UNIVERSITY OF OKLAHOMA

GRADUATE COLLEGE

LEARNING GEOMETRY THROUGH INTRINSIC REPRESENTATIONS AND DATA-DRIVEN PROCESSING

A DISSERTATION

SUBMITTED TO THE GRADUATE FACULTY

in partial fulfillment of the requirements for the

Degree of

DOCTOR OF PHILOSOPHY

By HAOLIANG ZHANG

Norman, Oklahoma

2026

Downloaded from SHAREOK. Persistent Handle URL: https://hdl.handle.net/11244/343177

LEARNING GEOMETRY THROUGH INTRINSIC REPRESENTATIONS AND DATA-DRIVEN PROCESSING

A DISSERTATION APPROVED FOR THE
SCHOOL OF ELECTRICAL AND COMPUTER ENGINEERING

BY THE COMMITTEE CONSISTING OF

Dr. Samuel Cheng, Chair

Dr. Christian El Amm

Dr. Jie Cao

Dr. Qi Li

Dr. Shangqing Zhao

© Copyright by HAOLIANG ZHANG 2026

All Rights Reserved.

Abstract

This thesis explores data-driven and intrinsic methods for efficient 3D shape representation and processing. While many learning-based approaches rely on large labeled datasets or shallow geometric cues, we instead leverage intrinsic structures—such as curvature, geodesic distance, and the logarithmic map—to enhance generalization, robustness, and geometric fidelity.

We first investigate learning-based methods for efficient mesh representation. To better capture local geometric structure, we propose an improved pooling operator for 3D morphable models that incorporates intrinsic geometric information through vertex normals. Building on this representation, we introduce Neural QSLIM, a geometry-aware framework that learns QSLIM-guided topological updates to improve mesh reconstruction quality in autoencoders. Beyond mesh representation, we study intrinsic geometry for non-rigid shape analysis by proposing a fully unsupervised method for discovering compact, stable, and ordered intrinsic landmarks via Karcher means, which can be directly incorporated into existing shape correspondence methods. Finally, we extend these ideas to mesh repair with a data-free learning framework that reconstructs missing regions directly on the input mesh while preserving the original topology and geometric fidelity, eliminating the need for external training datasets.

Together, these contributions demonstrate how intrinsic geometry and self-supervision can reduce reliance on large-scale annotations in neural geometric pipelines.

Acknowledgements

This dissertation reflects a long and sometimes difficult journey that shaped both my research and my life.

I would like to sincerely thank my advisor, Cheng Samuel. I have always been someone who prefers to follow my own ideas, sometimes stubbornly, and throughout my PhD he gave me the freedom to do that. Although we did not always agree on research directions, he ultimately gave me the space to pursue ideas that I genuinely cared about. Over time, he understood that I preferred to follow my own path, and I am grateful that he respected that. There were also periods when research was slow, or when I spent more time focusing on life outside academia, and he was always understanding and supportive. I truly appreciate the patience and trust he gave me during these years.

I am also grateful to Christian El Amm for supporting me financially during my PhD. Although we did not communicate frequently, he was always kind, understanding, and easy to work with. He never made things difficult for me and was always willing to help whenever needed. Looking back, I sometimes wish I had reached out more often, as I truly appreciated his support and trust. I especially valued the freedom he gave me to explore my own research direction, even during periods when progress was uncertain. That trust meant a lot to me.

I would like to thank my longtime roommate, Han Dong. We spent many years together in Tulsa and shared a lot of memorable experiences. I still remember the difficult summer when the air conditioning in our apartment was broken for nearly two months. I also remember the tornado that caused a large power outage across the city, leaving us to stay in the lab because it was the only place with electricity and air conditioning. Those moments were uncomfortable at the time, but looking back, they became part of what made graduate school memorable.

Most importantly, I want to thank my wife. Throughout this entire journey, she gave me constant emotional support and always believed in me. She often reminded me that I was capable and that I would eventually succeed. Having someone who consistently believed in me and encouraged me during difficult periods meant more than I can fully express.

I would also like to thank my current roommates, Shayan and Reza, for getting me into fitness. Before this, I had never trained consistently or followed a structured workout routine. I originally started with the simple goal of getting abs, and after eight months of training I

can clearly see meaningful changes in my physique. The progress has been real, even if the original goal is still a work in progress. More importantly, I developed habits that I believe will stay with me for a long time.

Finally, I am grateful for the experience itself. A PhD is not only about research or publications. It is also about persistence and accepting that meaningful results sometimes take a long time to appear. Much of research involves continuing to work without immediate feedback and trusting that the effort will eventually lead somewhere. This dissertation marks the end of an important chapter in my life.

Contents

1 Introduction 1

1.1 Data-Driven Geometry Processing 1

1.2 Intrinsic Features in Geometric Processing 2

2 Efficient Pooling Operator for 3D Morphable Models 4

2.1 Introduction 4

2.2 Related Work 6

2.2.1 Deep learning efforts on meshes 7

2.2.2 Geometry learning 7

2.2.3 Graph convolution networks 8

2.3 Method 9

2.3.1 Mesh sampling 10

2.3.2 Unpooling layer 11

2.4 Experimental evaluation 14

2.4.1 Datasets 14

2.4.2 Implementation details 14

2.4.3 Interpolation experiment 15

2.4.4 Pooling and unpooling layers 16

2.4.5 Comparison of network hyperparameters 17

2.5 Conclusion 19

2.6 Appendix 19

2.6.1 Curvature and Vertex Normal 19

3 Neural QSLIM for Mesh Autoencoders 21

3.1 Introduction 21

3.2 Related works 23

3.2.1 Geometric deep learning 23

3.2.2 Comparing with subdivision surfaces . . . 24
3.3 Method . . . 25
3.3.1 Build transformation matrix . . . 25
3.3.2 Data generation and training . . . 26
3.3.3 Up-sampling . . . 27
3.4 Applications . . . 30
3.5 Conclusion and Limitations . . . 32

4 Unsupervised Landmark Discovery via Karcher Means for Non-Rigid Shape Correspondence . . . 34
4.1 Introduction . . . 34
4.2 Related Work . . . 36
4.3 Preliminaries . . . 38
4.3.1 Notation . . . 38
4.3.2 Basic Pipeline . . . 38
4.3.3 Geometric Medians . . . 40
4.4 Algorithm . . . 42
4.4.1 Ordered Landmarks . . . 43
4.4.2 Correspondence Augmentation . . . 47
4.5 RESULTS . . . 49
4.5.1 Ablation Study . . . 53
4.6 Conclusion . . . 56

5 Data-Free Learning for Mesh Repair . . . 57
5.1 Introduction . . . 57
5.2 Related Work . . . 59
5.2.1 Traditional Geometry-Based Shape Repair . . . 59
5.2.2 Data-Driven Shape Inpainting . . . 60
5.2.3 Self-Prior Learning in Geometry Processing . . . 61
5.3 Method . . . 62
5.3.1 Geometric Preprocessing . . . 62
5.3.2 Self-Supervision . . . 64
5.4 Results and Applications . . . 67

5.4.1 Accuracy and Performance 67
5.4.2 Comparison with Supervised and Self-Prior Models 68
5.4.3 Applications 70
5.5 Limitations & Future Work 72

6 Conclusion 74
6.1 Future Directions 75

Chapter 1

Introduction

Geometric data has become a core building block in modern shape understanding, enabling applications ranging from shape correspondence to deep surface learning. Meshes, especially triangle meshes, serve as the standard representation for 3D geometry, playing a similar foundational role to images in computer vision. While image data benefits from regular grid structure, geometric data on meshes often lacks such structure, posing challenges for algorithm design. This thesis investigates how intrinsic and extrinsic representations of mesh geometry affect downstream geometry processing tasks. Specifically, we explore how geometry can be encoded not just by the vertex positions in $\mathbb{R}^3$, but also by intrinsic quantities such as edge lengths and geodesic distances. These intrinsic representations can reveal structure that is invariant under isometric deformation and, in some cases, offer better stability and generalization.

We present algorithms that leverage both views of geometry to address problems such as shape correspondence, mesh encoding, and neural representation learning. By integrating discrete differential operators and spectral methods with modern learning architectures, we aim to provide practical tools that extend across both analytical and data-driven domains of geometry processing.

1.1 Data-Driven Geometry Processing

Recent advances in synthetic shape generation and 3D scanning technologies have greatly increased the accessibility of geometric data. This motivated the development of *data-driven* approaches for geometry processing. Rather than focusing on individual shapes in isolation,

data-driven methods exploit statistical regularities across a collection of shapes to inform representation, synthesis, or matching. In some cases, this may involve transferring structural knowledge from a few annotated exemplars to novel geometries. In others, it may involve learning high-level reasoning directly from data without explicit supervision.

In this thesis, we explore different strategies to efficiently learn latent representations and extract compact features from 3D shapes. Many existing approaches rely on shallow geometric features—such as raw vertex positions or simple edge-based operators—we build on concepts from discrete differential geometry to design methods that more faithfully capture intrinsic structure (such as mean curvature, Laplace-Beltrami and cotangent weights). Built on top of this core idea, in Chapter 2, we propose an improved pooling operator for 3D morphable models that leverages intrinsic cues such as vertex normals and face areas, moving beyond purely Euclidean contraction. Our method improves generalization and efficiency by enhancing pooling structure without requiring per-instance optimization. In Chapter 3, we extend this direction by introducing *Neural QSLIM*, a geometry-aware framework for mesh up-sampling in autoencoders. Unlike conventional approaches that rely on fixed quadric simplification and point-wise projection, our method learns to reconstruct fine-grained geometry from coarse triangle meshes via QSLIM-guided topological updates. This design improves structural fidelity and enables more flexible non-linear decoding across different mesh autoencoder settings.

These contributions lay the groundwork for leveraging both intrinsic geometric cues and data-driven learning to design efficient and generalizable pipelines for 3D shape understanding. In the next section, we shift focus to intrinsic features and their role in geometric processing.

1.2 Intrinsic Features in Geometric Processing

In many settings, training data is scarce or high quality labels are prohibitively expensive to obtain. This motivates algorithms that extract rich information directly from the input geometry, often without supervision. Intrinsic features—such as geodesic distances, exponential maps, and spectral descriptors—encode invariant structure and can drive self-supervised or optimization-based methods. In this thesis, we explore how intrinsic cues enable robust shape understanding in the absence of external labels. In Chapter 4, we leverage intrinsic geomet-

ric cues to automatically detect stable and ordered landmarks on near-isometric surfaces. By computing Karcher means and selecting maximally distant points, our method produces an ordered landmark set that is consistent across near-isometric surfaces, enabling accurate and robust correspondence. These landmarks enable accurate and stable correspondence across near-isometric shapes, without requiring supervision. Our approach eliminates the need for manually defined landmarks and can be integrated into existing shape matching pipelines as a plug-and-play module. Chapter 5 extends this direction to mesh repair, where complete training shapes are often unavailable. We develop a data-free learning framework that combines geometric processing with self-supervision to reconstruct missing regions directly on incomplete meshes. The method preserves the original geometry and topology outside the repaired regions while avoiding the need for external training data.

Overall, this dissertation studies how geometric structure can be incorporated into learning-based geometry processing. The work progresses from mesh representation and reconstruction to intrinsic shape analysis and mesh repair. Although these problems address different tasks, they share a common goal of using the underlying geometric structure to improve learning while preserving important properties of the input geometry.

The text and figures in this thesis have significant overlap with several papers published by myself over the course of my PhD. These publications are the original source for such content, used with permission. In particular, see the following publications:

- **Haoliang Zhang**, Samuel Cheng, Christian El Amm, Jonghoon Kim. "Efficient pooling operator for 3D morphable models." *IEEE Transactions on Visualization and Computer Graphics 30 (7)*, (2023).
- **Haoliang Zhang**, Xintong Li, Jonghoon Kim, Samuel Cheng, Christian El Amm. "Neural QSLIM for Mesh Autoencoders." *International Conference on Artificial Neural Networks*, (2025).
- **Haoliang Zhang**, Samuel Cheng, Christian El Amm. "Unsupervised Landmark Discovery via Karcher Means for Non-Rigid Shape Correspondence." *The Visual Computer*, 42.1 (2026).
- **Haoliang Zhang**, Jiali Zhang, Samuel Cheng, Christian El Amm. "Data-Free Neural Mesh Inpainting: Preserving Topology and Geometry." *The Visual Computer*, 42(11) (2026).

Chapter 2

Efficient Pooling Operator for 3D Morphable Models

Learning the latent representation of three-dimensional (3D) morphable geometry is useful for several tasks, such as 3D face tracking, human motion analysis, and character generation and animation. For unstructured surface meshes, previous state-of-the-art methods focus on designing convolution operators and share the same pooling and unpooling operations to encode neighborhood information. Previous models use a mesh pooling operation based on edge contraction, which is based on the Euclidean distance of vertices rather than the actual topology. In this study, we investigated whether such a pooling operation can be improved, introducing an improved pooling layer that combines the vertex normals and adjacent faces area. Furthermore, to prevent template overfitting, we increased the receptive field and improved low-resolution projection in the unpooling stage. This increase did not affect processing efficiency because the operation was implemented once on the mesh. We performed experiments to evaluate the proposed method, whose results indicated that the proposed operations outperformed Neural3DMM with 14% lower reconstruction errors and outperformed CoMA by 15% by modifying the pooling and unpooling matrices.

2.1 Introduction

Several important real-world datasets are presented in the form of graphs: social networks, knowledge graphs, protein-interaction networks, and others [1]; however, progress in representation learning of geometric data has been limited. Learning the latent representation of mesh data is useful for several tasks, such as 3D face tracking, human motion analysis, and character generation and animation. Previous methods used principal component analysis

(PCA) [2, 3, 4, 5] or manually defined blendshapes [6, 7, 8] to construct the linear latent representation for a 3D human face. Similarly, Blanz and Vetter [9] proposed a method learned from a set of face images with associated 3D face scans and represented by selected basic shapes and textures. Owing to the linear base, the aforementioned representation power can be limited.

Convolution neural networks (CNNs) are successfully used to capture statistical features in large-scale datasets. However, conventional convolution and pooling operations require regular input data to perform weight share and kernel optimizations [10]. Owing to irregular sampling and connections in the mesh, CNNs cannot be directly applied to meshes as in the case of regular two-dimensional (2D) grid data. Taigman et al. [11] proposed a locally connected convolution (LCConv) that defines the convolution weights for each vertex; however, this method requires considerable memory and is prone to overfitting. Furthermore, in the case of irregular data format, the transformed point features are fed into a convolutional operator through a multilayer perceptron (MLP), which generates a potentially canonical order [12]. To reduce the parameter count, Zhou et al. [13] proposed a fully convolutional mesh autoencoder. Weights on vertices are sampled based on the weights obtained using different functions; however, this process increases parameter count by 7–20 times compared with other processes.

Common compromise methods [14, 15, 16, 17] that bijectively map a mesh in $\mathbb{R}^3$ to $\mathbb{R}^2$ or some regular surface (uv space, spheres, and torus) enable the model to apply CNNs to train on mesh geometry. However, they suffer from distortion and are adversely affected by artifacts along seam lines in UV mapping. Thus, Ji et al. [18] convolve a 3D kernel to the cube formed by stacking multiple frames together. However, volumetric convolutions [19] are memory-intensive and restricted to low-resolution volumes.

As a more elegant approach, Ranjan et al. [20] proposed CoMA that follows the work [21] using fast Chebyshev filters as convolution layers and applies quadric mesh sampling to preserve the topological structure at different hierarchical multiscale levels. Feature aggregation adopts edge contraction for downsampling and barycentric coordinates for upsampling. Bouritsas et al. [22] proposed Neural3DMM that replaces the spectral convolution layers with spiral operators [23]. The spiral path around each vertex is determined by the geometry of the template mesh. It outperforms CoMA and other preceeding methods. One of the reasons for

their success is the multiscale hierarchical structure that allows the model to learn translation-invariant localized features of a mesh. Both models results are promising on surface meshes, but share the same pooling/unpooling operations in the multiscale hierarchical structure. The pooling operation selects a valid edge for contraction based on Euclidean distance rather than actual mesh topology; moreover, it neglects area information by assuming that every triangle face contributes equally. As lossless unpooling is not feasible, it maps discarded vertices into the closest triangle of the coarse mesh, thus leading to template overfitting.

To address this issue, we introduce a mesh autoencoder with area and vertex normal aware pooling operations. In contrast to the previous pooling operations [20, 22] that use uniform weights over the entire surface, our method combines the normal attribute and the area of vertex-adjacent faces with each vertex. In the unpooling stage, we train the unpooling matrix in different meshes that share the same pooling operation. The proposed unpooling operation is only applicable to low-resolution projection to maintain computational efficiency. Thus, we efficiently aggregate the global and local features in a multiscale hierarchical structure.

In subsequent experiments, we demonstrate that the proposed mesh autoencoder outperforms the state-of-the-art (SOTA) methods in several datasets. The proposed model can compress and reconstruct meshes with minimal error, does not require additional supervision or preprocessed shape descriptors, and adequately maintains the mesh shape and details. Moreover, it does not affect processing efficiency because the operation is performed once. To the best of our knowledge, the proposed method provides a better multiscale hierarchical structure to capture global features and outperforms competitive baseline models by a significant margin in all tasks evaluated.

2.2 Related Work

A series of efforts have been made to train deep neural generative models to learn latent geometric representations from registered meshes. The proposed model draws inspiration from mesh generative models and recent studies on graph convolution networks. We provide a brief overview of the related work in both fields.

2.2.1 Deep learning efforts on meshes

CNNs cannot be directly applied to meshes due to irregular sampling and connections. Previous studies have focused on learning the latent representation of the 3D human face using linear transformation [2, 3, 4, 5, 6, 7, 8]. Tretschk et al. [24] added an embedded deformation layer to the mesh autoencoder to generalize and apply deep learning methods to a non-Euclidean domain (graph and manifolds). Later methods [14, 15, 16, 17] map a mesh to $\mathbb{R}^2$ or regular surfaces (uv space, spheres, and torus) using bijective mapping, and apply CNNs to train on mesh geometry. However, there is no bijective map to tell whether two topological spaces are homeomorphic, which may destabilize a generative model while modeling surface structures. Moreover, parameterized methods also suffer from distortion and discontinuity along seam lines.

Current studies [18, 25] directly apply the convolution operator by simply convolving a 3D kernel to the cube formed by stacking multiple frames together or several convolution layers at the autoencoder. The SOTA method [20] uses fast Chebyshev filters as convolution layers and quadric mesh sampling to preserve the topological structure at different hierarchical multiscale levels; however, it selects the valid edge (v_1, v_2) for contraction based on Euclidean distance instead of actual mesh topology, leading to nonuniformity of the receptive field in regard to network neurons, and latent feature discontinuity. For the Body [26] and Hand [27], to project a new shape to the model space, non-linear skinned vertex-based models (SMPL [28] and MANO [29]) require joint localization and solving special optimization problems. Zhou et al. [13] apply trainable density parameters to each vertex for feature aggregation in pooling and unpooling layers, however; this process increases the number of parameters by 7–20 compared with other processes.

As described in Section 2.3, the proposed method does not require additional supervision and adequately maintains mesh shape and details; it exhibits better representation power and enables the model to efficiently aggregate neighbor information.

2.2.2 Geometry learning

QSLIM algorithm [30] uses iterative contractions of edges to simplify models and maintains surface shape using quadric matrices. An improved approach [31, 32] is to combine the quadric

error metric with appearance attributes, such as texture and color. Yao et al. [33] add the cost of curvature to the quadric error metrics, but their approach is computationally inefficient. We show that by combing vertex normal attributes, our method encodes curvature implicitly (Appendix 2.6.1).

While the QSLIM algorithm is crucial for mesh simplification, it is only half of the two main components that comprise the mesh autoencoder model. The other half consists of the unpooling layers. The closest solution to mesh unpooling is the subdivision technique. Subdivision surfaces [34, 35, 36] are defined by recursive upsampling of a discrete surface mesh. Each input mesh is divided by splitting edges and adding vertices. The positions of vertices are smoothed by taking a weighted average of their neighbors' positions. Rather than maintain the mesh topology, subdivision typically results in smoothed surfaces. This topic includes a wide variety of subdivision schemes that would be outside the scope of this paper to thoroughly cover.

Ranjan et al. [20] deform a template with a transformation matrix that encodes the mesh's topological information. Discarded vertices are projected into the downsampled mesh surface using barycentric coordinates. The closest-point approach decreases accuracy of mesh reconstruction and damages structure (Figure 2.4). Using a concept similar to CoMA, Bouritsas et al. [22] (Neural 3DMM) formulates an ordering-based graph convolutional operator, contrary to the permutation invariant operator Chebyshev GNN used in the CoMA model. However, while Neural3DMM improves the graph convolutional operator, it uses the same pooling/unpooling layers of CoMA. Hanocka et al. [37] propose MeshCNN which combines convolution and pooling layers that operate on the mesh edges. MeshCNN directly learns the local structure via undirected edges and shows applications in deterministic tasks. In contrast, we focus on generative tasks.

2.2.3 Graph convolution networks

Spectral and spatial approaches are two main strategies to design filters in a graph. A spectral filter performs a weighted averaging of neighboring vertices using the *Laplacian* operator [38], formulating convolution-like operations to match with local patches on graphs or manifolds. However, evaluating spectral convolutions is computationally expensive. Hammond et al. [39] suggest an approximation using a filter based on blue a truncated expansion of Chebyshev

polynomials up to K^{th} order. Inspired by Weisfeiler-Lehman (WL) isomorphism test [40], Hamilton et al. [41] extend the graph convolutional network (GCN) to the inductive setting. It iteratively concatenates the current representation of a node with aggregated neighborhood features to obtain additional information from further reaches of the graph. Monti et al. [42] proposed a spatial-domain method with regard to graphs and manifolds that could generalize several previous models [43, 21, 44, 20, 45, 46]. These methods consider pair vertex features to compute attention weights with different types of kernels or learnable functions. Some methods [47, 25] use these graph convolution layers in their models. [23] proposed a spiral convolution for a mesh autoencoder; the spiral sequences for each vertex are determined from neighboring vertices following a fixed order.

Graph convolution operators used in the networks described above do not support transpose convolution. Therefore, tasks that require multiscale architectures with pooling and unpooling layers need to apply additional layers (e.g., Verma et al. [48] use Graclus algorithm [49] which is efficient at clustering various graphs). The coarse weight is the sum of two matched nodes, and the coarse graph has approximately half the nodes. Consequently, quadric error minimization [30] is implemented in such models [20, 50, 22].

