

Garland M, Heckbert PS, 1997. Surface simplification using quadric error metrics. Proc 24th Annual Conf on Computer Graphics and Interactive Techniques, p.209-216. https://doi.org/10.1145/258734.258849

Haim N, Segol N, Ben-Hamu H, et al., 2019. Surface networks via general covers. Proc IEEE/CVF Int Conf on Computer Vision, p.632-641. https://doi.org/10.1109/ICCV.2019.00072

Han B, Zhang XY, Ren S, 2022. PU-GACNet: graph attention convolution network for point cloud upsampling. *Image Vis Comput*, 118:104371. https://doi.org/10.1016/j.imavis.2021.104371

Hanocka R, Hertz A, Fish N, et al., 2019. MeshCNN: a network with an edge. *ACM Trans Graph*, 38(4):90. https://doi.org/10.1145/3306346.3322959

Haris M, Shakhnarovich G, Ukita N, 2018. Deep back-projection networks for super-resolution. Proc IEEE/CVF Conf on Computer Vision and Pattern Recognition, p.1664-1673. https://doi.org/10.1109/CVPR.2018.00179

Hertz A, Hanocka R, Giryes R, et al., 2020. Deep geometric texture synthesis. *ACM Trans Graph*, 39(4):108. https://doi.org/10.1145/3386569.3392471

Hoppe H, DeRose T, Duchamp T, et al., 1994. Piecewise smooth surface reconstruction. Proc 21st Annual Conf on Computer Graphics and Interactive Techniques, p.295-302. https://doi.org/10.1145/192161.192233

Hu SM, Liu ZN, Guo MH, et al., 2022. Subdivision-based mesh convolution networks. *ACM Trans Graph*, 41(3):25. https://doi.org/10.1145/3506694

Hu YX, Schneider T, Wang BL, et al., 2020. Fast tetrahedral meshing in the wild. *ACM Trans Graph*, 39(4):117. https://doi.org/10.1145/3386569.3392385

Huang JW, Zhang HT, Yi L, et al., 2019. TextureNet: consistent local parametrizations for learning from high-resolution signals on meshes. Proc IEEE/CVF Conf on Computer Vision and Pattern Recognition, p.4435-4444. https://doi.org/10.1109/CVPR.2019.00457

Jiao JB, Tu WC, He SF, et al., 2017. FormResNet: formatted residual learning for image restoration. Proc IEEE Conf on Computer Vision and Pattern Recognition Workshops, p.1034-1042. https://doi.org/10.1109/CVPRW.2017.140

Karčiauskas K, Peters J, 2007. Concentric tessellation maps and curvature continuous guided surfaces. *Comput Aided Geom Des*, 24(2):99-111. https://doi.org/10.1016/j.cagd.2006.10.006

Kim D, Shin M, Paik J, 2023. PU-Edgeformer: edge transformer for dense prediction in point cloud upsampling. Proc IEEE Int Conf on Acoustics, Speech and Signal Processing, p.1-5. https://doi.org/10.1109/ICASSP49357.2023.10095541

Kim J, Lee JK, Lee KM, 2016. Deeply-recursive convolutional network for image super-resolution. Proc IEEE Conf on Computer Vision and Pattern Recognition, p.1637-1645. https://doi.org/10.1109/CVPR.2016.181

Kingma DP, Ba J, 2015. Adam: a method for stochastic optimization. Proc 3rd Int Conf on Learning Representations.

Lahav A, Tal A, 2020. MeshWalker: deep mesh understanding by random walks. *ACM Trans Graph*, 39(6):263. https://doi.org/10.1145/3414685.3417806

Lai WS, Huang JB, Ahuja N, et al., 2017. Deep Laplacian pyramid networks for fast and accurate super-resolution. Proc IEEE Conf on Computer Vision and Pattern Recognition, p.5835-5843. https://doi.org/10.1109/CVPR.2017.618

Levie R, Monti F, Bresson X, et al., 2019. CayleyNets: graph convolutional neural networks with complex rational spectral filters. *IEEE Trans Signal Process*, 67(1):97-109. https://doi.org/10.1109/TSP.2018.2879624