2.3 Method

In this section, we introduce the proposed pooling and unpooling operations. The pooling layers in current autoencoder models [20, 50, 22] suffer from non-uniform vertices. In contrast, our autoencoder model leverages the benefits of attributes-weighted QSLIM [31, 32] to include more geometric information. The previous unpooling matrix is developed using the closest triangle barycentric coordinates of the coarse template, which cause template overfitting. To avoid overfitting, we downsample the randomly selected meshes and train a fully connected layer based on corresponding meshes. Finally, we construct a hierarchical multiscale mesh autoencoder using the proposed methods.

We define a mesh by a set of vertices and faces, $M = (V, \mathcal{E}, F)$, with vertex $v \in V$ having a geometric position in the Euclidean space, $V \in \mathbb{R}^3$ and edges $(v_i, v_j) \in \mathcal{E}$. Each triangle face $f \in F$ is denoted by three vertices (v_1, v_2, v_3).

2.3.1 Mesh sampling

Current SOTA models downsample meshes based on the QSLIM scheme, each vertex v is assigned with the symmetric 4×4 quadric matrix $Q_g^v(\mathrm{v})$:

$$Q^v(\mathrm{v}) = \sum_{f \in \mathrm{v}} \mathrm{Q}_\mathrm{g}^f(\mathrm{v}) \tag{2.1}$$

where $\mathrm{v}^\mathrm{T}\mathrm{Q}_\mathrm{g}^\mathrm{f}\mathrm{v}$ is equal to the geometry distance of a point $\mathrm{v} = [v_\mathrm{x}, v_\mathrm{y}, v_\mathrm{z}, 1]^\mathrm{T}$ to the plane containing the face f. It iteratively contracts vertex pairs $(v_1, v_2) \rightarrow v$ that minimizing $Q^{v_1}(\mathrm{v})+ \mathrm{Q}^{v_2}(\mathrm{v})$. First, we modify the QEM function, as expressed in Eq. (2.2); each vertex v of the mesh is assigned with the sum of quadrics weighted by its adjacent faces area respectively.

$$Q^v(\mathrm{v}) = \sum_{f \in \mathrm{v}} \mathrm{area}(f)\mathrm{Q}_\mathrm{g}^f(\mathrm{v}) \tag{2.2}$$

A vertex normal is a directional vector associated with a vertex [51] (see Figure 2.1). It also

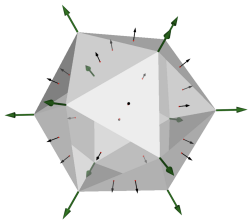


Figure 2.1: Vertex and face normals

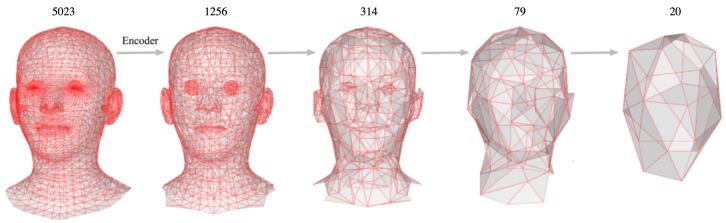
consists of the curvature information of the shape (Appendix 2.6.1). The contribution of faces normal to a vertex normal is the ratio of the corner angle in which the vertex is related to the sum of all corner angles surrounding the vertex. Thus, the error defines the distance from $v = \begin{pmatrix} p \\ vn \end{pmatrix} \in \mathbb{R}^6$ to the affine subspace spanned by face (v_1, v_2, v_3). where $||p - p'||^2$ is the geometric distance error and $||vn - vn'||^2$ is the normal vector error. This metric may underestimate the error when two points have a closer normal value but are geometrically distanced from each other. The weight λ_{vn} is used to scale the normal deviation. The total error metric is defined as the sum

$$Q^v(\mathrm{v} = \begin{pmatrix} p \\ vn \end{pmatrix}) = \sum_{f \in \mathrm{v}} \mathrm{area}(f)\mathrm{Q}_g^f(\mathrm{v}) + \lambda_{\mathrm{vn}}\mathrm{Q}_{\mathrm{vn}}^f(\mathrm{v}) \tag{2.3}$$

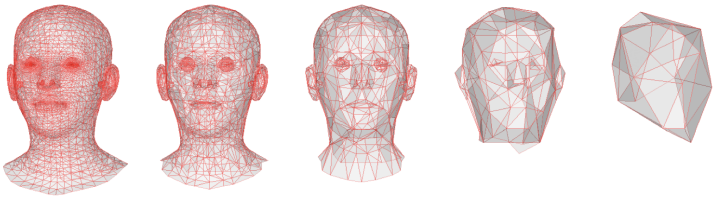
where $Q_g^f(\mathrm{v})$ is the same as obtained using the QEM methods but weighted by the adjacent faces area, and $Q_{vn}^f(v) = (g_{vn}^T p + d_{vn} - vn)^2$ is the squared normal deviation. (g_{vn}, d_{vn}) are computed using Eq. (2.4)

$$\begin{pmatrix} p_1^T & 1 \\ p_2^T & 1 \\ p_3^T & 1 \\ n^T & 0 \end{pmatrix} \begin{pmatrix} g_j \\ \\ d_j \end{pmatrix} = \begin{pmatrix} vn_1 \\ vn_2 \\ vn_3 \\ 0 \end{pmatrix} \tag{2.4}$$

where (g_j, d_j) are the gradients of the scalar function of the triangle face and scalar respectively. The pooling operator methods used in the current SOTA models do not effectively downsample the mesh, due to their inherent limitations. Figure 2.2 shows a comparative analysis, in which the sample rate is four in each layer. As shown in Figure 2.2b, after downsampling three times using QEM, the neck and jaw parts of the mesh disappear. We set $\lambda_{vn} = 0.005$ in our experiment, as shown in Figure 2.2a; our graph sampling scheme maintains the head shape and detail in the multiscale hierarchy structure.



(a) Our sampling process



(b) Quadric mesh sampling

Figure 2.2: Network architecture and comparison for mesh sampling

2.3.2 Unpooling layer

The unpooling layer provides an approximate location of discarded vertices in a fine mesh, it initializes the value for training to be conducted later. Hence, a bad initial guess will undermine the reconstruction accuracy. Crude methods [20, 50, 22] generate an unpooling matrix

during the downsampling process. Vertices in the fine mesh mapped to the closest triangle in the coarsened mesh are denoted by barycentric coordinates. The vertices in fine mesh $V_f = M_u V_c$ are obtained by multiplying using unpooling matrix M_u with coarse mesh V_c. From this perspective, the transform matrix is the zero-bias FC layer to connect fine and coarse meshes.

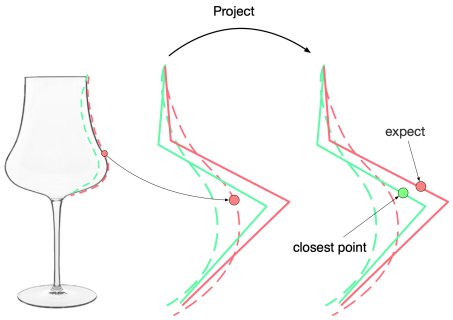


Figure 2.3: Closest point projection

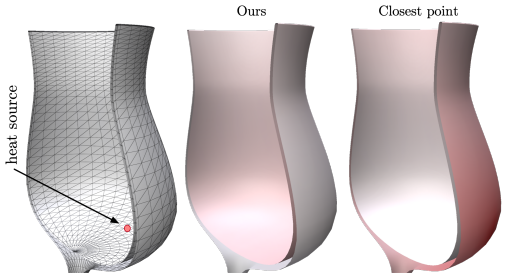


Figure 2.4: The heat source is placed inside the wine glass. The closest projection caused the error, where the heat does not diffuse along the surface

The model overfits the template because this operation is only applicable to the template mesh. More importantly, for curved surfaces embedded in 3D, the use of the closest point projection will lead to failure cases. As shown in Figure 2.3, the dotted and solid lines represent the fine and coarse meshes, respectively. For the red point in the fine mesh, its closest point in the coarsened mesh is the green point in the right figure. However, this is an incorrect estimation because the model will use the green vertex features in the unpooling layer to determine the red point in the fine mesh. In Figure 2.4 we create a stress test using a wine glass model. The heat source is placed inside the wine glass. The heat does not diffuse along the surface because of incorrect vertex projection.

To solve the aforementioned problem, we randomly selected 0.5% of the overall meshes of the training set and pooling with the shared matrix. The unpooling matrix can be considered as the fully connected layer without bias. In the proposed method, the unpooling matrix is

trained to map the input coarse mesh to the corresponding fine mesh. The input feature to the module consists of only vertices coordinates.

This implies that our unpooling layer has more generalization ability. It also increased the receptive field and improved low-resolution projection in the unpooling stage.

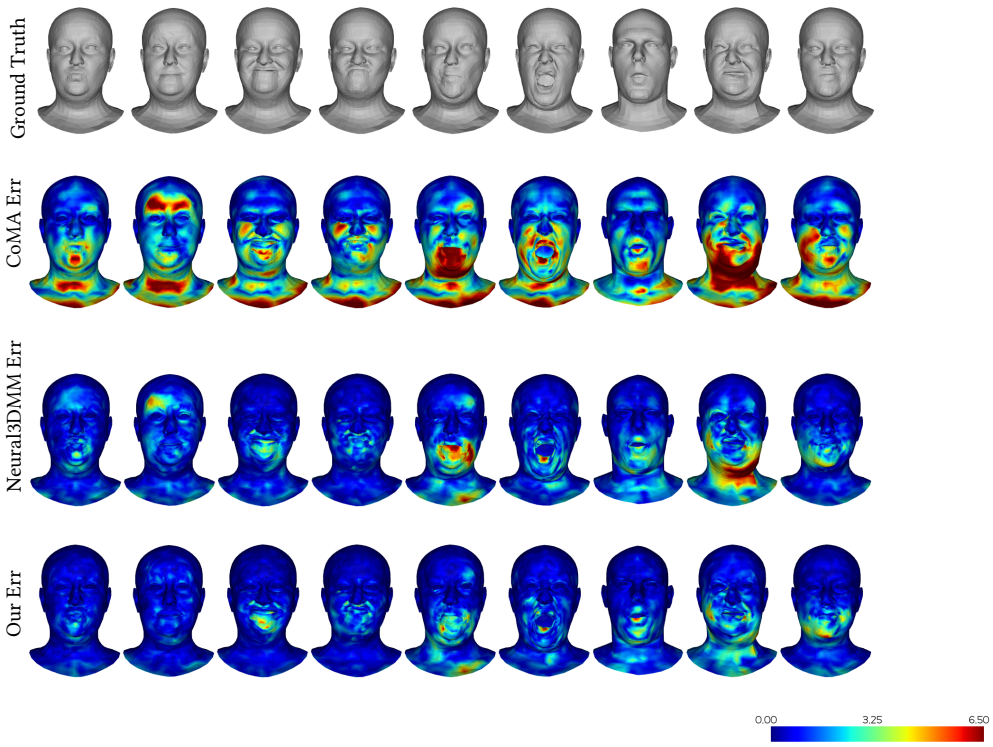


Figure 2.5: Qualitative comparison results for interpolation experiment. Colour coding of the per vertex euclidean error of the reconstructions. The error values are saturated at 6.5 (millimeters). Hot colors represent large errors.

Parameters Analysis: As the non-trainable parameters, the pooling and unpooling matrices are loaded at the beginning of the process, frozen, and used for aggregation. The original unpooling matrix used three weights to denote the vertex in the fine mesh, which implies that the unpooling matrix M_u is sparse and include $3N$ entries, where N is the number of vertices of the previous fine mesh. The closest point fault tends to occur in low-resolution meshes rather than in high-resolution meshes. Thus, we considered the outermost two unpooling layers as the original matrix and changed the other two layers. In other words, we increased the receptive field of the vertex in the low-resolution mesh. In our layer, we set the vertex weight to zero in cases less than 0.02. We implemented sparse matrix multiplication via scattering and added vertex features corresponding to cluster nodes across a batch of input vertex feature matrices. We run an experiment to evaluate the template overfitting, for more details,

please refer to Section 2.4.4.

2.4 Experimental evaluation

In this section, we present the result of a series of ablation studies to compare our operator with SOTA 3D mesh AEs, by evaluating our model on 3D shape reconstruction. Furthermore, we quantitatively show that our method can yield better unpooling results than the closest point projection, as described in Section 2.4.4. Finally, we compare the performances of different network hyperparameters under the same network architecture. All the experiments were trained with L1 reconstruction loss only and reported with point mean Euclidean distance error if not specified.

2.4.1 Datasets

COMA [20]: The facial expression dataset contains 12 different subjects with 12 extreme facial expressions. The dataset includes 20K + 3D meshes, each registration includes 5023 vertices.
DFAUST [26]: The dynamic human body shape dataset comprises 40K + 3D meshes of ten unique identities performing actions with 6,890 vertices each. We followed the data split setting in [22].

2.4.2 Implementation details

Spiral sequences are consistent, robust, and based on an ordered fixed-length serialization. We followed the specified settings in [22] to combine the spiral convolution with our pooling operator. The spiral length is set to nine to be aligned with Neural3DMM [22]. Training is conducted using Adm [52] for 300 epochs with a learning rate of 0.001 and a learning rate decay of 0.99 per epoch. Our autoencoder includes an encoder and a decoder, we follow the settings in [22, 20], the number of mesh vertices reduce/increase by a factor of four during pooling and unpooling. We set $\lambda_{vn} = 0.005$ in our experiment to scale the normal deviation. We use raw 3D coordinates as input node features instead of SHOT descriptors which were used in SpiralNet [23]. We use $[\cdot]$ denote network architectures, for encoder filter size:[16,16,16,32] and decoder size:[32,16,16,16], the structure is represented as $3 \times \{Conv(16) \rightarrow Pool(4)\} \rightarrow \{Conv(32) \rightarrow Pool(4)\}$. We only illustrate the encoder in the later sections as the decoder is the mirror of the encoder, in which the pooling layers are

replaced with the unpooling layers. Fully connected layers with the designed latent vector connect the encoder and decoder.

2.4.3 Interpolation experiment

We evaluate the reconstruction capability of the autoencoder. For the COMA dataset, following the interpolation experimental setup [20, 22], we split the dataset into training and test sets at a ratio of 9:1. For the DFAUST, we follow the data split setting in [22] that randomly split the data into a test set of 5000, 500 validation, and 34,5K+ train. For COMA, the convolutional filters of the encoder exhibited a size of [64,64,64,128] and for DFAUST the sizes were [16,32,64,128] and [128,64,32,32,16].

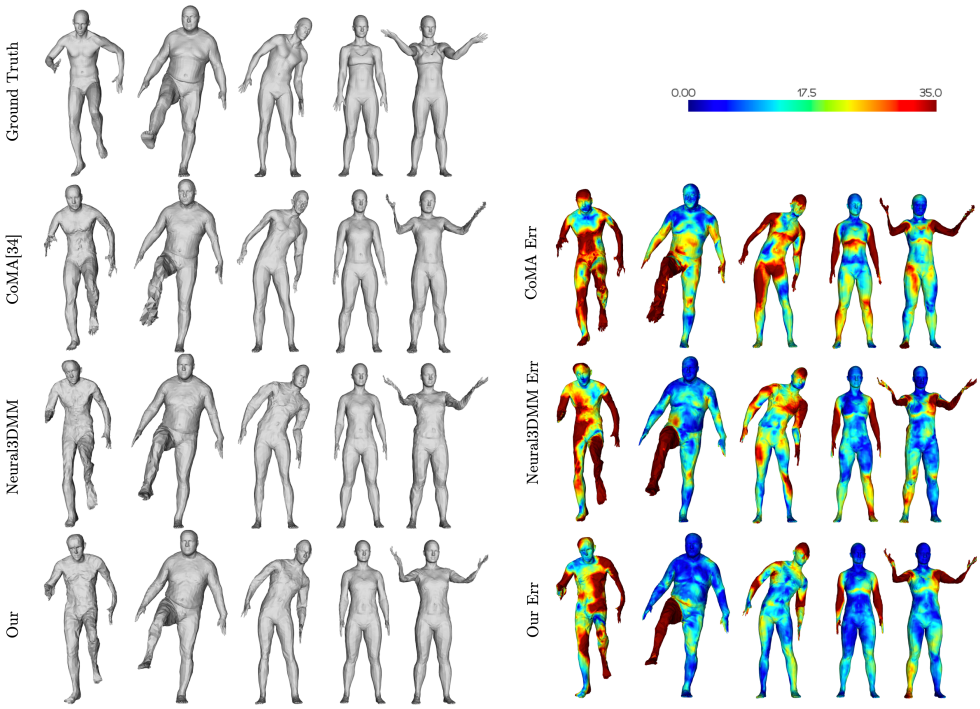


Figure 2.6: Colour coding of the per vertex euclidean error of the reconstructions. Top row is ground truth. The error values are saturated at 35 (millimeters)

We compared our model with two high baseline models: CoMA [20] and Neural3DMM [22]. For both datasets, as clearly illustrated in Figure 2.7, our model consistently outperformed the CoMA and Neural3DMM models in every latent dimension. The nine COMA exemplar errors and five DFAUST results of the reconstructions samples are well defined in Figure 2.5 and Figure 2.6 respectively. CoMA and Neural3DMM reconstruct the vertices in approximately correct locations but struggle to recover the expression detail during the extreme expressions. Our model balanced details and main head shape resulting in high-quality

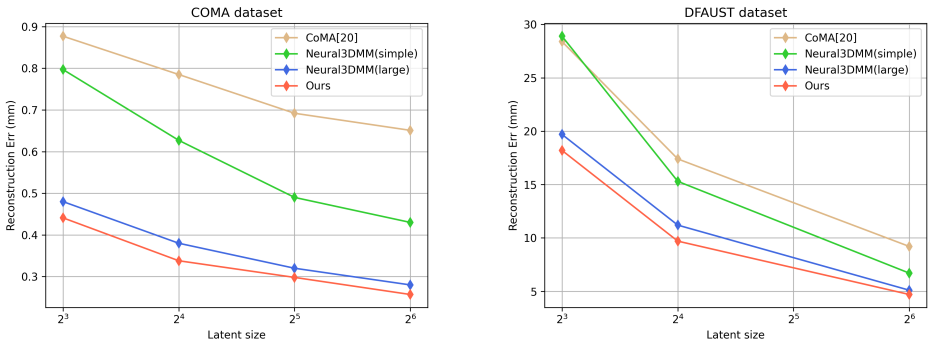


Figure 2.7: Reconstruction errors of different methods

reconstructions. We use an eight-dimensional latent space to encode the meshes in all models.

2.4.4 Pooling and unpooling layers

The performance of the model is closely related to the pooling and unpooling procedures, which have been widely used in previous models. The unpooling matrix is calculated during pooling because loss-less unpooling is not feasible [20]. Average aggregation uses the same pooling as [20, 22] and assigned aggregation weights identically for vertices in the receptive field. MeshCNN [37] performs convolution on the edges; however, it is computationally inefficient and does not perform well after 300 epochs. For the current unpooling operation, the contracted vertices in the previous layer are mapped into the closest triangle of the downsampled mesh surface and stored by the barycentric coordinates. Unlike a regular grid or 2D meshes we use bilinear interpolation or barycentric coordinates to construct vertices. For curved surfaces embedded in 3D, the use of closest point projection will lead to failure cases. Furthermore, it overfits one template and does not represent the actual topology of the mesh. To test the geometry learning improvement of our model, the convolution operation was frozen in the experiment. The model architectures and the kernel size were fixed. We trained with 300 epochs. The encoder filter was set to [16,16,16,32], and the decoder mirrored the encoder. The latent vector and convolutional filters are the same as [20]. As listed in Table 2.1, our operation achieves $>$ 15% lower error than the previous pooling and unpooling operations. Moreover, we demonstrate the reconstruction result of the overfitting effect and illustrate the unpooling result in Figure 2.8. We have randomly selected an extreme expressions dataset and downsampled it four times as input. The coarse mesh was unpooled four times and compared with the reference to analyze the unpooling accuracy. As shown in Fig-

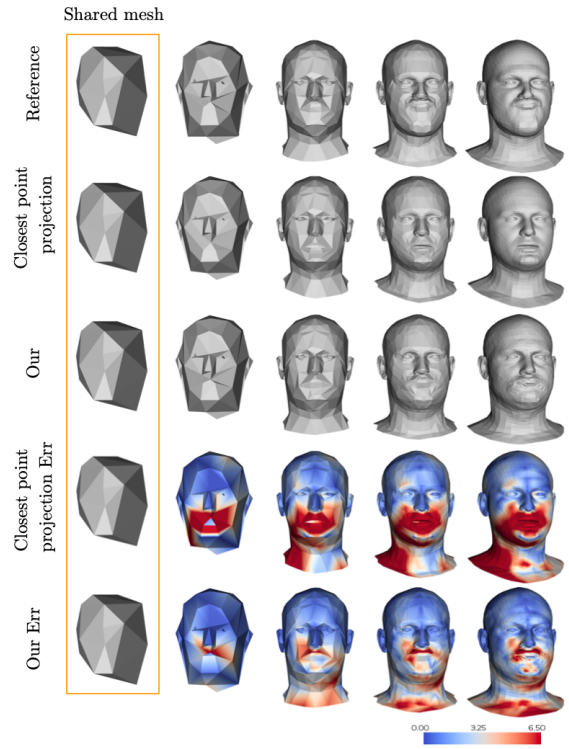


Figure 2.8: Visualization of unpooling meshes with error: The first row lists the pooling meshes in reverse order. The two unpooling methods share the same initial coarse mesh. The second and third rows present the results of the closest point projection [20, 22] and our proposed unpooling method, respectively. The error values are saturated at 6.5 (millimeters)

ure 2.8, the closest point projection methods exhibited significant template overfit. However, the proposed method provided an effective projection during the unpooling of the layers.

Table 2.1: Comparison with original pooling and unpooling: Interpolation experiments. The best results are highlighted in bold. The mean error with standard deviation and median errors are shown in the table.

	Mean Error	Median Error
Average	0.982	-
Variant weight [13]	0.932	-
CoMA [20]	0.884 ± 1.003	0.543
CoMA (Our)	**0.749 ± 0.888**	0.443

2.4.5 Comparison of network hyperparameters

In this section, we compare different network hyperparameters with our operators. Based on the architecture defined in Section 2.4.2, we considered the block number, the input/output channel, and vertex number (the same number) except for PCA. In CoMA [20], the convolutions use Chebyshev filtering with K = 6, the convolutional filters of the encoder had sizes

Table 2.2: Comparison with CoMA and Neural3DMM: Interpolation Experiment. Neural3DMM^1 for simple Neural3DMM with filter size [8,16,16,32,32], Neural3DMM^2 for larger Neural3DMM with filter size [64,64,64,128]. The best results per architecture appear in bold. Errors are in millimeters

	Mean Error	#Parameters
Ours ([64,64,64,128])	**0.441 ± 0.558**	489,995
Ours ([32,32,32,64])	**0.547 ± 0.641**	134,411
Ours ([16,16,32,32])	**0.692 ± 0.804**	53,419
Ours ([16,16,16,32])	**0.745 ± 0.874**	39,563
Neural3DMM^1 [22]	0.804 ± 0.877	48,067
Neural3DMM^2 [22]	0.483 ± 0.527	530,963
CoMA [20]	0.845 ± 0.994	33,856
PCA	1.639 ± 1.638	120,552

[16,16,16,32], and the decoder is a mirror of the encoder. Neural3DMM [22] modified the COMA architecture for simple Neural3DMM by adding an extra layer in the encoder and the decoder respectively (encoder filter sizes: [8,16,16,32,32], decoder: mirror of the encoder). The larger Neural3DMM uses [64,64,64,128] for COMA ,[16,32,64,128] and [128,64,32,32,16] for DFAUST.