Levin A, 2006. Modified subdivision surfaces with continuous curvature. *ACM Trans Graph*, 25(3):1035-1040. https://doi.org/10.1145/1141911.1141990

Li RH, Li XZ, Fu CW, et al., 2019. PU-GAN: a point cloud upsampling adversarial network. Proc IEEE/CVF Int Conf on Computer Vision, p.7202-7211. https://doi.org/10.1109/ICCV.2019.00730

Lim B, Son S, Kim H, et al., 2017. Enhanced deep residual networks for single image super-resolution. Proc IEEE Conf on Computer Vision and Pattern Recognition Workshops, p.1132-1140. https://doi.org/10.1109/CVPRW.2017.151

Lim I, Dielen A, Campen M, et al., 2019. A simple approach to intrinsic correspondence learning on unstructured 3D meshes. Proc European Conf on Computer Vision, p.349-362. https://doi.org/10.1007/978-3-030-11015-4_26

Liu HTD, Kim VG, Chaudhuri S, et al., 2020. Neural subdivision. *ACM Trans Graph*, 39(4):124. https://doi.org/10.1145/3386569.3392418

Liu K, Ma N, Wang ZH, et al., 2023. Implicit neural distance optimization for mesh neural subdivision. Proc IEEE Int Conf on Multimedia and Expo, p.2039-2044. https://doi.org/10.1109/ICME55011.2023.00349

Loop C, 1987. Smooth Subdivision Surfaces Based on Triangles. MS Thesis, University of Utah, Salt Lake City, USA.

Maron H, Galun M, Aigerman N, et al., 2017. Convolutional neural networks on surfaces via seamless toric covers. *ACM Trans Graph*, 36(4):71. https://doi.org/10.1145/3072959.3073616

Milano F, Loquercio A, Rosinol A, et al., 2020. Primal-dual mesh convolutional neural networks. Proc 34th Int Conf on Neural Information Processing Systems, Article 81.

Monti F, Shchur O, Bojchevski A, et al., 2018. Dual-primal graph convolutional networks. https://doi.org/10.48550/arXiv.1806.00770

Nair V, Hinton GE, 2010. Rectified linear units improve restricted Boltzmann machines. Proc 27th Int Conf on Machine Learning, p.807-814.

Oliensis J, 1993. Local reproducible smoothing without shrinkage. *IEEE Trans Patt Anal Mach Intell*, 15(3):307-312. https://doi.org/10.1109/34.204914

Paszke A, Gross S, Massa F, et al., 2019. PyTorch: an imperative style, high-performance deep learning library. Proc 33rd Int Conf on Neural Information Processing Systems, Article 721.

Perez D, Shen YZ, Li J, 2023. Mesh convolutional networks with face and vertex feature operators. *IEEE Trans Vis Comput Graph*, 29(3):1678-1690. https://doi.org/10.1109/TVCG.2021.3129156

Poulenard A, Ovsjanikov M, 2018. Multi-directional geodesic neural networks via equivariant convolution. *ACM Trans Graph*, 37(6):236. https://doi.org/10.1145/3272127.3275102

Qi CR, Su H, Mo KC, et al., 2017. PointNet: deep learning on point sets for 3D classification and segmentation. Proc IEEE Conf on Computer Vision and Pattern Recognition, p.77-85. https://doi.org/10.1109/CVPR.2017.16

Qian GC, Abualshour A, Li GH, et al., 2021. PU-GCN: point cloud upsampling using graph convolutional networks. Proc IEEE/CVF Conf on Computer Vision and Pattern Recognition, p.11678-11687. https://doi.org/10.1109/CVPR46437.2021.01151

Schaefer S, Vouga E, Goldman R, 2008. Nonlinear subdivision through nonlinear averaging. *Comput Aided Geom Des*, 25(3):162-180. https://doi.org/10.1016/j.cagd.2007.07.003

Sharp N, Attaiki S, Crane K, et al., 2022. DiffusionNet: discretization agnostic learning on surfaces. *ACM Trans Graph*, 41(3):27. https://doi.org/10.1145/3507905

Smirnov D, Solomon J, 2021. HodgeNet: learning spectral geometry on triangle meshes. *ACM Trans Graph*, 40(4):166. https://doi.org/10.1145/3450626.3459797