To evaluate the interpolation capability of our autoencoder, we compared it with the above current SOTA framework for the registered COMA dataset. We used the Euclidean distance for comparison with the CoMA and Neural3DMM methods. The mean errors with standard deviation are listed in Table 2.2 for comparison. For a fair comparison, we compared the eight-dimensional latent vector with 300 epochs with regard to training. We evaluate our model with four selected designs, which have the same or fewer parameters (encoder filter sizes: [16,16,16,32]/[16,16,32,32]/[32,32,32,64]/[64,64,64,128], decoder: mirror of the encoder). Based on the architecture defined in Section 2.4.2, Table 2.2 lists the architectural designs, the errors, and the parameter count.

As listed in Table 2.2, our hierarchical multiscale representation of the mesh consistently reduced the error under different settings. The proposed model([16,16,16,32]) outperforms CoMA models with 18% lower reconstruction error and outperformed Neural3DMM (simple) models with 14% lower reconstruction error, while using 17.6% fewer parameters. Moreover, it outperformed the large Neural3DMM model ([64,64,64,128]) with 8.7% lower reconstruction error.

2.5 Conclusion

We introduce a mesh autoencoder that produced SOTA results for fixed-topology 3D morphable meshes. We designed a novel pooling/unpooling operator and combined it with spiral convolution. The proposed method avoids template overfitting and provides efficient localized interpolation and generalization ability using the proposed pooling/unpooling operator. Moreover, we applied the proposed method to the 3D shape reconstruction task. The experimental results showed that our approach outperformed competitive baselines in all tasks evaluated. Although our method generates a pooling-unpooling map, bijectivity is not assured. A better future approach would be to extend our method to an arbitrary topology pooling/unpooling process, that maintains bijectivity between the fine and coarse meshes.

2.6 Appendix

2.6.1 Curvature and Vertex Normal

One way to come up with normals is to look at the volume gradient. In the discrete case, the volume gradient actually yields a different definition for vertex normals than the one from the area gradient. With vertex normal, our algorithm includes mesh curvature implicitly. We provide mathematical support for this conclusion.

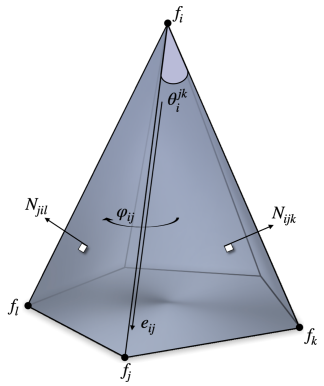


Figure 2.9: Discrete schematic diagram, N_{ijk} is unit normal of triangle ijk, θ_i^{jk} is interior angle at vertex i of triangle ijk, φ_{ij} us dihedral angle at oriented edge ij.

The discrete gaussian curvature (angle defect) at vertex i is the deviation of the sum of

interior angles from the Euclidean angle sum of 2π.

$$K_i = 2\pi - \sum_{ijk} \theta_i^{jk} \tag{2.5}$$

And mean curvature of Mollified surface:

$$H_{ij} = \frac{1}{2}\ell_{ij}\varphi_{ij} \tag{2.6}$$

the discrete Gauss curvature normal

$$\begin{aligned} 2\int_C KNdA &= \int_C dN \wedge dN = \int_C d(N \wedge dN) \\ &= \int_{\partial C} N \wedge dN = \int_{\partial C} N \times dN(\gamma')ds \\ &= \sum_j \int_{\partial C} e_{ij}/|e_{ij}|ds = \sum_j e_{ij}/\ell_{ij}\varphi_{ij} \end{aligned}$$

Similarly, integrating HN over a circumcentre dual cell C yields

$$\begin{aligned} 2\int_C HNdA &= \int_C dN \wedge df = \int_C d(N \wedge df) \\ &= \int_{\partial C} N \wedge df = \int_{\partial e_{ij}^\star} N \wedge df \end{aligned}$$

the ratio of dual length is given by the cotan formula, yielding

$$(HN)_i = \frac{1}{2}\sum_{ij\in E}(cot\alpha_{ij} + cot\beta_{ij})(f_i - f_j) \tag{2.7}$$

two principal curvatures

$$\mathcal{K}_1 = H - \sqrt{H^2 - K}$$
$$\mathcal{K}_2 = H + \sqrt{H^2 - K}$$

Chapter 3

Neural QSLIM for Mesh Autoencoders

Mesh autoencoders rely heavily on fixed quadric sampling schemes, which preserve only topological information, often leading to poor-quality shape up-sampling and overfitting. This paper introduces Neural QSLIM, a novel framework for geometry-aware mesh up-sampling. By randomly generating bijective coarse-to-fine mesh counterparts, our method trains a neural network to learn complex mesh shapes. During decoding, our approach takes a coarse triangle mesh from the bottleneck and reconstructs finer geometry using QSLIM-guided topological updates, while predicting vertex positions with the trained neural network. This design improves the accuracy of vertex positions while maintaining the desired topological structure, rather than relying on simple point projection as in classical mesh autoencoders. We demonstrate that our method enables more effective non-linear mesh up-sampling, resulting in efficient and flexible representations across various mesh autoencoder models.

3.1 Introduction

Mesh autoencoders have become popular for learning latent representations of registered meshes, relying heavily on a multiscale hierarchical representation. This architecture requires convolution-like operations to capture local geometric features on graphs. However, classical pooling methods cannot be applied directly to 3D meshes [14, 15, 16, 17] due to irregular sampling and complex connectivity. Ranjan et al. [20] introduced the CoMA model, which first applied mesh sampling operators for down-sampling and up-sampling in neural networks. In the mesh sampling process, the contracted vertices are stored at their barycentric locations in the transformation matrices during downsampling and later used in the decoder stage. This

approach enables the model to capture both global and local context and has been adopted by subsequent works [22, 53, 24, 50, 54, 55].

Classical mesh autoencoders rely on pointwise loss, which requires the input and output shapes to have the same number of vertices. While MeshCNN [37] and Neural Subdivision [56] are specifically designed for mesh down-sampling and up-sampling, respectively, without user guidance, they alter the mesh topology, leading to features with different dimensionalities. The Fully convolutional mesh autoencoder proposed in [13] achieves high reconstruction accuracy but relies on a large number of parameters, making it computationally expensive compared to more lightweight architectures.

Furthermore, current methods apply mesh sampling once on a template to construct down-sampling and up-sampling transformation matrices, which are then shared across all meshes. This approach leads to template overfitting. An alternative method is to apply mesh sampling individually to all training meshes and average the barycentric coordinates. However, Quadric Simplification (QSLIM) [30, 57] a classical method that simplifies meshes via iterative edge collapses while minimizing geometric error, requires closest-point-on-mesh calculations (similar to the concept of Chamfer distance [58, 59]), which are computationally expensive for high-resolution meshes, and averaging operations cause all projected coordinates to converge toward the centroid, leading to loss of geometric detail. To address these problems, we propose *Neural QSLIM* (as shown in Fig. 3.1), which updates mesh vertices using a transformation matrix while predicting vertex positions with a trained neural network. The transformation matrix preserves mesh topology and adjusts vertex positions based on a neural network conditioned on the mesh geometry. Compared to existing methods, our mesh unpooling eliminates template overfitting. By training across a set of morphable meshes, our approach learns a generalizable rule based on local patches from multiple training meshes rather than relying on a single template.

In summary, our work introduces a mechanism for up-sampling meshes in mesh autoencoders. Our method enhances mesh autoencoder representations by providing more accurate vertex initialization in upsampling. Our main contributions are:

1. We propose a geometry-aware mesh up-sampling method that provides more generalized mesh representations than traditional approaches.

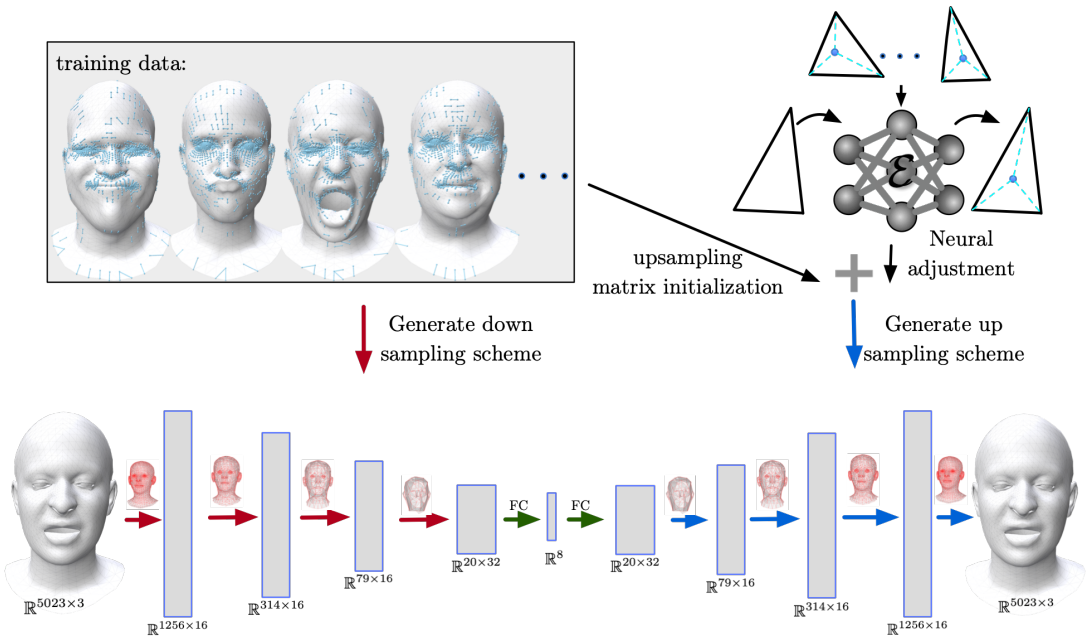


Figure 3.1: Unlike standard mesh autoencoders that build down/up matrices from a single mesh, our method employs a novel coarsening scheme and neural unpooling layer on a fixed topology mesh dataset. The resulting transformation matrices or hierarchies can be seamlessly integrated into mesh generative models without altering the user's network architecture.

2. We show that our method effectively eliminates overfitting caused by sharing transformation matrices across meshes.

3. We demonstrate that a shallow 3-layer multi-layer perceptron (MLP) is sufficient for predicting vertex positions.

3.2 Related works

Our work is designed for geometric deep learning, and its decoder stage resembles subdivision methods. In this section, we introduce previous mesh representation models and compare mesh autoencoders with mesh subdivision approaches.

3.2.1 Geometric deep learning

CoMA [20] introduces the first sampling operations for mesh autoencoders using precomputed down-sampling and up-sampling matrices. Its distinctive pyramid-shaped structure optimizes memory efficiency and enables the model to capture both global and local context.

Following the CoMA architecture, later studies [22, 53, 60, 50, 61] improved mesh convolution networks while inheriting the down-sampling and up-sampling operations. Neu-

ral3DMM [22] and SpiralNet++ [53] replace spectral convolution layers with operators that convolve along a fixed spiral serialization [23] of neighboring vertices, outperforming previous approaches. Alternative implementations, such as [13], define a shared weight basis kernel at every vertex location but come at the cost of increased parameters. MeshCNN [37] instead selects edges as input features and applies filters directly to each edge, making it well-suited for classification and segmentation tasks. In contrast, our work focuses on latent representation and mesh generation.

Previous research improves convolution networks but retains the same down-sampling and up-sampling operations, which rely solely on topological information without considering geometry. Furthermore, constructing the transformation matrix from a single mesh leads to severe overfitting. In contrast, our method proposes an efficient geometry-aware mesh up-sampling operation to enhance the mesh autoencoder.

3.2.2 Comparing with subdivision surfaces

The primary objective of a mesh autoencoder is to learn non-linear representations of meshes. This framework assumes that the input mesh is an exact description of the shape of interest. The output of the decoder should precisely preserve the given shape, even if the input mesh only approximates the true geometry (e.g., 3D scans). If defects such as topological errors or noise are present, these artifacts are also retained by the model. Mesh autoencoders are designed to maintain the topological structure of the input while approximating geometric details as accurately as possible. However, while they effectively learn latent representations of meshes, they do not improve the quality of the underlying geometry.

In contrast, the idea of subdivision is to "define a smooth surface as the limit of a sequence of successive refinements" [62]. However, they are incapable of recuperating lost details or semantic features. Traditional methods like subdivision methods [63, 64, 65] process the mesh by combinatorial update (splitting faces, adding vertices, or/and fipping edges [66]) and vertex smoothing. On the other hand, contemporary non-linear neural subdivision methods [67] and subdivision-based networks (SubdivNet) [68] use fixed and uniform topological updates while leveraging neural networks to capture intricate details. Although SubdivNet introduces pooling and unpooling layers, it cannot be directly applied to arbitrary meshes because it requires the input to maintain subdivision connectivity.

Geometry generation models are generally evaluated based on reconstruction losses, assessing the output mesh's fidelity in approximating a known target. For a mesh autoencoder, both input and output meshes share the same topological structure, and the model aims to minimize vertex-to-vertex loss. In contrast, subdivision-based models typically employ point-to-surface distances, establishing correspondences through closest-point queries. In essence, while subdivision alters the original topological structure, it fails to generate meaningful semantic details from highly coarse meshes. A detailed exploration of this issue is beyond the scope of this paper.

3.3 Method

In this section, we provide an overview of the main components of Neural QSLIM: the construction of the transformation matrix, data generation, and the network architecture.

3.3.1 Build transformation matrix

Although the dataset shares the same topology, the down-sampling results obtained by QSLIM [30] vary significantly. To achieve optimal performance, we contract vertices based on frequency. Specifically, by applying QSLIM to a randomly selected 1% of meshes from the training dataset, we sort the collapsed edges by their occurrence frequency (as shown in Fig.3.2). The most common edges are then used to construct the down-sampling transformation matrix.

Vertices discarded during down-sampling are mapped onto the surface of the down-sampled mesh using barycentric coordinates. Following [20], we denote the down-sampling and up-sampling transformation matrices as $Q_d \in \{0,1\}^{n\times m}$ and $Q_u \in \{0,1\}^{m\times n}$, respectively. For the up-sampled mesh $\mathcal{V}_u \in \mathbb{R}^{m\times 3}$ and the corresponding down-sampled mesh $\mathcal{V}_d \in \mathbb{R}^{n\times 3}$, the relationships are given by $\mathcal{V}_u = Q_u\mathcal{V}_d$ and $\mathcal{V}_d = Q_d\mathcal{V}_u$. By applying QSLIM to different mesh expressions and selecting the most common contracted edges, our method generates a more generalized transformation matrix.

Following CoMA [20], Q_d and Q_u are implemented as sparse selector matrices with binary entries indicating retained or restored vertices. In our method, Q_u provides an initial estimate for upsampled geometry, which is further refined by a shallow MLP to capture local shape variations.

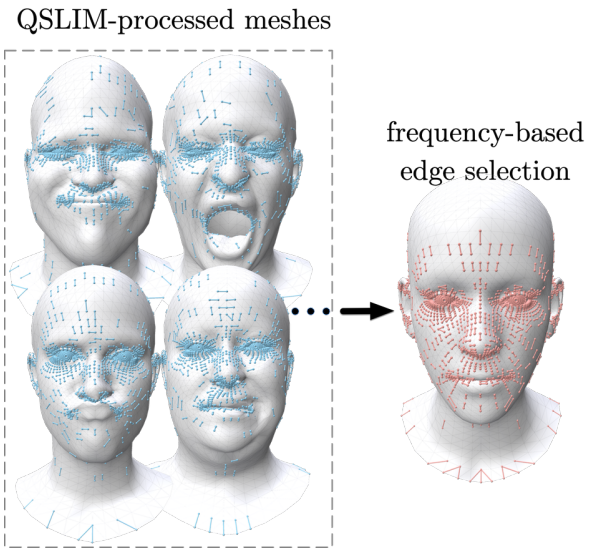


Figure 3.2: QSLIM is applied to the meshes, and edges are selected based on their frequency. Red edges show those contracted during down-sampling.

3.3.2 Data generation and training

Our approach is inspired by neural subdivision, but our upsampling is guided by a precomputed transformation matrix instead of using classic Loop subdivision [64]. This allows our method to generate meshes with any desired number of vertices and work with mesh autoencoders.

We first generate pairs of meshes with different discretizations to train the neural network. Specifically, we apply random edge collapses to create coarse meshes and use bijective mappings to locate corresponding positions on the ground-truth mesh.

Mullen et al. [69] introduced free-boundary conformal parameterizations for a triangular mesh. In the context of an edge collapse, this conformal parameterization [69] can be employed to map the vertices of the 1-ring of the edge into its local UV space, as illustrated in Fig. 3.3. This UV parameterization serves as a verification tool. A critical aspect to note is the shared boundary vertices between the contracted point post-edge collapse and the collapsed edge. Given the UV boundary constraints, the post-collapse vertex retains identical UV boundaries.

We adopt sequential downsampling UV mapping as an alternative to the traditional approach, ensuring accurate triangle projection across each mesh and preventing the inside-out issue associated with naive bundle closest projection. Fig.3.4 illustrates an external vertex

Algorithm 2: Generate Down/Initial-Up Matrix

Input : M, n ▷ selected meshes, target edge count
Output: Q_d,Q_u ▷ down & up sampling matrices

▷ initialization
$E \leftarrow$ EmptyPriorityQueue () ▷ edge frequency
$O \leftarrow$ EmptyVertexBarycentric () ▷ barycentric coordinate
foreach *mesh* $m \in M$ **do**
    $e \leftarrow$ QSLIM (m) ▷ extract collapsed edges
    UpdateBarycentric (O, m, e)
    Enqueue (E, e)
▷ build
while *VertexPopCount*$(Q) < n$ ***and!****Empty* (Q) **do**
    $e_{ij} \leftarrow$ Pop (E) ▷ extract maximum-frequency edge
    UpdateDownMatrix (Q_d, e_{ij})
    UpdateUpMatrix (Q_u, O, e_{ij}) ▷ initial vertex & barycentric coordinate
return (Q_d, Q_u);

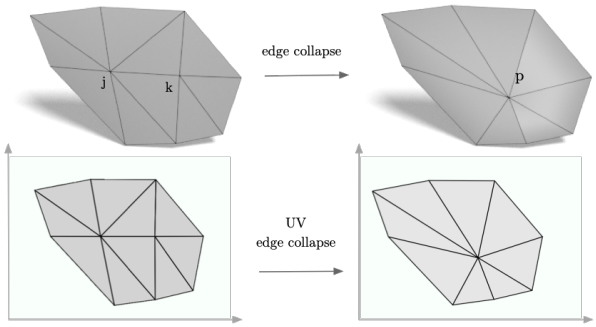


Figure 3.3: Utilizing conformal parameterization to map the 1-ring of a collapsed edge to the UV space. Notably, the vertices post and pre-collapse maintain identical boundary vertices.

projecting onto an internal face along a saddle-shaped surface due to internal heat source interference, disrupting expected heat diffusion along the surface. We employ a highly sparse matrix $Q_d \in \{0,1\}^{m \times n}$, where $m > n$, in the pooling layer to derive coarse meshes. Here, $Q_d(p,q) = 1$ indicates retention of the q-th vertex is kept. Analogous to the template-only method, we compute the transformation matrix Q_d only once. This practice sustains the model's efficiency while enhancing its generalization performance.

3.3.3 Up-sampling

Down-sampling, while essential for mesh simplification, represents only one of the two core components integral to the functionality of the mesh autoencoder model. Given that lossless pooling/unpooling is unattainable, previous methods constructed an un-sampling matrix

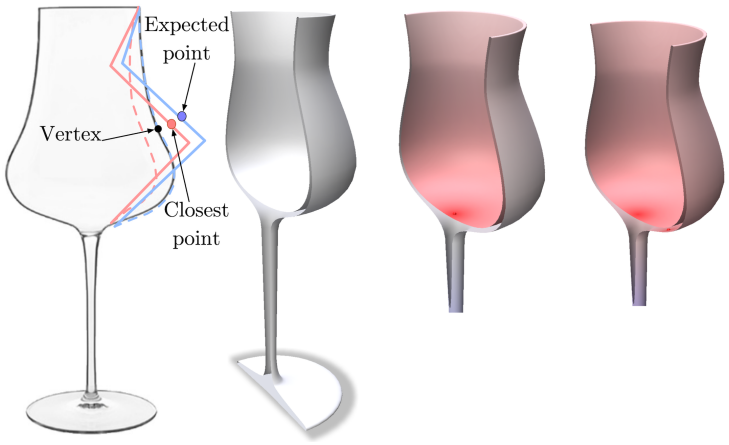


Figure 3.4: Heat diffusion illustrated on a wine glass. Through local mapping, inside-out projection is avoided, resulting in an outward heat spread on the surface (center right), as opposed to the incorrect heat flow caused by closest point projection (right).

during the down sampling phase. In these earlier approaches, discarded vertices of the fine mesh were projected onto the nearest triangle in the down-sampled mesh, a process articulated through barycentric coordinates. However, as depicted in Fig.3.5, templates prove inadequate in capturing varying geometric features. This limitation persists even when identical edge contractions are applied, leading to barycentric coordinates that are not aligned with the template.

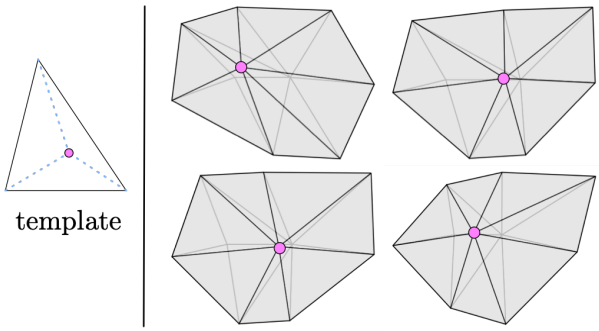


Figure 3.5: Barycentric coordinates resulting from mapping discarded vertices to the closest triangle are depicted. Despite contracting the same edge to the same end, the projection point differs depending on the given geometry. Relying solely on the template representation leads to an inaccurate up-sampling matrix.

Our training data is constructed by applying the transformation matrix decided in down-sample step to selected meshes. During the down-sampling, we collect each triangle and corresponding barycentric coordinates pair. Suppose $\mathcal{E}(v)$ is the barycentric coordinates output by the network $\mathcal{E}$ (Fig. 3.6), and $f(v)$ denote the actual barycentric coordinates. We measure the per-vertex loss with the $\mathcal{L} = ||f(v) - \varepsilon(v)||_2$.

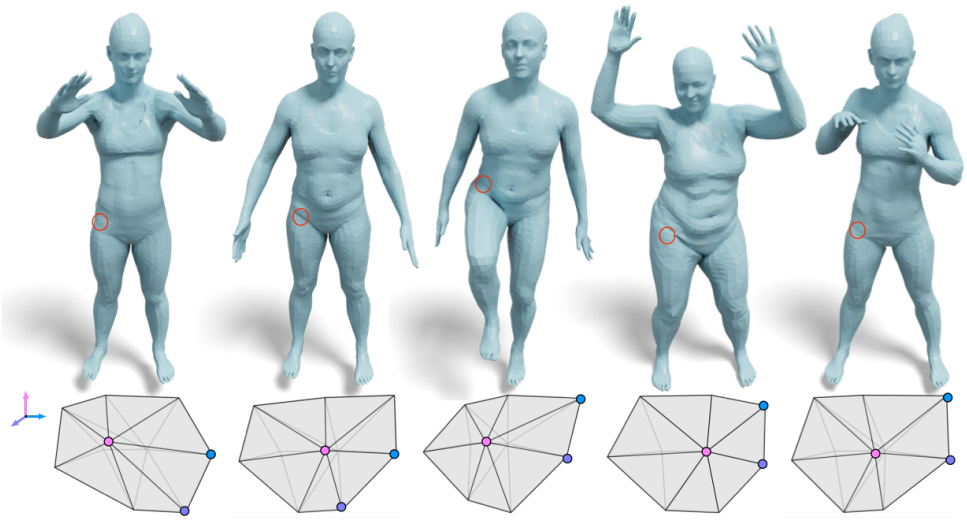


Figure 3.6: Registered data allows us to order the three vertexes of the triangle canonically. We feed vertex features into multi-layer perceptrons (MLP).

Network architecture: Our network is inspired by classical mesh autoencoder which has two operators: (1) mesh sampling from template, and (2) the convolution operator. The two pivotal steps in our process are depicted in Fig. 3.1. The down-sample matrix and the initial up-sample matrix are determined from selected meshes. Simultaneously, we gather each triangle and the corresponding pairs of barycentric coordinates for training the network $\mathcal{E}$. One distinguishing feature of our approach is that we apply training meshes, instead of relying solely on templates. This strategy enables us to utilize neighborhood information derived from a broader range of mesh expressions.

We detail the process through which our model acquires rules from the local patches across all training meshes. During each instance of edge collapse, rather than employing fixed weights from a template mesh, we integrate a training module to the mesh. This allows the network to learn the rules associated with the bijective mapping of the local patch dynamically. Our methodology is characterized by the advantage of constant-cost mapping ("constant-cost mapping" refers to using precomputed barycentric coordinates from the template mesh, which are shared across all training shapes.). We formulate pairs of local coarse and fine meshes and utilize shallow multilayer perceptrons to learn the nuances of the up-sampling mappings.

The final step in our architecture design is the representation of the input and output. Consider a naive approach that use global xyz-coordinates of triangle and projection point pairs. This approach has a major caveat. Cartesian coordinates of a triangle must be described

with repect to some global coordinate system. This lead to incorrect and badly training set up because the rotation and transplation of mesh. To mitigate this issue, we incorporate the use of local coordinates of the mesh, ensuring invariance to rigid transformations. Within the context of this study, the half-edge data structure is employed for mesh storage, offering a unique canonical orientation for the vertices of triangular faces. The local coordinate frame is defined by the three vertices of the mesh, enabling the construction of input features in a canonical manner. This approach underscores each mesh's reliance on its intrinsic coordinate system, a factor that not only expedites the training process but also augments the model's generalization capabilities.

Our refinement network is a shallow MLP with three layers (64, 64, 3) and ReLU activations. We experimented with deeper networks (up to 5 layers) and wider hidden dimensions (128), but observed no significant improvement in accuracy or convergence speed. For training loss, we compared ℓ_2 and Huber loss, and found ℓ_2 slightly more stable across different meshes.

3.4 Applications

In this section, we present a series of experimental results to evaluate our operator. We begin by demonstrating its invariance to rigid motions and its ability to improve reconstruction quality. We commence by illustrating its capability to remain invariant to rigid motions, a feature that notably enhances the quality of the outcomes. Our presentation extends to a visual demonstration, underscoring the superiority of our method in delivering enhanced unpooling results, especially when juxtaposed against the closest point projection in tasks associated with mesh reconstruction. It is imperative to note that, to engender a robust and equitable comparison, we ensured uniformity in the model architectures and maintained constancy in the kernel size across the diverse convolutions under consideration.

We implemented our network using the PyTorch [70]. We use Relu activation [71], and the ADAM optimizer [52] with a learning rate of 0.002. We implemented our technique using the Spectra [72], and tested it on a workstation with an Intel Core i7 2.6 GHz CPU, and 16 GB of RAM. For the mesh reconstruction task, each method was trained and evaluated using a single NVIDIA RTX 3080 Ti. Our method incurs negligible training overhead: the base architecture matches prior work, and the added MLP takes under 10 minutes to train.

In practical applications, it is commonplace for modelers to adjust the mesh into various poses prior to the application of alternative operators. Fig. 3.7 exemplifies the paramount importance of invariant representation in optimizing the performance of a mesh autoencoder. Our findings reveal a notable vulnerability in global-based models; even with training on an identical shape, a minor rigid motion significantly undermines the applicability of the learned weights. In contrast, the incorporation of local coordinates as input features not only enhances the model's resilience to such perturbations but also augments the pace of convergence by approximately 2%-3.5%. While these enhancements may appear marginal, they contribute incremental improvements in the training phase.

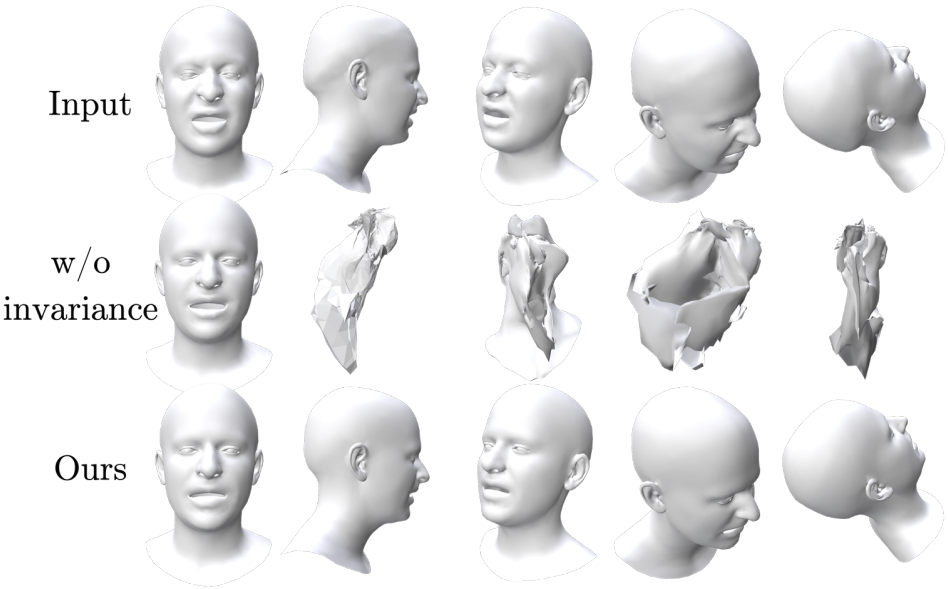


Figure 3.7: Employing local coordinates embedded within each mesh as both inputs and outputs, our methodology achieves invariance to rigid motions. This adaptation yields a marked enhancement in quality relative to preceding methods that lack such invariance.

We further evaluated successive local mapping on mesh generative tasks. For registered data such as CoMA [20] or FAUST [73], modelers often manipulate meshes with different expressions and embed the GNN in a pooling/unpooling layer. Earlier models that employed in-network sampling and shared the transformation matrices of the template across all meshes encountered a pervasive issue of template overfitting. As depicted in Fig. 3.8, we diverge from this approach by upsampling the coarse mesh without relying on the shared weights of the template, demonstrating that our model is adept at offering accurate initial estimations.

Our method for pooling and unpooling is highly compatible with numerous standard mesh processing pipelines. It is engineered to facilitate local bijective mapping while metic-

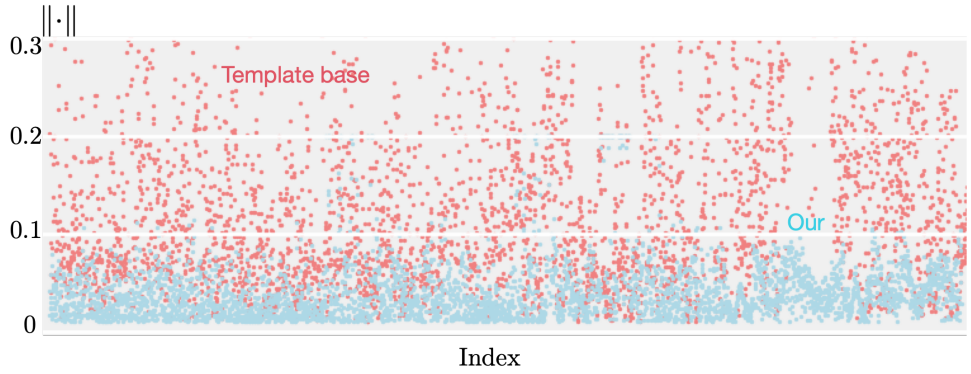


Figure 3.8: By looking at the error in the function of the Euclidean domain of upsampling mesh. We can see that the shared weight cannot handle the diverse mesh. We randomly picked mesh from the CoMA dataset, the reduction from 5024 to 315 vertices (6%) then upsampling by a different strategy. Each point illustrates the error between estimated and original coordinates.

ulously preserving the intrinsic properties of the original mesh. We exemplify this attribute within the context of a mesh autoencoder application.

For a quantitative examination of our network's generalization capabilities to accommodate unseen shapes, we employed the CoMA dataset. This dataset encompasses a diverse collection of over 20K+ 3D meshes, featuring 12 distinct subjects each exhibiting 12 extreme facial expressions, with each registration consisting of 5023 vertices. We split the dataset in training and test samples with a ratio of $9:1$.

We employed a conventional autoencoder architecture, composed of an encoder and a decoder. The encoder is structured as: $3\times\{Conv(16)\rightarrow Pool(4)\}\rightarrow\{Conv(32)\rightarrow Pool(4)\}\rightarrow FC(8)$, accompanied by an ELU activation function post each Conv layer. The decoder mirrors the encoder's structure in reverse order, substituting pooling layers with unpooling layers. The integration between layers is facilitated through a spiral convolution, the efficacy of which is illustrated through a visual comparison of the reconstruction outcomes depicted in Fig. 3.9. A comprehensive list of mean errors, accompanied by standard deviations, is presented in Table 3.1 for an exhaustive comparative analysis.

3.5 Conclusion and Limitations

We proposed a neural-driven quadric mesh unpooling framework that improves vertex positioning and captures complex non-linear transformations across diverse meshes. Our method learns local vertex features to guide downsampling and upsampling, effectively preserving

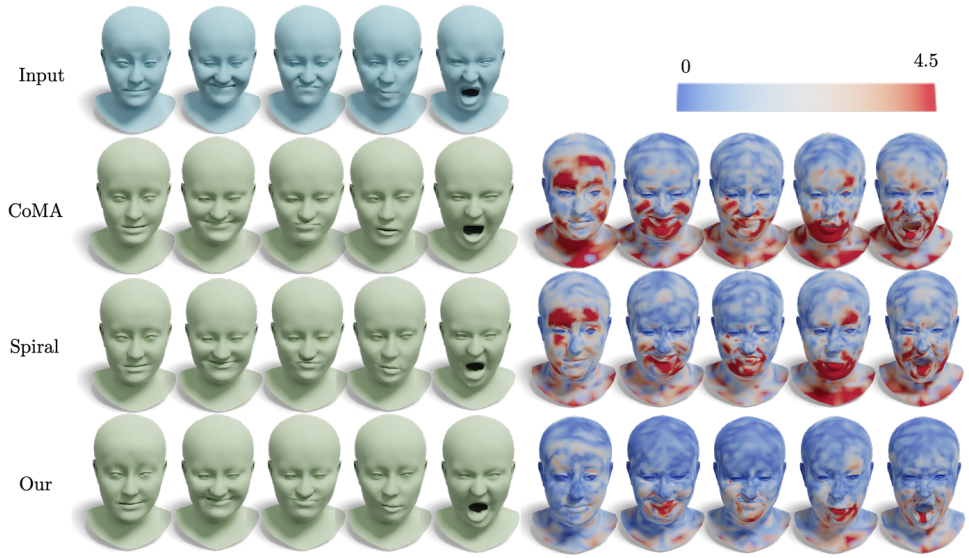


Figure 3.9: Qualitative assessment of reconstruction outcomes. Each vertex's Euclidean error is color-coded, with the error values capped at 4.5 millimeters. Warmer hues denote larger errors.

Table 3.1: Comparison with CoMA and Neural3DMM. Errors are in millimeters. Simplified Neural3DMM with filter size [16,16,16,32].

	Mean Error	#Parameters
Ours	**0.541 ± 0.588**	37,563
Neural3DMM [22]	0.804 ± 0.877	48,067
CoMA [20]	0.845 ± 0.994	33,856
PCA	1.639 ± 1.638	120,552

fine geometric details and mitigating template dependence. Experimental results show that our model outperforms template-based mesh autoencoders, achieving lower point-to-point reconstruction error and better generalization across different mesh expressions. Our current framework assumes topologically consistent, registered meshes during training. As such, it is not directly applicable to partial or noisy scans with missing geometry or varying connectivity. Extending the method to handle topological noise would require rethinking the construction of transformation matrices and refinement targets, which we leave for future work.

Chapter 4

Unsupervised Landmark Discovery via Karcher Means for Non-Rigid Shape Correspondence

This chapter presents an unsupervised intrinsic landmark detection method that enhances non-rigid dense shape correspondence. Our approach identifies intrinsic landmarks on near-isometric surfaces, enabling efficient and robust point-wise correspondence. A key observation is that a small set of intrinsic landmarks is sufficient to reconstruct an isometric map, allowing for effective shape matching. Our method eliminates the need for manually defined landmarks by leveraging intrinsic geometric properties. Specifically, we compute the Karcher means of the surface and sequentially select the farthest points from these medians. This provides a stable and repeatable landmark set, which can be used in various shape correspondence frameworks. We evaluate our approach on multiple shape matching benchmarks and show that it consistently improves the accuracy of existing correspondence methods without requiring modifications. The results demonstrate that our method is computationally efficient and can serve as a plug-and-play component in shape matching pipelines.

4.1 Introduction

Given a point on the first shape, how do we find a meaningful correspondence on all other shapes? This "meaningful correspondence" has also been termed registration, alignment, or shape matching. Establishing correspondences provides a valuable initial stage for a diverse array of applications, including texture transfer, shape analysis, co-segmentation, and exploration, just to name a few. To compute this correspondence, one idea is to identify a set of meaningful points on two shapes and then maximize the resemblance between the descrip-

tors associated with these points. With the aid of rich local descriptors, the correspondence problem for 2D images is well solved. In contrast, constructing stable and repeatable descriptors for plain, non-rigid 3D shapes is highly challenging; even just identifying geometrically salient points with reliable descriptors is already quite costly. Instead of using the original representation of a surface, data-driven methods [74] learn descriptors to induce a more accurate correspondence. However, these methods are hindered by the scarcity of annotated 3D shapes and the associated financial costs.

Identifying an isometric mapping between two shapes is challenging due to the vastness of the potential mapping space. Ovsjanikov et al. [75] introduced a functional map-based approach for non-rigid shape matching, which leverages function correspondences to compute a mapping that facilitates the transfer of real-valued functions between shapes. This mapping is then converted into a point-to-point correspondence. A key advantage of this approach is that computations are performed on a predefined functional basis, significantly reducing the dimensionality of the search space. However, while functional map methods are both powerful and efficient, they do not inherently distinguish among multiple plausible solutions. In addition, the process of converting a functional map into a point-to-point correspondence often involves solving complex quadratic optimization problems.

Many shape matching algorithms [76, 77, 78] have shown that a small number of correspondences is sufficient to recover the full isometry. However, in practice, we often lack even a single known correspondence between shapes. Motivated by this, we propose an efficient algorithm that: (1) generates consistently ordered and salient landmark points, (2) matches shapes undergoing near-isometric deformations, and (3) computes dense shape correspondences efficiently.

While landmarks have been used to improve shape mapping in prior work, our goal is to achieve this without data-driven fitting or user-specified inputs. Our algorithm requires no manually registered datasets or predefined correspondences.

In this paper, we present a geometry-based approach for identifying consistent landmarks for shape correspondence. Our method is built upon two key properties:

- We make no assumptions about the availability of input data such as pre-registered correspondences. Instead, our approach relies purely on geometric processing: it takes

raw shape geometry as input and automatically generates consistent landmarks.

- Our method is general and avoids making specific assumptions about meshes. It can be applied broadly across various shape correspondence algorithms, including functional maps, heat kernel maps, and functional map optimization and refinement methods. This makes it a plug-and-play component for a wide range of shape matching pipelines.

Given a pair of near-isometric meshes, our algorithm automatically generates an ordered set of landmarks to resolve symmetry ambiguities. These consistent intrinsic landmarks can be leveraged in various applications—from improving correspondence accuracy in the functional map framework to enhancing existing optimization methods. To be clear, our method does not resolve ambiguities in perfectly symmetric shapes (e.g., spheres). However, it remains effective on real-world datasets, where exact symmetries are rare. We demonstrate the versatility of our approach across several shape matching algorithms, including functional maps, multiplicative operators, Iterative Closest Point (ICP), and ZoomOut.

4.2 Related Work

The literature on shape matching is extensive [79, 80, 81, 82]. In the following section, we review methods most relevant to ours, focusing on techniques designed to compute dense correspondences between non-rigid shapes or those that can be adapted for this purpose.

Point-wise Methods. Early approaches to non-rigid shape matching focused on directly establishing correspondences between points on two shapes by minimizing an energy function, often based on approximate preservation of geodesic distances [76, 77, 78, 83, 84, 85] or conformality (local angle preservation) [86, 87, 88]. These methods are typically limited to hundreds of points due to high computational cost, a limitation that persists even in recent formulations tackling the quadratic assignment problem [89, 90, 91].

While not scalable to dense matching, these approaches are appealing for their theoretical properties—most notably, that a small number of landmarks can suffice to recover the full isometric map [86, 78]. However, their reliance on user-specified landmarks or training data limits practical applicability. In contrast, our method is fully automatic and produces a consistent ordering of landmarks across similar shapes without requiring any additional input, enhancing downstream correspondence quality.

Functional Maps. Other techniques proposed for non-rigid shape matching fall within the category of methods rooted in the concept of functional maps [75], which has been extended in several follow-up works [92, 93, 94, 95]. These methods significantly reduce the number of variables to optimize by selecting truncated Laplacian eigenbases for the function space defined on each shape and representing the mapping as a matrix that signifies a change of basis. However, converting a functional map to a point-wise map can be difficult and error-prone [96, 97]. These methods implicitly impose restrictive assumptions, such as near-isometry, on the shapes being analyzed, and often encounter challenges in practice, leading to low-quality outcomes and limited surjectivity. Vestner et al. [98] proposed a filter to improve the quality of the input mapping, and Huang and Ovsjanikov [99] introduced adjoint regularization, both ensuring a bijective correspondence between the shapes. Nevertheless, the application of these techniques is still limited as they rely on hand-picked initial matches for initialization and fail when there are no annotated landmarks.

Descriptor Learning. The performance of functional maps is heavily reliant on accurate descriptors and precise hyper-parameter tuning [100]. In practice, constructing a stable and repeatable descriptor is highly challenging. Dense, specific dimensional descriptor fields on two shapes are typically employed as corresponding functions. Examples include GPS [101], Heat and Wave Kernel Signatures (HKS and WKS) [102, 103], SHOT [104], and geodesic distance descriptors [105]. Litman and Bronstein [106] introduced a data-driven spectral descriptor model that generalizes the HKS and WKS. Litany et al. [74] further improved descriptor quality by introducing a deep Functional Maps Network (FMNet) for shape correspondence, which takes base descriptors as input and returns matches. FMNet refines the input descriptors by minimizing a geometric loss, yielding soft correspondence. One of the advantages of this supervised method [107] is its versatility, capable of handling different shape discretizations. However, it has the inherent drawback of requiring training data. It is worth mentioning that, even though they are termed "unsupervised methods" [108, 109], these approaches are still fundamentally data-driven. Within the Ovsjanikov et al. [78] framework, our method consistently enhances the performance of the above models.

Iterative Map Refinement. Intrinsic symmetries, such as left-to-right symmetry, can cause functional maps to yield mixed results from multiple potential solutions. The Iterative Closest Point in spectral embedding was employed in the original functional maps approach [75]

for post-processing, effectively mapping vertices from the source shape to the target shape. Typically, the ambiguity arising from symmetry is resolved using user-specified landmarks or training data [45]. However, this diminishes the practical utility of the ensuing pipeline. ZoomOut [110] and subsequent works [111, 95, 112] enhance results quality through the iterative updating of functional and point-to-point maps. Nonetheless, these methods face scalability issues with an increasing number of vertices; for instance, BCICP [93] takes an average of 2 minutes on meshes with $1,000$ vertices and exhausts memory around 30K vertices. Absent external information, the intensive computation required for basis pre-computation and iterative updating of point-wise maps [100] renders these methods impractical for real-time applications.

4.3 Preliminaries

The core idea of our method is to improve dense shape matching through ordered intrinsic landmarks, computed automatically without any manual annotations. We first provide a concise overview of the key steps in computing functional maps and their properties, then describe a set of fundamental building blocks that are integrated to form the final algorithm in section 4.4.