Sorkine O, Cohen-Or D, Lipman Y, et al., 2004. Laplacian surface editing. Proc Eurographics/ACM SIGGRAPH Symp on Geometry Processing, p.175-184. https://doi.org/10.1145/1057432.1057456

Stam J, 1998. Exact evaluation of Catmull-Clark subdivision surfaces at arbitrary parameter values. Proc 25th Annual Conf on Computer Graphics and Interactive Techniques, p.395-404. https://doi.org/10.1145/280814.280945

Tai Y, Yang J, Liu XM, 2017. Image super-resolution via deep recursive residual network. Proc IEEE Conf on Computer Vision and Pattern Recognition, p.2790-2798. https://doi.org/10.1109/CVPR.2017.298

Tatarchenko M, Park J, Koltun V, et al., 2018. Tangent convolutions for dense prediction in 3D. Proc IEEE/CVF Conf on Computer Vision and Pattern Recognition, p.3887-3896. https://doi.org/10.1109/CVPR.2018.00409

Taubin G, 1995. A signal processing approach to fair surface design. Proc 22nd Annual Conf on Computer Graphics and Interactive Techniques, p.351-358. https://doi.org/10.1145/218380.218473

Vaxman A, Müller C, Weber O, 2018. Canonical Möbius subdivision. *ACM Trans Graph*, 37(6):227. https://doi.org/10.1145/3272127.3275007

Veličković P, Cucurull G, Casanova A, et al., 2018. Graph attention networks. https://doi.org/10.48550/arXiv.1710.10903

Wang NY, Zhang YD, Li ZW, et al., 2018. Pixel2Mesh: generating 3D mesh models from single RGB images. Proc 15th European Conf on Computer Vision, p.55-71. https://doi.org/10.1007/978-3-030-01252-6_4

Wang XT, Yu K, Wu SX, et al., 2018. ESRGAN: enhanced super-resolution generative adversarial networks. Proc European Conf on Computer Vision, p.63-79. https://doi.org/10.1007/978-3-030-11021-5_5

Wang Y, Sun YB, Liu ZW, et al., 2019. Dynamic graph CNN for learning on point clouds. *ACM Trans Graph*, 38(5):146. https://doi.org/10.1145/3326362

Wang YF, Perazzi F, McWilliams B, et al., 2018. A fully progressive approach to single-image super-resolution. IEEE/CVF Conf on Computer Vision and Pattern Recognition Workshops. https://doi.org/10.1109/CVPRW.2018.00131

Wang YF, Wu SH, Huang H, et al., 2019. Patch-based progressive 3D point set upsampling. Proc IEEE/CVF Conf on Computer Vision and Pattern Recognition, p.5951-5960. https://doi.org/10.1109/CVPR.2019.00611

Wang ZH, Chen J, Hoi SCH, 2021. Deep learning for image super-resolution: a survey. *IEEE Trans Patt Anal Mach Intell*, 43(10):3365-3387.

Yang WM, Zhang XC, Tian YP, et al., 2019. Deep learning for single image super-resolution: a brief review. *IEEE Trans Multim*, 21(12):3106-3121. https://doi.org/10.1109/TMM.2019.2919431

Yu LQ, Li XZ, Fu CW, et al., 2018. PU-Net: point cloud upsampling network. Proc IEEE/CVF Conf on Computer Vision and Pattern Recognition, p.2790-2799. https://doi.org/10.1109/CVPR.2018.00295

Zhou QN, Jacobson A, 2016. Thingi10K: a dataset of 10,000 3D-printing models. https://doi.org/10.48550/arXiv.1605.04797

Zorin D, 2007. Subdivision on arbitrary meshes: algorithms and theory. Proc Mathematics and Computation in Imaging Science and Information Processing, p.1-46. https://doi.org/10.1142/9789812709066_0001

Zorin D, Schröder P, Sweldens W, 1996. Interpolating subdivision for meshes with arbitrary topology. Proc 23rd Annual Conf on Computer Graphics and Interactive Techniques, p.189-192. https://doi.org/10.1145/237170.237254