4.3.1 Notation

Given two shapes M and N, typically represented as triangle meshes, let $T\colon N \to M$ be a bijective mapping between the manifolds M and N. Given a function f defined on shape M, $f\colon M \to \mathbb{R}$, we can transfer f onto N through composition using T, defining $g = f \circ T$. Here, $g\colon N \to \mathbb{R}$. We use W and A to denote the cotangent weights and area matrices, respectively, which collectively define the positive semi-definite Laplace-Beltrami operator as $L = A^{-1}W$. $d_M(x, y)$ represents the shortest geodesic distance between any two points $x, y \in M$. TM denotes the tangent bundle of M, which is the set of all tangent vectors.

4.3.2 Basic Pipeline

Let M, N contain, respectively, n_M and n_N vertices. The fundamental workflow for computing a map using functional maps typically involves the following main steps, as outlined in Chapter 2 of [113]:

1. We compute a fixed set of basis functions for shapes M and N, storing them as columns in matrices Φ_M and Φ_N respectively. These matrices are of sizes $n_M \times k_M$ and $n_N \times k_N$, where k_M and k_N are much smaller than n_M and n_N respectively.

2. We compute q descriptor functions on each shape, and project them onto the truncated Laplace-Beltrami basis to obtain coefficient vectors f^i and g^i. These coefficients form the columns of matrices F and G, of size $k_M \times q$ and $k_N \times q$, respectively.

3. Solving the functional map C via

$$\arg\min_C \|CF - G\|^2 + \alpha\|\Delta_N C - C\Delta_M\|^2. \tag{4.1}$$

   Where Δ_N and Δ_M are the Laplace-Beltrami operators of the two shapes, represented in their respective bases, and α is a small scalar weight. In some models, the second constraint is omitted.

4. Deriving a point-to-point correspondence from a given functional map matrix C.

Although a small number of Laplace-Beltrami basis functions can be used to represent each shape, the resulting linear transformation matrix between the two function spaces is only an approximation of the true map. However, acquiring a continuous point-to-point map through this workflow has been found to be both difficult and error-prone. In step (3), factors such as descriptor preservation do not disambiguate different possible solutions due to the symmetries of the shapes. It is worth noting that step (4) leads to difficult quadratic optimization problems, making the framework less practical for real-world use. For example, PMF [98] solves an equation at each iteration to determine a bijection, involving high algorithmic complexity and also requires the shapes to possess an identical number of vertices. BCICP runs out of memory for meshes around 30K vertices and takes 1 minute per shape pair with 600 vertices (Section 5.2 [114]). These two challenges are closely related. One can accelerate the model by obtaining a better C from step 3 and thereby reducing the need for step 4.

Prior work has shown that only a few landmarks are sufficient to recover a full isometric map. Previous methodologies relied on external information, such as user-specified landmarks. In this study, we leverage the intrinsic structure of isometries between a pair of shapes. We show that under near-isometric conditions, it is possible to identify significant landmarks on both shapes.

4.3.3 Geometric Medians

To compute ordered intrinsic landmarks, we rely on log and exponential maps to define Karcher means, and on geodesic distances as a core metric. While the previous section outlines the basic functional map framework, these geometric tools are essential for our landmark strategy, enabling the extraction of representative points that respect the intrinsic structure of the shape. We consider several important implementations from geometry processing that are used in our algorithm. To maintain a focused discourse, We restrict our discussion to triangulated surfaces.

Logarithmic Map. The logarithmic map is a special 2D parameterization of a surface. At any point x on a smooth surface M, the exponential map [115], denoted as $\exp_x(rv) : T_xM \rightarrow M$, provides the point $y \in M$ reached by traversing a geodesic in the direction v (a unit vector) for a distance r. The logarithmic map $\log_x(y)$ is the inverse operation which is summarized in algorithm 3. Intuitively, we can imagine that exp "wraps" the tangent plane around the surface. Logarithmic map provides a tool of "translation" for curved domains. It enables us to easily compute an average of points on a surface by mapping all points to the tangent plane using the log map.

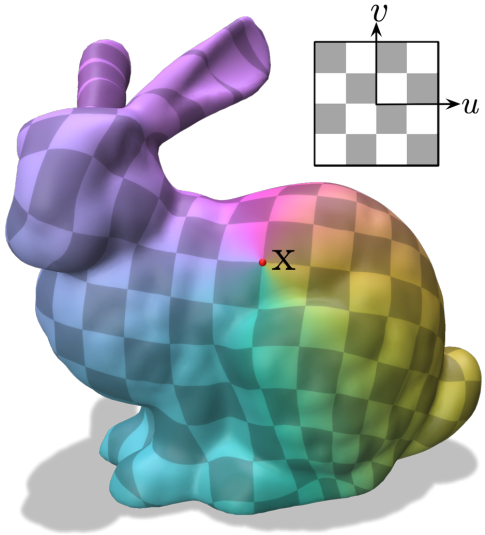


Figure 4.1: The logarithmic map refers to a specific 2D parameterization of a surface, relative to a chosen point x. A global logarithmic map facilitates the translation of algorithms from the Euclidean space to curved domains.

Geometric Medians. If we think of the points in the plane, the average center of points can be easily calculated using the Euclidean mean. However, for a curved surface, the average

Algorithm 3: Logarithmic Map

Input : A point $x \in M$ and a zero direction $H_0 \in T_xM$.
Output: A function $(u, v) : M \to \mathbb{R}^2$

1 Find the shortest geodesic R from x to target point y.
2 Compute the angle ϕ between H_0 and R.
3 Construct $\exp_x(R) = y$; equivalently, $\log_x(y) = R$.
4 Compute $(u, v) = |R|(\cos\phi, \sin\phi)$ at point y.

of points may not necessarily lie on the surface. Given a set of points $y_1, \ldots, y_n \in M$, any point that minimizes the energy function can be considered a center.

$$E(x) := \frac{1}{2n} \sum_{i=1}^{n} d(x, y_i)^p. \tag{4.2}$$

For $p = 2$, centers on the surface are called *Karcher means* or *Frechet means* if unique. For $p = 1$, centers on surfaces are known as *geometric medians*. Both *geometric medians* and *Karcher means* can be efficiently computed via the log map in two steps (as shown in fig. 4.2), where $\tau > 0$ is the step size in algorithm 4.

$$v \leftarrow \frac{1}{n} \sum_{i=1}^{n} \log_{x^k}(y_i), \qquad x^{k+1} \leftarrow \exp_{x^k}(\tau v). \tag{4.3}$$

In general, an average on a surface is not unique; the resulting points may vary depending on the initialization.

Algorithm 4: Geometric Mean

Input : A set of points $y_1, \ldots, y_n \in M$.
Output: A point $m \in M$

1 Random initial point m^k from the set M.
2 **while** $|v| < \varepsilon$ **do**
3 **foreach** $y_i \in \{y_1, ..., y_n\}$ **do**
4 Compute the logarithmic map at m^k via algorithm 3.
5 Update vector v += $\frac{1}{n} \log_{m^k}(y_i)$.
6 Update $m^{k+1} = \exp_{m^k}(\tau v)$, which represents moving forward along the vector v from point m^k for a time duration of τ.

Geodesic Distances. In our algorithm, we need to compute geodesic distances between a large number of point pairs, making it essential to adopt an accurate and efficient distance

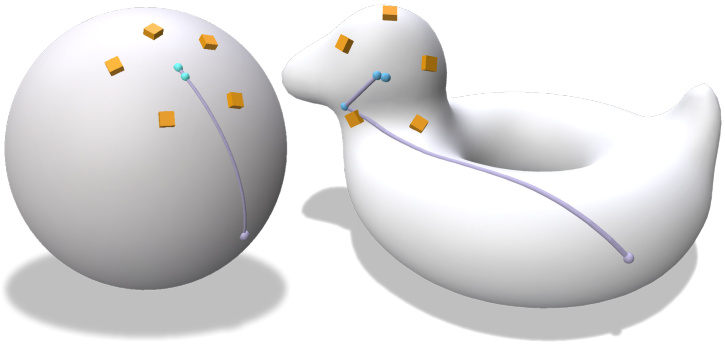


Figure 4.2: In the case of the Karcher mean, given a collection of points, we can get the expected solution in just 2–3 steps; here we show selected points (orange), and the initial guess (purple).

computation method. We compare the heat method with Dijkstra and Fast Marching in fig. 4.3 to highlight its advantages in smoothness and efficiency, motivating its use in our pipeline. We use Heat Kernel Maps (HKM) to construct robust pointwise descriptors that capture intrinsic geometric information around each landmark. Specifically, for a given shape, we compute the HKM at each selected landmark and treat these as scalar functions defined over the surface. These functions are then projected onto the Laplace-Beltrami eigenbasis to obtain compact functional descriptors. When applied to a pair of shapes, the descriptors derived from corresponding landmarks are used to compute the functional map via least-squares fitting. This integration allows HKM to serve as a reliable bridge between intrinsic landmarks and the functional representation, improving the consistency and accuracy of the resulting correspondences.

4.4 Algorithm

We now have all the ingredients to implement our method using the tools from section 4.3. Unlike many existing approaches, our method is entirely unsupervised and does not require any manual initialization or user-provided landmarks. Instead, we rely solely on intrinsic geometric information to automatically identify a consistent set of intrinsic landmarks.

In this section, we present our full unsupervised pipeline for intrinsic landmark discovery and dense shape correspondence. The process consists of three main steps: (1) compute pairwise geodesic distance maps using the heat method; (2) extract intrinsic landmarks via robust geometric center estimation (section 4.4.1); and (3) use these landmarks to guide downstream correspondence refinement (section 4.4.2). We now describe each component in detail.

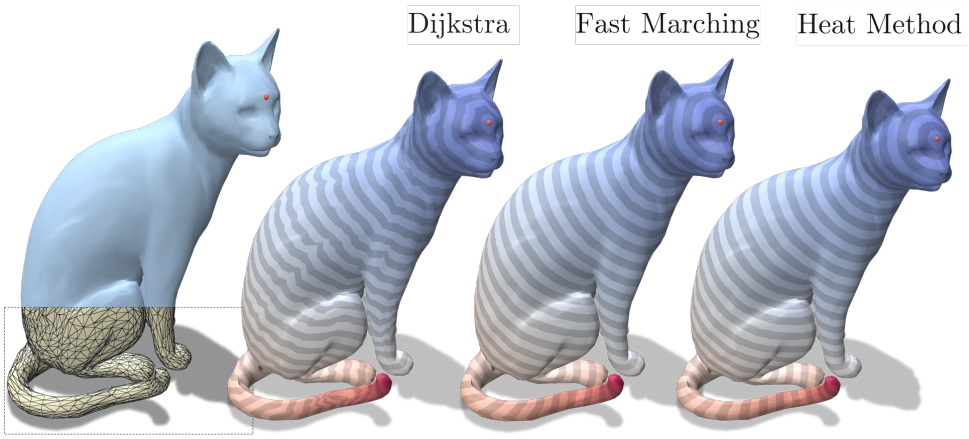


Figure 4.3: Comparison of geodesic distance fields computed using Dijkstra, Fast Marching, and the Heat Method. The heat method, based on solving linear elliptic PDEs, produces smoother distance isolines with fewer numerical artifacts, especially on irregular meshes. According to benchmarks by Crane et al. [116], the heat method achieves up to 68× speedup over Fast Marching on large meshes (e.g., 1.45s vs. 98.1s on a 1.6M-triangle model) with comparable or better mean error.

4.4.1 Ordered Landmarks

We refer to ordered landmarks as the selected points ranked by their average geodesic distance from the set of Karcher means. We now introduce our primary algorithm, the **intrinsic landmarks method**, which is summarized in algorithm 5. The basic idea is to first calculate a set of Karcher means from various starting guesses. To obtain the landmarks, we compute the average geodesic of all vertices relative to this set and sort the list. In fig. 4.4, we show the results of Karcher means computation from different initializations, demonstrating the consistency of the selected landmarks across a variety of shapes.

Consider a surface where salient points have been identified and a Karcher mean m has been computed. For any set of salient landmarks $p_1, p_2, \ldots, p_k$ on surface M, the geodesic distance between any two of these landmarks, represented as $d(p_i, p_j)$. This value should be greater than or equal to the maximum geodesic distance between any landmark and the Karcher mean. This can be formulated as:

$$d(p_i, p_j) \geq \max d(p_l, m), m \in K. \tag{4.4}$$

Let K denote the set of Karcher means on the surface M. The hypothesis suggests that the

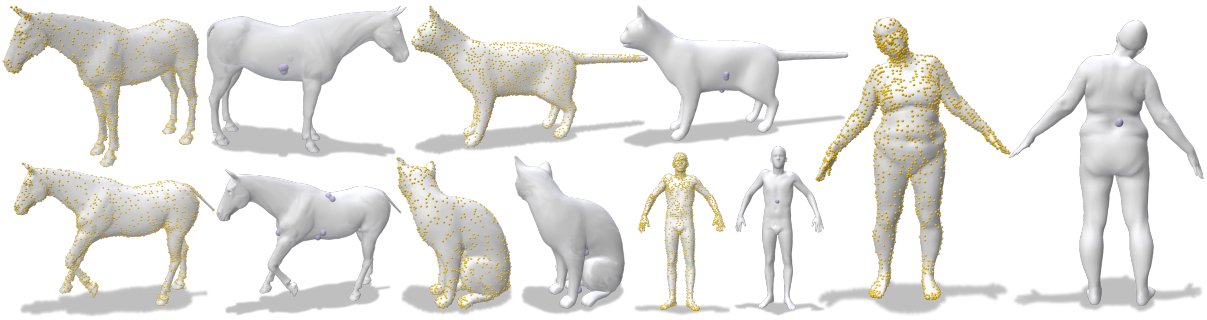


Figure 4.4: We compute the *Karcher means* of the yellow points, which were obtained from a substantial number of initial guesses (covering over 90% of all vertices) and are indicated by the purple points. While there can be more than one Karcher mean, we demonstrate here with five center points. In some cases, iterations push the solution towards a local minimum or yield only approximate averages. For human shapes, however, the Karcher means are highly stable.

Karcher mean serves as a central reference, being closer to each salient point than any pair of salient points are to each other. This hypothesis might not be applicable to non-generic manifolds. The sphere is a classic example of a non-generic manifold, for which counterexamples are easily found. For instance, as shown in fig. 4.2, the geodesic distance between two points marked in orange boxes could be less than the distance from any orange box to the Karcher mean (depicted in turquoise). However, most real-world shapes, such as human bodies, possess only reflectional symmetry. Salient points on these shapes are typically stable and reliable in practical applications, as demonstrated in fig. 4.6.

Let us first note that incorporating landmark constraints into the functional map formulation does not alter its runtime complexity, as it still reduces to solving a least-squares problem [93]. Although we do not provide a formal generalization analysis for eq. (4.4), we observe that our algorithm consistently yields stable, ordered landmarks across various shapes. This helps mitigate key factors affecting map quality, all with negligible preprocessing overhead.

For instance, on a mesh with $7K$ vertices, generating 5 ordered landmarks using 10 Karcher mean samples takes approximately 2.37 seconds for Karcher means computation and 0.015 seconds for landmark selection. While few existing methods perform analogous preprocessing, our approach remains lightweight and can be readily integrated into existing pipelines—potentially improving correspondence accuracy or accelerating convergence. Benefiting from the efficiency of the heat method, our approach remains lightweight and readily integrates into existing pipelines—potentially improving correspondence accuracy or accelerating convergence. Even on a mesh with 100K vertices, computing a single Karcher mean takes

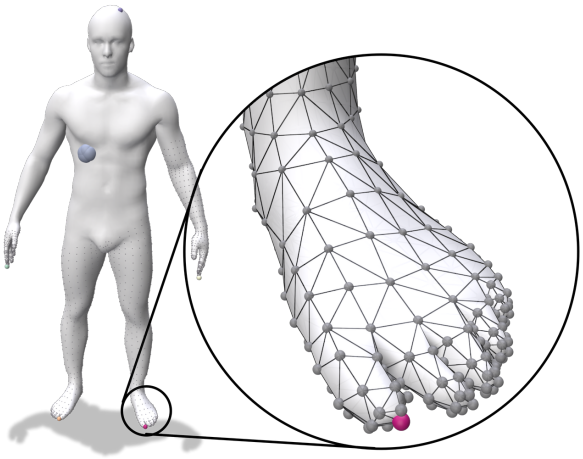


Figure 4.5: Selecting the vertices with the greatest geodesic distance from the set of Karcher means, shown in light blue
, while also verifying if the point satisfies eq. (4.4). In the image, the large purple sphere represents the set of Karcher means. Although the gray points have large geodesic distances to the Karcher mean, they are too close to existing landmarks and thus not selected. The primary landmarks are highlighted in red (visible in the zoomed-out view).

only 1.13 seconds, confirming the scalability of our method. Compared to prior optimization-based pipelines that require user-specified landmarks or extensive learning-based training, our method offers a significantly faster and fully automated alternative. In practice, we set the number of landmarks n to a small value (typically 3 to 5), which balances coverage and stability across shapes.

On meshes containing approximately $30K$ vertices, the computation of the *Karcher means* can become relatively slow, often taking more than 60 seconds. In these cases, it may be advisable to initialize the Karcher mean with a smaller set of points. Nonetheless, the efficiency of landmark generation remains unaffected because the *Logarithmic Map* is only applied to a select set of landmarks, which guarantees quick processing.

While this mechanism already performs quite effectively, we can enhance both robustness and accuracy by adjusting the vertex density according to the geometry of the input mesh. The basic idea is to construct an alternative Karcher mean energy for the same piecewise Euclidean surface which leads to better average points for subsequent operations. In this case, we can define a central energy.

$$E(x) := \frac{1}{2n} \sum_{i=1}^{n} \rho(y_i) d(x, y_i)^p. \tag{4.5}$$

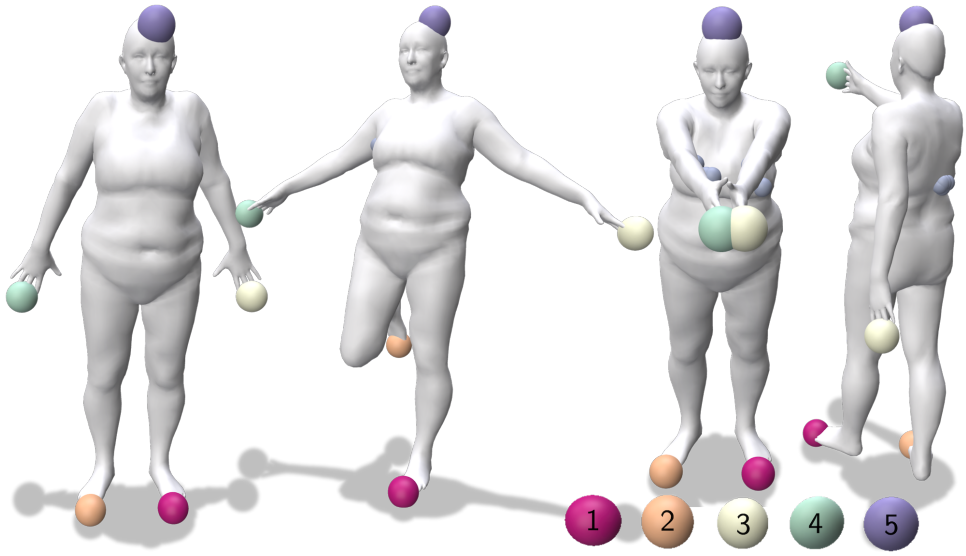


Figure 4.6: Our algorithm efficiently provides consistent ordered landmarks, facilitating the most shape correspondence algorithm

Algorithm 5: Ordered Landmarks

Input : M, n ▷ selected meshes, # landmarks
Output: L ▷ Ordered landmarks

▷ initialization
$K \leftarrow$ KarcherMean(M) ▷ algorithm 4
$D \leftarrow$ EmptyList() ▷ geodesic to m
foreach *mean* $k \in K$ **do**
  $l_k \leftarrow$ LogarithmicMap (k) ▷ calculate logarithmic map
  foreach *vertex* $v \in M$ **do**
    $D[v] += l_k[v]/n$ ▷ vertex geodesic to mean
  Sort(D) ▷ descending order
▷ build landmarks
while $\#L < n$ ***and*** *notEmpty(D)* **do**
  $v \leftarrow$ Pop (D) ▷ extract maximum geodesic point
  if *isEmpty* (L) ***or*** $d(v, l) \geq D[v] \quad \forall l \in L$ ▷ *eq.* (4.4)
  then
    L.append (v)
return L

The only modification to the algorithm involves taking a weighted average across all vertices, where $\rho(y_i) > 0$ denotes the density at each vertex. In our implementation, the weights $\rho(y_i)$ are derived from the vertex dual areas, and they are normalized implicitly during the update step. We use a fixed initial step size $\tau = 1.0$ and a convergence threshold based on local face size (1/3 of the face scale). To ensure stability across different initializations, we run Algorithm 2 multiple times (typically 10) with different random seeds and retain the resulting set of stable Karcher means. This strategy improves robustness and reduces the influence of local minima. In fig. 4.7, we construct a stress test by employing an extremely uneven triangulation. Despite these conditions, our algorithm capitalizes on the advantages of area-weighted geodesics to derive more robust landmarks.

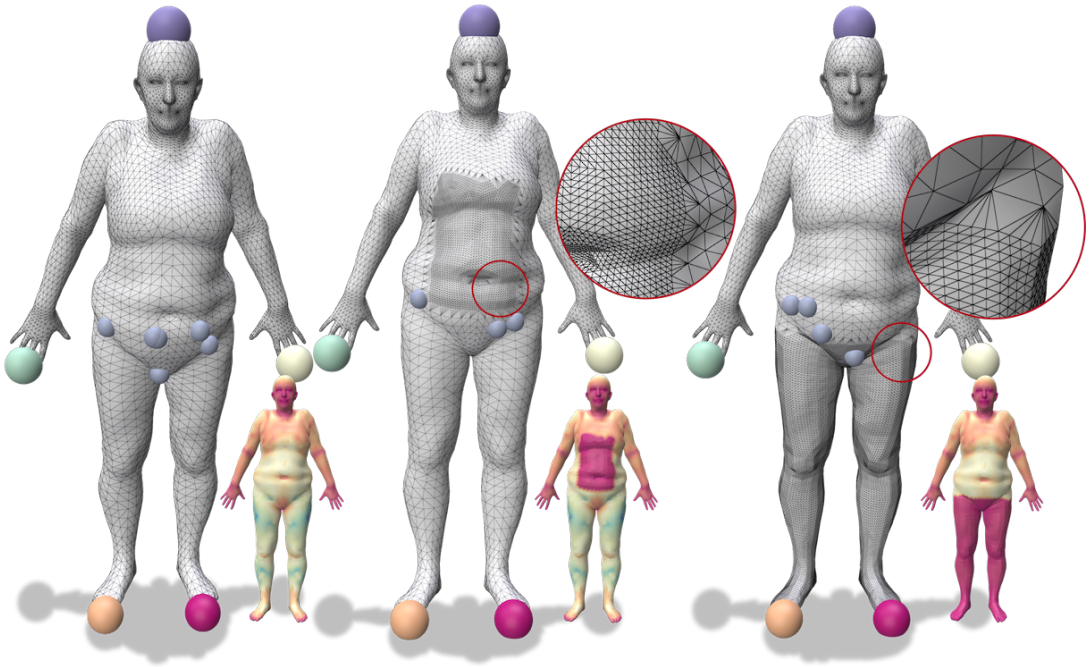


Figure 4.7: We subdivided a section of the mesh and compared our method under various scenarios. Our approach demonstrates insensitivity to input triangulation, ensuring that the landmarks remain robust against nonuniform vertex density. The red color represents areas of high vertex density (small vertex dual area). The light blue spheres indicate the Karcher means computed from the input mesh.

4.4.2 Correspondence Augmentation

Previous algorithms that address symmetry ambiguity typically depend on user-specified landmarks or incorporate additional optimization constraints, such as multiplicative operators [117] or orientation-preserving operators [93]. In contrast, our method relies solely on geometric processing to generate ordered landmarks. Our algorithm operates in two distinct

stages: initially, it establishes a select number of landmark correspondences using algorithm 5, and then it extends these to a dense map encompassing the entire shape. The heat kernel map theory posits that even a single correspondence can be adequate for isometry recovery, as detailed in Theorem 3.1 of [78]. Furthermore, our algorithm's provision of multiple correspondences significantly bolsters stability.

Perhaps the most closely related work to our method is the study by Ovsjanikov et al. [78]. They demonstrated that a single correspondence can be utilized to reconstruct an isometry defined across entire shapes [78]. The original heat kernel map utilizes a related, but fundamentally different technique from our algorithm: the former requires user-specified landmarks, while the latter automatically generates ordered landmarks.

To begin with, it's important to remember that the heat kernel map associated with any point x in M and a predetermined source point p is defined as follows:

$$\Phi_p^M : M \longrightarrow F,\ \Phi_p^M(x) = k_t^M(p, x), \tag{4.6}$$

where F is the space of functions. Essentially, the heat kernel $k_t^M(p, x)$ gauges the heat transfer from point p to point x over time t. It's noteworthy that the source point p can be any point on the manifold M. For each point x in M, we discretize $k_t^M(p, x)$ into a collection of time samples $\{t_j\}$. Consequently, for every point, we compute a vector of size $J = |\{t_j\}|$, where the j^{th} component is $k_{t_j}^M(p, x)$. Analogously, for y in N, we calculate $k_{t_j}^M(q, y)$. Subsequently, we evaluate the correspondence by computing the squared distances between the heat kernel maps:

$$E(p, q_i) = \sum_{x \in M} \min_{y \in N} ||\Phi_p^M(x) - \Phi_{q_i}^N(y)||^2. \tag{4.7}$$

We now understand that certain landmarks $p = \{p_1, p_2, \ldots, p_k\} \in M$ correspond to landmarks $q = \{q_1, q_2, \ldots, q_k\} \in N$. For the landmark correspondence (p_k, q_k), we can allocate to each point x a combined heat kernel map, where the additional J coordinates represent $k_{t_j}^M(p, x)$. In practice, we incorporate each correspondence (p_k, q_k) by minimizing the combined error:

$$f(x) = \arg\min_{y \in N} \|\Phi_p^M(x) - \Phi_q^N(y)\|. \tag{4.8}$$

Although the heat kernel map introduces additional coordinates, all the resulting quantities remain commensurable [118]. Consequently, we can employ the same distance metric for their comparison.

4.5 RESULTS

We have implemented the proposed algorithms in C++. We first detail the competing methods and their implementations. Subsequently, we describe the datasets used and present a series of both quantitative and qualitative evaluations. We demonstrate the efficacy of intrinsic landmarks across a diverse array of shapes and illustrate how our approach significantly enhances the performance of other algorithms.

Description of competing algorithms. We conduct a comparative analysis of our method, termed *Intrinsic Landmarks Correspondence (ILC)*, against various competing algorithms.

- ICP: Uses ICP to refine soft correspondences, following the approach outlined in the original functional maps framework [75].
- ZoomOut [110, 100]: Improves mappings or correspondences by iteratively upsampling in the spectral domain within the functional map framework.
- Multiplicative Operator Commutativity [117]: Views descriptors as linear operators that act on functions via multiplication, incorporating descriptor preservation constraints formulated via commutativity with the unknown map.
- symmOP + BCICP [93]: Improves the basic ICP by formulating orientation preservation in the spectral domain, enhancing performance in both the spatial and spectral domains.
- (WKS/HKS) [102, 103] Initialization: Employs the basic functional map framework with multiplicative operators to compute initial soft correspondences, employing region-based correspondences computed by an automated algorithm as descriptors.

Datasets. We utilize various standard datasets to compare the different techniques: FAUST, TOSCA, and SCAPE.

- FAUST: The dataset [119] comprises 300 triangulated meshes of 10 distinct subjects, with each subject scanned in 30 different poses.

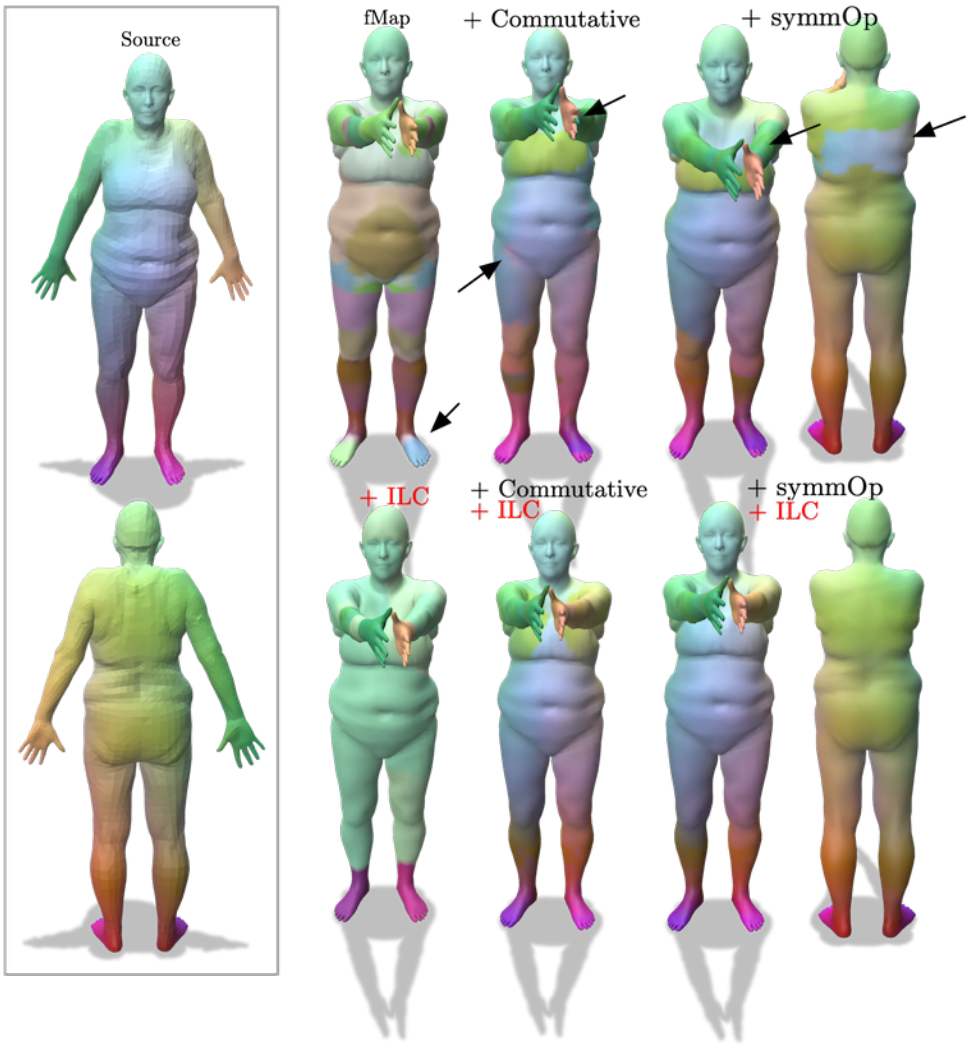


Figure 4.8: Comparison of dense correspondence results with and without intrinsic landmarks. The two shapes in the leftmost column serve as the source reference for color transfer. The top row shows the results without using intrinsic landmarks across three functional map frameworks (fMap, + Commutative, + symmOp), where the black arrows highlight significant mapping errors. The bottom row shows the corresponding results enhanced with our intrinsic landmark constraints (indicated by red "+ IL"), leading to consistently improved correspondences across all methods.

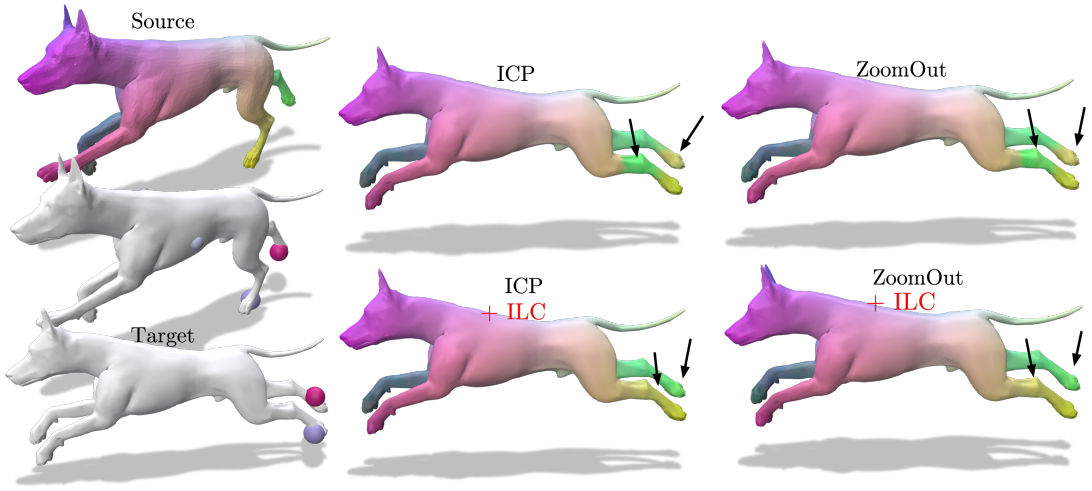


Figure 4.9: Qualitative comparison on the TOSCA dataset. The leftmost shapes are the source and target meshes, with two identified intrinsic landmarks indicated by red and purple spheres. The top row shows the correspondence results from ICP and ZoomOut without using intrinsic landmarks, while the bottom row shows the results with our intrinsic landmark constraint (denoted by red"+ IL"). Black arrows highlight areas where the original methods produce visible misalignments, which are corrected by our method. Our approach improves the accuracy of correspondences with minimal preprocessing time (¡2s) and captures finer structural details.

- TOSCA: The TOSCA dataset [120] consists of 80 meshes categorized into nine groups. Each category comprises objects with shared triangulation and an equal number of vertices, numbered compatibly.
- SCAPE [121]: comprises human shapes only. The datasets contains 71 registered meshes of a particular person in different poses.

Metrics. We assess various methods using standard metrics for dense shape correspondence [93]. Ground truth correspondences are provided in each dataset. To evaluate the continuity of the dense map $T : N \rightarrow M$, we compute the geodesic distance between $T(x)$ and $T^*(x)$ for all $x \in N$, where T^* represents the ground truth map.

We first demonstrate a correspondence augmentation that seamlessly integrates into the basic functional map pipeline, as depicted in eq. (4.1). In fig. 4.8, we utilized 35 eigenvectors to compute the HKS descriptors for initializing a functional map. The term +Commutative refers to adding the term $w_{d-comm} \sum_i \|C\Gamma_1^i - \Gamma_2^i C\|^2$ [117], while +symmOP denotes the addition of the symmetric operator $w_{orient} \sum_i \|C\Lambda_1^i - \Lambda_2^i C\|^2$ [93] to the basic pipeline. $+ILC$ indicates the inclusion of our five landmarks as mentioned in section 4.4.2. Here, the multiplicative operators Γ_1^i and Γ_2^i, as well as the orientation-preserving operators Λ_1^i and Λ_2^i, are associated with

Table 4.1: Quantitative assessment of refinement in shape matching. Beginning with a computed 20×20 functional map. The datasets include 190 FAUST pairs and SCAPE.

Method	Average Error ($\times 10^{-3}$)	Average Runtime (s)
Init	67.5	-
PMF [98]	25.9	311.2
ICP [75]	54.4	12.2
BCICP [93]	22.3	113.5
ZO [110]	20.1	87.4
Scalable ZO [100]	21.7	16.3
Ours + ICP	**47.3**	14.3
Ours + BCICP	**19.2**	115.2
Ours+ ZO	**18.5**	89.3
Ours+ Scalable ZO	**20.9**	18.9

the i^{th} descriptors. Incorporating the proposed landmarks results in a significant improvement in all three cases.

In addition, we evaluate the utility of our method to enhance various refinement techniques, as discussed. We compare our method to the following: We apply ICP to a functional map of dimensions 35×35, estimated using the same pipeline adopted for the initial map. ZoomOut, we refine the initial functional map and upsample this map with a step size of 1.

From the results in fig. 4.9, it is evident that the intrinsic landmark operators are highly beneficial. Our approach not only achieves quality improvements compared to all raw methods but is also significantly faster. In fig. 4.9 and fig. 4.8, arrows highlight regions where our intrinsic landmark augmentation improves the dense correspondence, especially in symmetric or highly articulated areas.

Many shape refinement methods exhibit poor scalability with the number of vertices. As discussed in section 4.4.2, the additional landmarks introduced in our method incur nearly negligible computational cost. The main advantage of our approach becomes evident when combined with other correspondence frameworks, as mentioned. Since our algorithm doesn't entail model training or n-dimensional matrices, its runtime scales linearly with the original number of randomly selected vertices.

Table 4.1 reports the average error and runtime. Our method achieves a 13.1%, 16.1%, 8.0%, and 3.7% improvement in accuracy over the state-of-the-art while incurring only a couple of seconds in computational cost. Furthermore, our method demonstrates greater versatility, enabling seamless integration into several existing frameworks, in contrast to typical

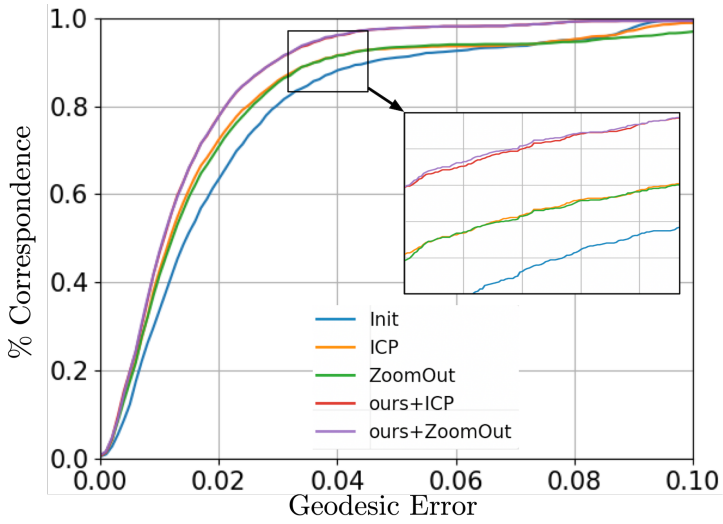


Figure 4.10: Evaluation on Dynamic FAUST using geodesic error measures. We present ICP and ZoomOut predictions both before and after applying our augmentation. The initialization parameters are set to $w_{d-comm} = 0.1$ and $w_{orient} = 0.1$.

algorithms. In fig. 4.10, we present the results of the ICP and ZoomOut methods both with and without our augmentation. It's noteworthy that both methods exhibit improvements when our intrinsic landmarks are applied.

4.5.1 Ablation Study

In this section, we evaluate the robustness and generalization ability of our method under challenging conditions. We first demonstrate that our approach is resilient to changes in mesh connectivity by testing on remeshed datasets. Then, we present representative failure cases to highlight the limitations of our pipeline in practical scenarios with incomplete or partially matching landmarks. To further evaluate the robustness of our method under varying mesh connectivity, we conduct experiments on the FAUST remeshed dataset [93], where the source and target meshes have same geometry but significantly different triangulations. As shown in fig. 4.11, our method consistently identifies semantically meaningful landmarks across differently meshed surfaces. This result highlights the sampling and resolution agnosticism of our pipeline, and demonstrates its reliability in real-world scenarios where mesh quality and connectivity may vary substantially.

Table 4.2 shows the failure rate of our method. A pair is considered a failure if its result is worse than that of ICP or ZoomOut with WKS initialization. Most failures occur in partial matching scenarios, where missing regions alter the geodesic structure.

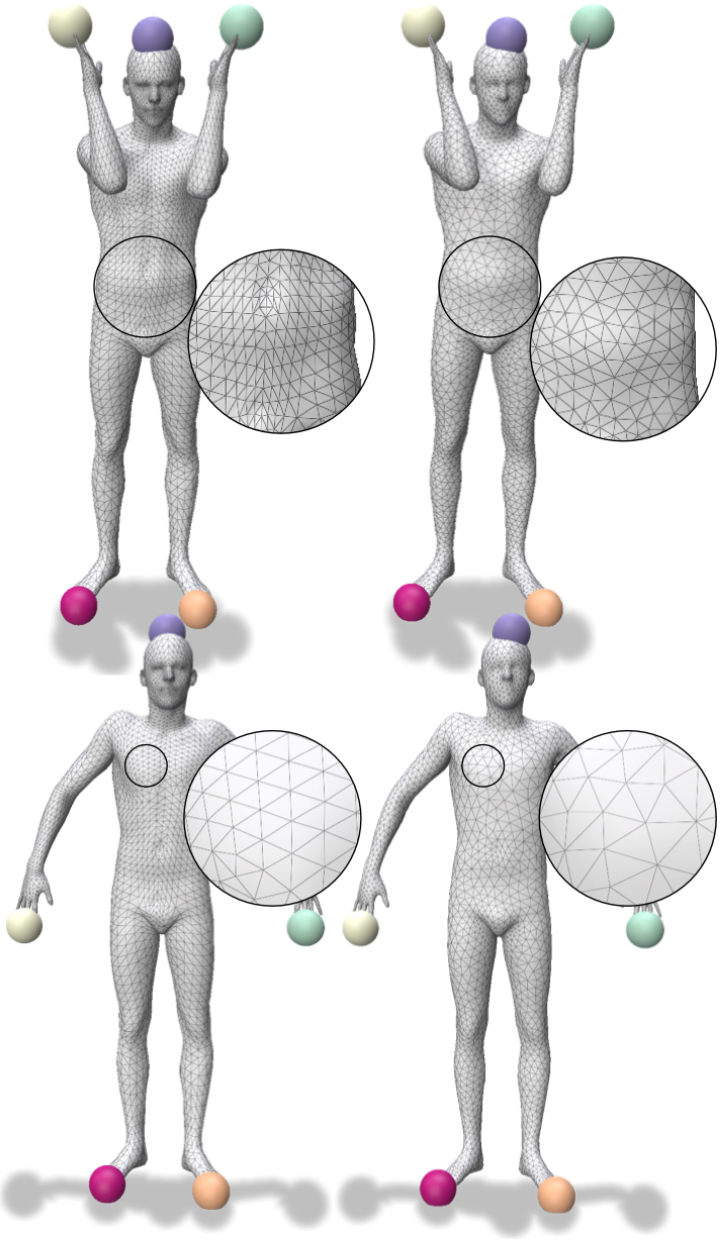


Figure 4.11: Our approach produces stable and consistent landmark placement across remeshed surfaces (right), demonstrating sampling and resolution agnosticism. This robustness enables reliable shape matching across a wide range of mesh representations, including those with significantly different vertex counts.

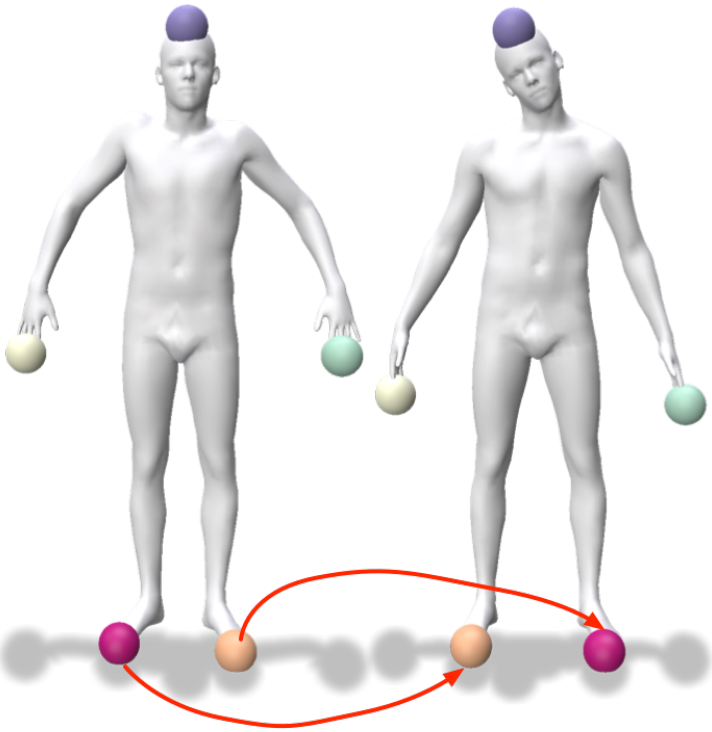


Figure 4.12: Failure case under partial matching. The intrinsic landmarks on the feet are mismatched, resulting in incorrect correspondences.

Table 4.2: Failure Rate. The table shows the percentage of shape pairs where our method performs worse than ICP or ZoomOut.

Dataset	Compare to ICP + ZoomOut (%)
Dynamic FAUST	8.75
TOSCA	13.19
SCAPE	7.97

Figure 4.12 illustrates a representative failure case where the left and right feet are incorrectly matched. Although the shapes are complete, the lack of strong extrinsic cues causes the intrinsic landmarks to become symmetrically ambiguous, matching the left foot to the right. This semantic confusion introduces localized distortion. While such ambiguity leads to a 10%–20% drop in accuracy, our method still performs comparably or better than baselines without intrinsic landmarks, though it does not match the best-case results achievable under unambiguous geometry.

Although our method achieves high-quality results, it has several limitations that we aim to address in future work. First, while intrinsic landmarks often improve correspondence, we cannot guarantee that their inclusion benefits all models. For example, in our experiments with fMap (+ILC), enforcing bijectivity near specific points occasionally degrades

performance elsewhere. Extending the current pipeline to provide localized guidance without introducing such side effects remains an open challenge. Second, our method is not directly applicable to partial shape correspondence, as topological differences significantly affect geodesic distances. Thirdly, our method is designed for shapes with approximately preserved intrinsic geometry, and is not intended for cross-category landmark discovery, where semantic correspondences are often ill-defined. In future work, we plan to extend our approach to handle partial inputs, enabling more flexible and robust shape matching across a broader range of scenarios.

4.6 Conclusion

We presented an unsupervised method for computing dense correspondences between non-rigid shapes via automatically generated intrinsic landmarks. Our approach computes a sequence of Karcher means and selects geodesically distributed points to construct an ordered set of landmarks, without requiring any user input or supervision. Built upon the functional map framework of Ovsjanikov et al. [78], our method integrates seamlessly into existing pipelines and improves correspondence accuracy with negligible additional cost (2–3 seconds). The core insight is that a small set of well-ordered intrinsic landmarks can effectively guide dense matching by leveraging only geometric information. Our method demonstrates strong compatibility with a range of existing shape correspondence algorithms, and achieves consistent improvements across benchmarks, highlighting its generalization ability across datasets and pipelines.

Chapter 5

Data-Free Learning for Mesh Repair

Mesh inpainting remains a critical challenge in geometry processing, particularly for real-world scans that often contain holes and missing regions. This paper introduces a data-free neural mesh inpainting framework that operates directly on incomplete input meshes, preserving geometric and topological consistency. Our approach involves stitching corresponding regions from the zero-level isosurface and refining vertex positions using a self-supervised graph convolutional network. Extensive experiments demonstrate that our method outperforms traditional dataset-independent approaches and exhibits greater robustness than existing deep learning models. The repaired meshes maintain sufficient integrity for downstream applications such as physical simulation and 3D printing.

5.1 Introduction

High-quality and geometrically consistent meshes are indispensable for applications such as 3D printing, physical simulation, and computer graphics animation. However, meshes reconstructed from real-world data are often incomplete, exhibiting holes and missing regions caused by occlusion, reflective surfaces, and limited scanning coverage. Therefore, a reliable and effective repair algorithm is essential to bridge the gap between real-world reconstruction and downstream applications.

Mesh inpainting is an ill-posed problem—the missing geometry cannot be determined from the available data. For the missing regions, neither the vertex positions nor their connectivity information is available. Traditional methods [122, 123, 124, 125, 126, 127] rely primarily on geometric priors such as smoothness and self-similarity. While these methods can

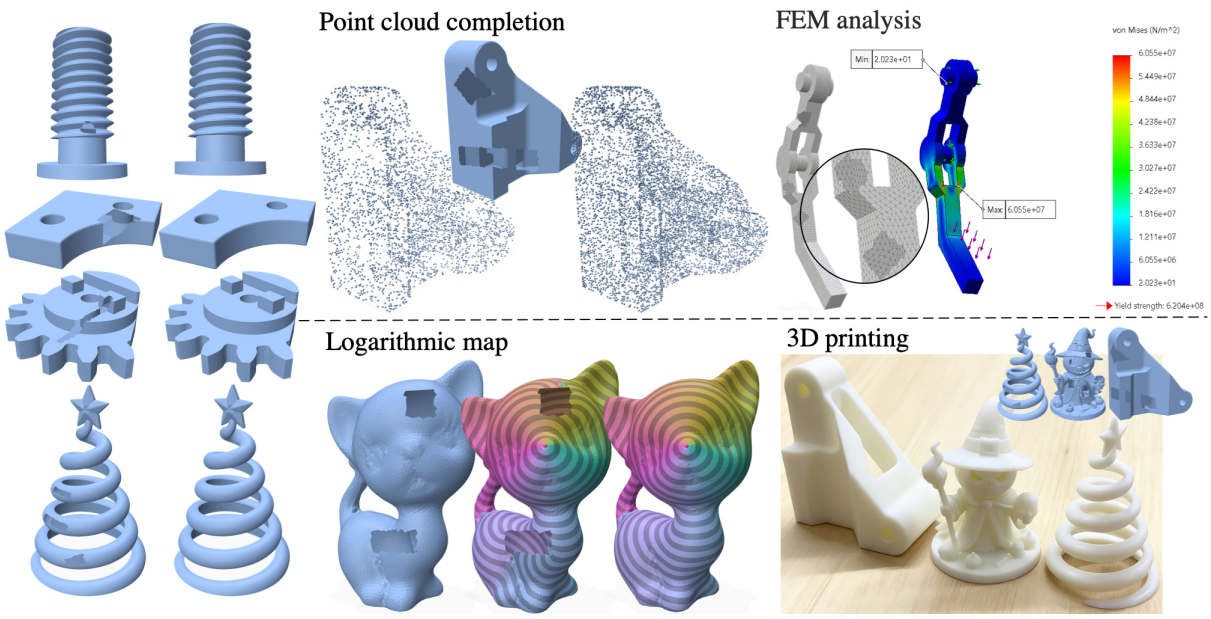


Figure 5.1: Our data-free neural mesh inpainting method restores missing regions directly on the input mesh while preserving geometric and topological consistency. The repaired meshes can be seamlessly applied to downstream tasks such as point cloud completion, FEM simulation, logarithmic map computation, and 3D printing.

effectively repair small or regularly shaped holes and preserve overall topology, they struggle to generalize to large or complex missing regions and often demand careful parameter tuning.

In contrast to purely geometry-based methods, recent works incorporate data-driven priors or apply supervised deep learning to improve completion quality. Some approaches developed for voxel grids [128] and point clouds [129, 130, 131, 132, 133, 134] ignore the intrinsic mesh connectivity. The primary challenge for point-cloud-based mesh repair lies in reliably reconstructing triangle connectivity. Even the latest generative models [135, 136, 137, 138, 139] typically produce meshes with only a few thousand faces, which limits their geometric fidelity. This constraint continues to hinder the advancement of mesh-based inpainting methods, even though large-scale 3D datasets are now available [140, 141]. More recent methods [142, 143, 144, 145] can produce high-quality meshes but rely on manually provided point clouds to guide the reconstruction of missing regions. Some self-prior-based methods do not require ground-truth inpainted geometries but still depend on large-scale shape datasets. These approaches [146, 147, 148] often alter the mesh format and change the topological properties of the input shape. These methods also perform poorly when encountering shapes that are not represented in the training datasets.

To address the above challenges, we present a Neural Mesh Inpainting (fig. 5.1) that per-

forms completion directly on the mesh domain. The geometric stage first projects the missing regions of the input mesh onto its zero-level isosurface to identify target areas for reconstruction. Based on this projection, a geodesic-driven triangulation algorithm enables cutting and stitching along arbitrary surface curves while maintaining valid topology. The process naturally supports parallel repair of multiple disconnected holes. Unlike previous approaches, our method eliminates the need for large-scale training datasets and offers a general, scalable solution applicable to arbitrary mesh geometries. In addition, the repaired meshes preserve the original structure, allowing seamless integration with existing neural architectures for fine-tuning.

Summary. In this paper, we present a data-free mesh inpainting method. Our approach is built around four key properties:

- Our method preserves the original mesh structure, modifying only the repaired regions while keeping the input geometry and topology completely intact.
- It scales efficiently to large and complex meshes, handling models with a high number of vertices and faces without compromising accuracy or stability.
- The approach operates entirely on a single input mesh without requiring any external training data.
- The repaired meshes can be directly applied to downstream tasks such as physical simulation and 3D printing, demonstrating its suitability for real-world use.

5.2 Related Work

5.2.1 Traditional Geometry-Based Shape Repair

Traditional mesh inpainting methods can be broadly classified into surface-based and volume-based approaches. Surface-based methods reconstruct missing regions by generating new triangles guided by local geometric cues such as vertex density, triangle area, or dihedral angles [122, 149, 127, 123, 124]. They typically apply remeshing or smoothing to improve surface quality and perform well on small, simple holes, but often fail on large or topologically complex missing regions. To address these limitations, context-based methods [125, 126] enhance self-similarity by synthesizing features from similar patches, while Kraevoy and Sheffer

[150] introduced a template-based approach that maps incomplete meshes to reference models. Although these extensions can better preserve geometric detail, they still depend on the availability of similar shapes, complete templates, and manual user interactions.

Volume-based methods first convert an input mesh into a discrete voxel representation and then perform hole filling within the volumetric domain. A common strategy is volumetric diffusion, which generates a complete voxelized volume from partial geometry [151, 152, 153]. Bischoff et al. [154] further proposed morphological dilation and erosion on an octree structure to efficiently fill holes in the voxel space. After completion, the surface mesh is reconstructed by extracting triangles from the isosurface. These approaches are advantageous for repairing complex surfaces and producing manifold meshes. However, they often cause unwanted deformation of nearby regions during diffusion or morphing, and their output resolution is constrained by the computational inefficiency of voxel-based representations.

5.2.2 Data-Driven Shape Inpainting

Data-driven shape inpainting methods used volumetric networks for shape recovery from partial scans [128, 155], while later works [129, 156] adopted multi-resolution prediction and coarse-to-fine refinement to improve reconstruction quality.

Point cloud completion has been widely investigated since point clouds are direct outputs of 3D scanners. Networks such as PCN [130] and SA-Net [131] reconstruct complete point clouds via encoder–decoder architectures, while GAN-based [157] and reinforcement learning approaches [158] further improve the realism of generated shapes. PMP-Net [159] infers point trajectories to restore missing structures without explicit point generation. Recently, implicit neural representations have gained popularity for shape inpainting due to their memory efficiency and smooth surface reconstruction. Methods based on signed distance functions (SDF) [148, 160, 161] and vector-quantized implicit functions (VQDIF) [145, 162] have demonstrated strong shape completion capability.

Mittal et al. [147] trained a network to predict occluded regions in point clouds by simulating missing points from range scans, while Chu et al. [165] completed polygonal meshes through a truncated signed distance field (TSDF) representation. Although these methods eliminate the need for complete ground-truth shapes, they still depend on large-scale datasets such as ShapeNet [166], KITTI [167], and Thingi10k [141]. Moreover, converting meshes to

Table 5.1: Method comparison. The Data-Free column indicates whether the method requires training datasets. Input Fidelity measures whether the original mesh structure is preserved. Scalability refers to the ability to handle large, high-resolution meshes. The Performance column reflects the overall reconstruction quality. Comments denote additional requirements such as real data or auxiliary tools.

	Data-Free	Type	Scalability	Input Fidelity	Performance
PMP-Net [159]	✗	Point	✗	✗	Medium
IF-Net [148]	✗	Point	✗	✗	Medium
ShapeFormer [145]	✗	Point	✗	✗	Low
SapceMesh [142]	✗	Mesh	✗	✗	High
SeMIGCN [163]	✓	Mesh	✓	✓	Medium
MeshFix [164]	✓	Mesh	✓	✓	Low
Ours	✓	Mesh	✓	✓	High

point-based or volumetric forms often alters their topological properties, leading to inconsistencies in reconstructed geometry. When meshes are first converted into point clouds and later re-meshed, the generated surfaces are typically low-resolution and fail to preserve the original connectivity structure. A comparison summary can be found in Table 5.1. In contrast, our method operates directly on the original incomplete mesh without altering its structure. It preserves the topology and geometric continuity of the input while removing the dependency on external datasets, offering a more reliable and data-free solution for mesh inpainting.

5.2.3 Self-Prior Learning in Geometry Processing

Self-prior learning has emerged as a data-driven yet data-independent paradigm in geometry processing. The Deep Geometric Prior (DGP) [168] learns local surface representations from a single point cloud by training separate multilayer perceptrons (MLPs) for each region. Point2Mesh [143] extends this idea to the mesh domain through MeshCNN [37], which employs shared-weight convolutions and dynamic mesh pooling, and was later adapted for textured mesh reconstruction from colored point clouds [169]. For sparse or irregularly sampled point clouds, recent works have adopted unsupervised reconstruction techniques based on SDF [170, 171] and point consolidation strategies [172] to improve geometric consistency. The approach in [163] relies on MeshFix [164] for initial completion, and the final reconstruction quality is therefore largely determined by the preprocessing result rather than the learned model.

Despite their progress, existing self-prior methods remain restricted to point-based or

volumetric representations and often reconstruct meshes from low-resolution surfaces, causing the loss of fine topology and structural connectivity. In contrast, we propose a data-free self-prior framework that operates directly on the input mesh, preserving its original geometry and topology throughout the inpainting process.

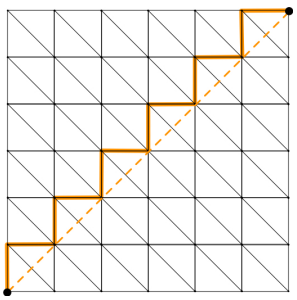
5.3 Method

In the following, we outline the main components of Neural MeshImplant: a geometric stage that provides the tools for mesh clipping, stitching, and implanting, followed by a neural stage that fine-tunes the surface.

5.3.1 Geometric Preprocessing

Our domain is a triangle mesh $M = (V, E, F)$ with arbitrary topology. For an incomplete mesh, we first perform a preprocessing to obtain a watertight surface. The zero-level isosurface generated from the GSD [173] provides a contour (with contouring via marching cubes [174].) on which hole boundaries can be mapped, as shown in fig. 5.2. However, this contour cannot be used directly, as it lacks geometric detail, is often non-manifold, and suffers from poor triangulation quality. Therefore, we refine it further and use it as an intermediate scaffold for subsequent processing.

To improve the quality of the initial isosurface mesh, we first apply a light QSLIM simplification [175] to remove spurious triangles. The remaining vertices are then processed with Poisson surface reconstruction [176] (octree depth of eight) to obtain a cleaner manifold surface. Finally, a QSLIM simplification followed by triangular remeshing [177, 178] ensures a vertex density comparable to the input mesh. The overall process, illustrated in fig. 5.3, produces the target surface for boundary mapping and patch cutting.



A simple way to generate cutting paths is to use traditional shortest path algorithms [179], but such methods are constrained by mesh edges and often produce jagged paths (see inset). We instead use geodesic paths computed on the continuous surface, which yield smooth and geometry-consistent cuts.

A geodesic is the shortest path between two points con-

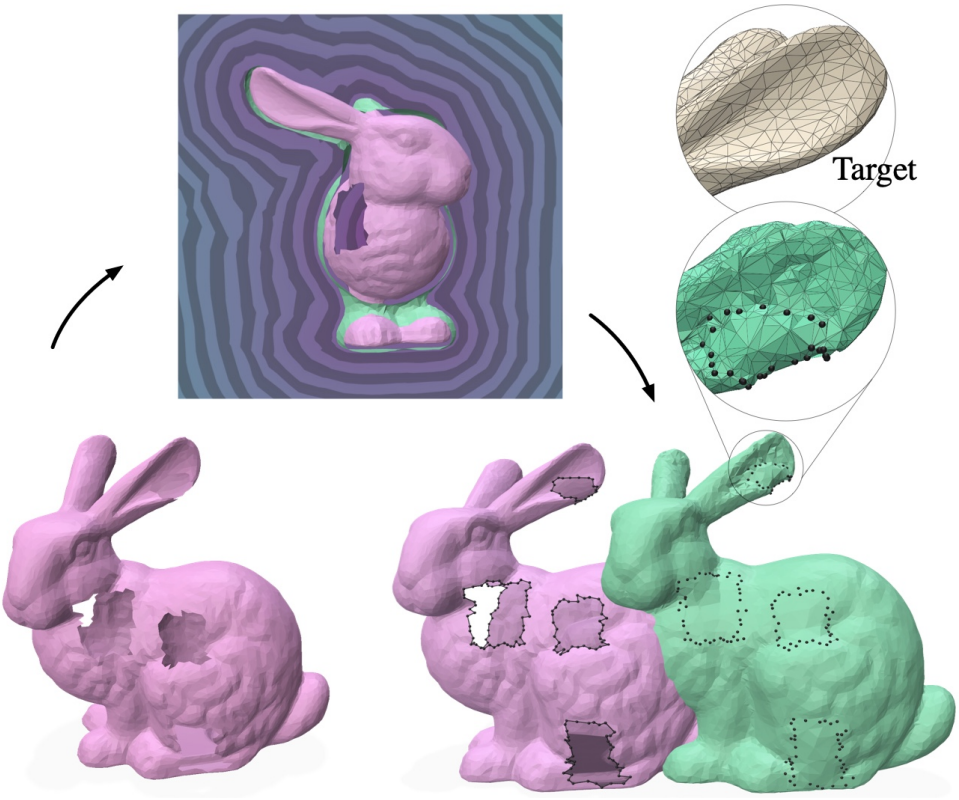


Figure 5.2: A zero-level isosurface is extracted from the GSD (top), onto which the original hole boundary points are mapped. However, the low-quality mesh must be refined before mapping.

strained to lie on the surface; when the triangles along the path are unfolded, it appears as a straight line [180] (see inset). We compute geodesics [181, 182] sequentially between consecutive boundary points and record the triangles they traverse. Projection points that fall inside a triangle are connected to its three vertices, splitting the face into three smaller ones. If two projection points lie in the same triangle, the subdivision is applied sequentially. For each triangle fully crossed by a geodesic, we select the crossing point on the intersected edge and connect it to the opposite vertex (fig. 5.4). We leverage a half-edge [183] mesh representation to ensure edge-manifoldness and maintain consistent orientation. With these operations, the geodesic path is entirely represented by edges of the processed mesh. For holes that are topologically independent (i.e., sharing no common faces), our cutting algorithm can process them in parallel, further improving efficiency.

Each projected point x is parameterized using barycentric coordinates of triangle $ijk \in F$ with vertex positions v_i, v_j, v_k, as $x = t_i v_i + t_j v_j + t_k v_k$. We snap a projected point to the nearest edge if one barycentric weight is small, or to the corresponding vertex if two weights are small. This operation improves numerical stability and avoids low-quality triangles in the

subsequent retriangulation step.

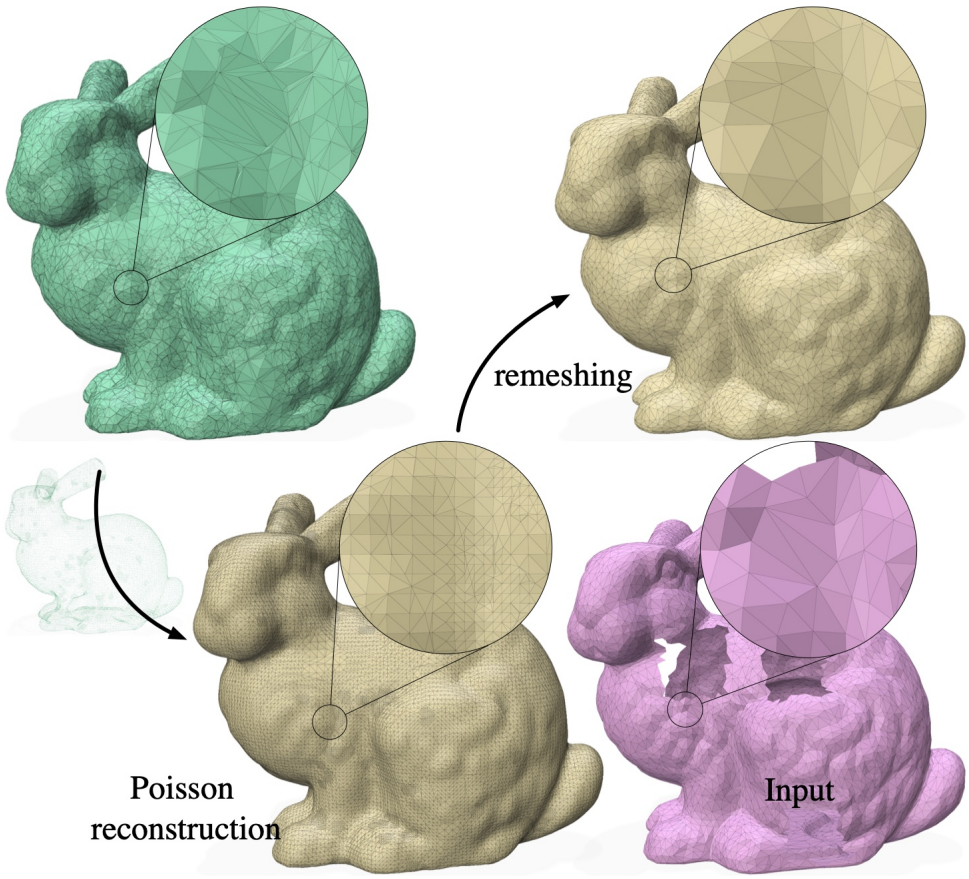


Figure 5.3: Poisson surface reconstruction is applied to the initial isosurface to improve mesh quality, followed by remeshing to ensure a uniform vertex density comparable to the input mesh.

During the stitching process, we update the projected boundary vertices of each patch to their corresponding positions on the input mesh hole. For new vertices generated between two projected points, we insert them proportionally along the corresponding original edges, forming watertight connections between the patch and the input mesh.

5.3.2 Self-Supervision

Inspired by the mesh autoencoder architecture [184, 185, 186], we adopt it to learn feature representations of the input mesh using a graph convolution operator that avoids the intricate design of kernel functions. The learned features are then used to fine-tune the repaired regions for improved geometric consistency.

Notation. We denote a mesh as $\mathcal{M}$ with vertex and face sets $\mathcal{V}$ and $\mathcal{F}$. Vertex positions and face normals are represented as $\mathbf{X} \in \mathbb{R}^{|\mathcal{V}|\times 3}$ and $\mathbf{N} \in \mathbb{R}^{|\mathcal{F}|\times 3}$, respectively. The input hole

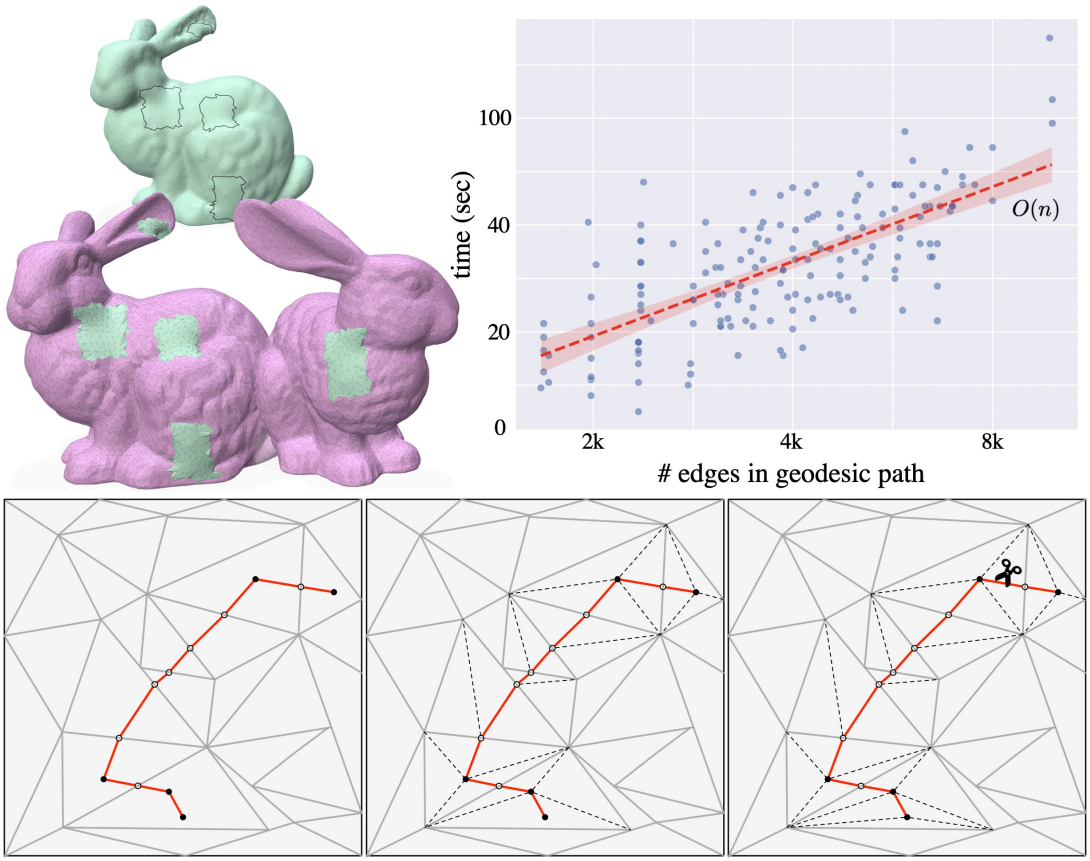


Figure 5.4: Top: path stitching between the projection and input meshes. Bottom: retriangulating the faces intersected by the geodesic yields a valid cutting path (red) represented by mesh edges. The measured runtime (top-right) confirms a linear $O(n)$ relationship between path length and computation time. In this example, the bunny model contains 14K faces, with 400 edges along the geodesic path, and the entire cutting process takes 2.1 seconds.

mesh is denoted as $\mathcal{M}_{\text{in}}$, and the watertight mesh obtained after preprocessing (section 5.3.1) is denoted as $\mathcal{M}_{\text{init}}$. The network outputs a completed mesh $\mathcal{M}_{\text{cmp}}$ with refined vertex positions at the target (hole-filled) regions. We define masks m_{v}^i and m_{n}^j to distinguish vertices and face normals in hole regions, where $m = 0$ for holes (real or fake) and $m = 1$ otherwise.

Pooling/Unpooling. The model employs a mesh pooling operation based on QSLIM [175]. During pooling, vertex pairs are iteratively contracted to simplify the mesh while preserving geometric fidelity through quadric error metrics. The output features are obtained by multiplying the input features with a precomputed downsampling transformation matrix. The corresponding unpooling matrix is also stored for feature recovery during decoding [20].

Model Architecture. fig. 5.5 illustrates self-supervised framework for mesh inpainting. Instead of directly predicting vertex positions, the Graph Convolutional Networks (GCN) estimates displacements from a Laplacian-smoothed mesh. The GCN takes per-vertex features

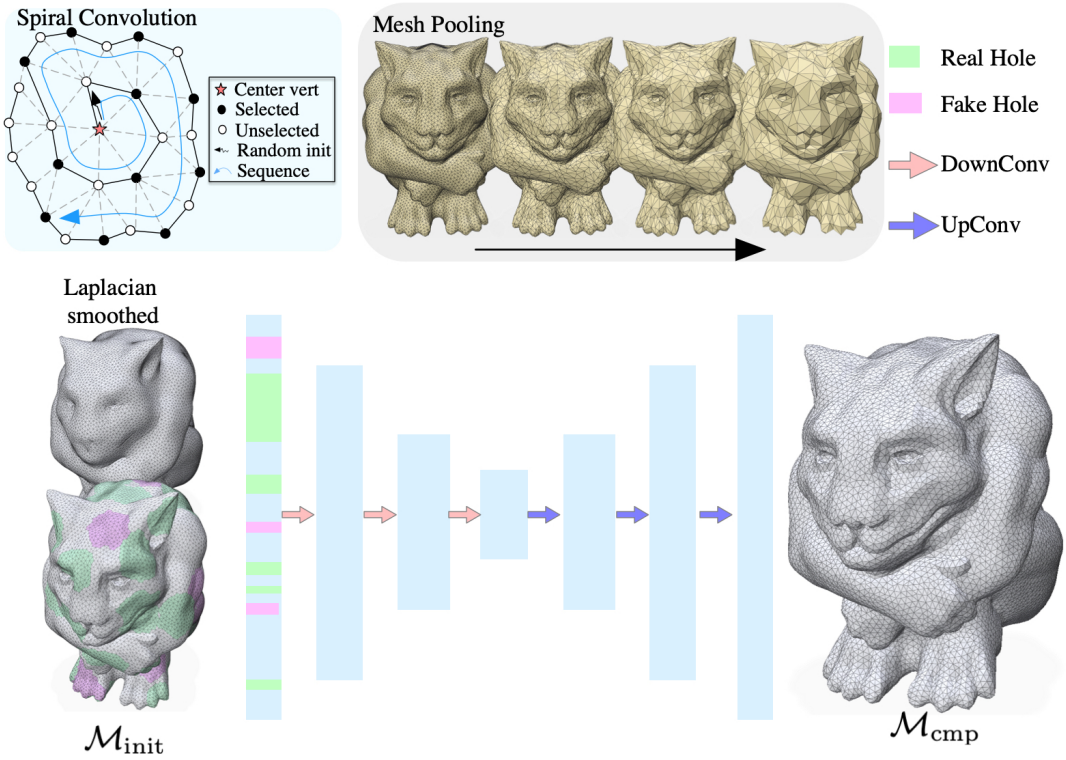


Figure 5.5: Self-supervised training of our our refinement model. Pooling and unpooling are implemented via QSLIM-based precomputed transformation matrices, consistent with prior mesh autoencoder designs [20]. The model predicts vertex displacements to reconstruct the inpainted surface from an oversmoothed mesh using both real (green) and fake (pink) holes.

$(\Delta x^i_{in}, \Delta y^i_{in}, \Delta z^i_{in}, m^i_{\text{v}})$ as input, where $m^i_{\text{v}} \in \{0, 1\}$ indicates whether the vertex belongs to a real or fake hole. It predicts the 3D displacement $(\Delta x^i_{out}, \Delta y^i_{out}, \Delta z^i_{out})$ to recover the inpainted surface.

Loss Functions. The vertex position loss E_{p} is computed as the root mean squared error (RMSE) between the predicted and initial hole-filled mesh vertices, excluding real hole regions using the vertex mask m^i_{v}.

$$E_{\text{p}} = \left(\frac{1}{\sum_i m^i_{\text{v}}} \sum_{i=1}^{|\mathcal{V}|} m^i_{\text{v}} \left\| \mathbf{x}^i_{\text{cmp}} - \mathbf{x}^i_{\text{init}} \right\|_2^2 \right)^{\frac{1}{2}}. \tag{5.1}$$

The facet normal loss E_{n} encourages local geometric fidelity by aligning the predicted and initial face normals, preserving sharp features. It is computed as a mean absolute difference over non-hole faces using the mask m^j_{n}, and evaluated only at the finest resolution to reduce computational cost:

$$E_{\text{n}} = \frac{1}{\sum_j m^j_{\text{n}}} \sum_{j=1}^{|\mathcal{F}|} m^j_{\text{n}} \left\| \mathbf{n}^j_{\text{cmp}} - \mathbf{n}^j_{\text{init}} \right\|_1. \tag{5.2}$$

The smoothness term E_{r} regularizes facet normals to suppress noise and preserve sharp geometric features.

$$E_{\mathrm{r}} = \frac{1}{|\mathcal{F}|} \sum_{j=1}^{|\mathcal{F}|} \left\| \mathbf{n}_{\mathrm{cmp}}^{j} - S_{\mathrm{bnf}}^{(t)}(\mathbf{n}_{\mathrm{cmp}}^{j}) \right\|_1 . \tag{5.3}$$

Here, $S_{\mathrm{bnf}}^{(t)}$ denotes t iterations ($t = 5$ in this paper) of bilateral normal filtering (BNF) [187], which penalizes the deviation between predicted normals and their filtered counterparts.

Total loss. The overall training objective combines vertex, normal, and smoothness terms:

$$E = \sum \lambda_{\mathrm{p}} E_{\mathrm{p}} + \lambda_{\mathrm{n}} E_{\mathrm{n}} + \lambda_{\mathrm{r}} E_{\mathrm{r}}. \tag{5.4}$$

Each coefficient λ controls the relative contribution of the corresponding loss term. We use three down-pooling layers with $\lambda_{\mathrm{p}} = 0.25$, and both λ_{n} and λ_{r} set to 0.5. Each simplification step reduces the number of vertices to 30% of the previous level.

5.4 Results and Applications

We evaluate our method across a wide range of experiments. As shown in Figure 5.4, even for meshes with hundreds of thousands of triangles, our model processes geodesic paths spanning thousands of edges within tens of seconds, and only a few seconds in typical cases.

5.4.1 Accuracy and Performance

We first compare our method with three baseline approaches: the Advancing Front Method (ADVF) [124], MeshFix (MFIX) [164], and Context-based Coherent Surface Completion (CCSC) [126]. All of these methods are data-free and operate directly on an input mesh. For intuitive visualization, we colorize the reconstructed meshes according to the distance between the output and the ground-truth surfaces.

Figure 5.6 presents representative qualitative results, while more comprehensive quantitative comparisons are provided in table 5.2. For visualization purposes, the reconstruction errors are normalized and color-mapped onto the surface. As shown in this figure, traditional methods such as ADVF [124] and MFIX [164] tend to fill holes with overly planar patches due to their reliance on local geometric heuristics. CCSC [126], while capable of producing smoother results, requires extensive optimization and takes over 500 seconds even for mod-

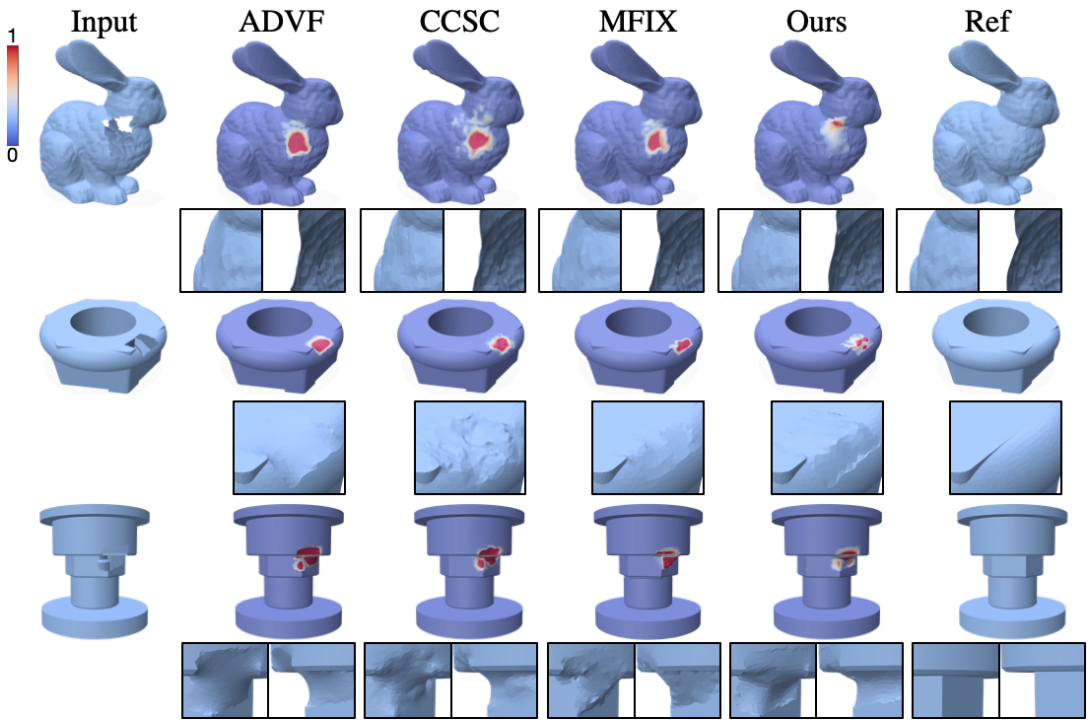


Figure 5.6: Inpainting results on meshes with holes. The reconstructed meshes are colorized according to the distance to the ground-truth surfaces. Traditional geometry-based methods tend to produce noticeable bulges or inward dents near the repaired regions, while our method yields smoother and more accurate reconstructions.

erately sized meshes. In contrast, our method reconstructs the missing regions in just a few seconds, generating geometrically consistent surfaces that preserve the original curvature and fine-scale details.

Table 5.2: Quantitative comparison of inpainting performance using the average error distance in the repaired regions (in units of 10^{-3}).

	ADVF [124]	CCSC [126]	MFIX [164]	Ours
Bunny	6.01	5.31	5.55	**4.33**
Fandisk	5.69	4.94	2.78	**0.54**
Bimba	3.52	4.46	2.83	**1.66**
Bust	1.72	2.19	1.29	**0.74**
Igea	2.96	1.40	2.27	**1.13**
Draon	7.3	6.04	3.99	**1.89**
Nut	3.96	4.75	4.69	**1.02**

5.4.2 Comparison with Supervised and Self-Prior Models

We further compare our approach with several recent data-driven models, including IF-Net [148], ShapeFormer [145], SeMIGCN [163], and SpaceMesh [142].

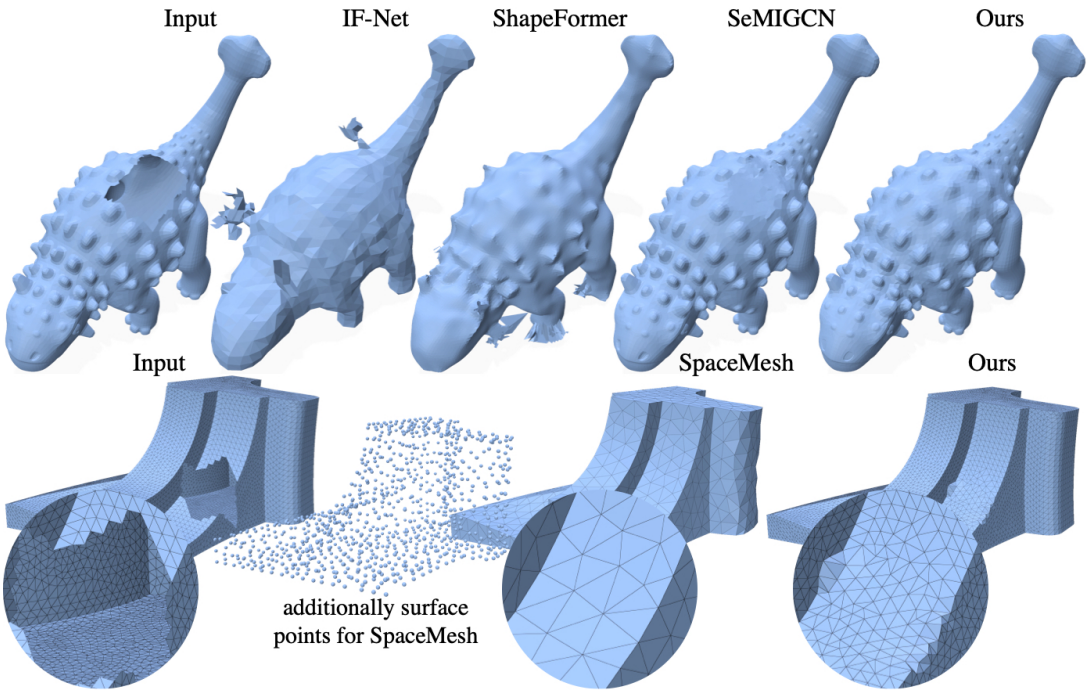


Figure 5.7: Comparison with representative data-driven methods. While IF-Net and ShapeFormer produce coarse reconstructions and SeMIGCN failed to inpaint the shape geometries, SpaceMesh additionally requires surface point inputs. Our method achieves clean, detailed, and watertight repairs directly from the incomplete mesh.

Figure 5.7 presents the results of these data-driven methods compared with our approach. For this comparison, we used the official pretrained weights and implementations provided by the respective authors of IF-Net and ShapeFormer.

For the transformer-based SpaceMesh [142], the model can only process meshes with a limited number of faces due to the inefficiency of its face tokenization algorithm. Moreover, it requires collaboration with other models or additional surface points around missing regions as conditions for mesh generation. SpaceMesh also alters the connectivity of the input mesh, resulting in structural inconsistencies.

More fundamentally, transformer-based architectures [135, 136, 137, 138] face intrinsic scalability limitations: constructing or processing meshes with tens of millions of faces is currently computationally infeasible due to the quadratic complexity of token attention and memory constraints.

All these data-driven methods fail to inpaint the missing geometries when the test meshes are not included in their training datasets. In contrast, our method is entirely independent of training data, operating robustly across arbitrary mesh inputs. SpaceMesh achieves nearly

zero error because it requires the point cloud of the missing regions as input; thus, it only reconstructs connecting surfaces and is not directly comparable.

5.4.3 Applications

Our method enables several downstream applications that benefit from reliable and topology-preserving mesh repair. These applications highlight the practicality and robustness of our approach for real-world geometric modeling and manufacturing workflows. We next illustrate several representative applications enabled by our repaired meshes.

Point Cloud. Our framework naturally serves as a bridge for point cloud completion. Given an incomplete point cloud, we first reconstruct a surface mesh, repair the missing regions using our mesh inpainting method, and then resample the completed mesh to obtain a dense, completed point cloud. Unlike data-driven completion methods, our approach requires neither large-scale training datasets nor costly model training, making it an efficient and data-free alternative for point cloud restoration.

Figure 5.8 shows a comparison with data-driven methods. PMP-Net [159] fails to recover missing regions for unseen shapes in its training data. Our data-free method, on the other hand, can inpaint arbitrary geometries while preserving natural shape continuity.

Simulation on Repaired Meshes. Our method preserves the original geometric and topological structure of the input mesh, making it directly compatible with downstream physical simulations. For example, in solving partial differential equations (PDEs) such as heat diffusion or deformation analysis on the surface, the repaired geometry can seamlessly inherit the original boundary conditions defined along the hole edges. Since our inpainting process maintains vertex connectivity and mesh consistency, simulation variables can be propagated continuously over the repaired regions without reinitializing the solver or remeshing the entire domain. This property enables efficient post-repair analysis and ensures that numerical stability and boundary fidelity are preserved.

To verify compatibility with downstream solvers, we perform surface PDE computations on the inpainted meshes. As shown in fig. 5.9, both the logarithmic map and parallel transport [188] of tangent vectors can be seamlessly extended from the hole boundaries to the repaired regions without reinitialization. Since our method preserves the original connectiv-

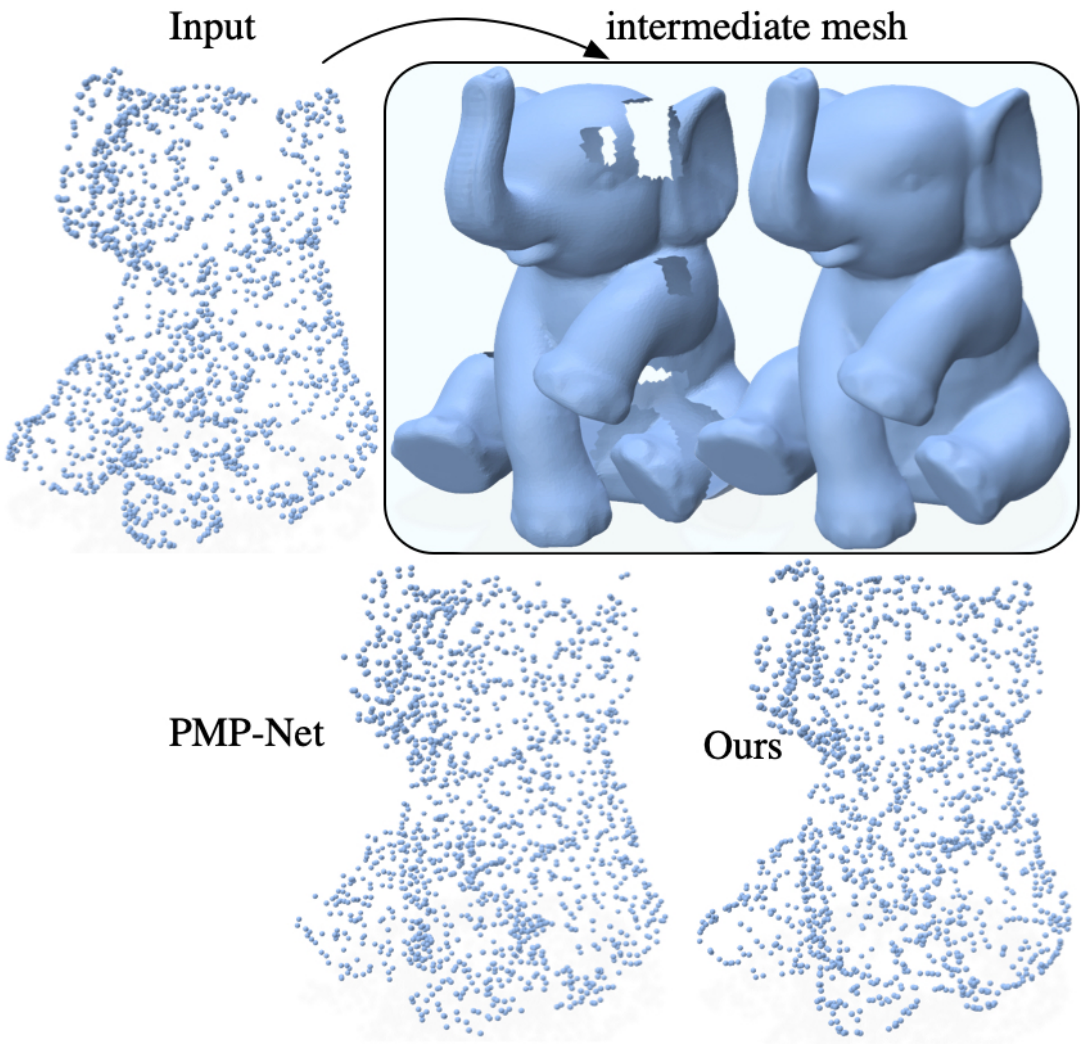


Figure 5.8: Point cloud completion results. Our method reconstructs an intermediate mesh from the incomplete input, repairs it, and resamples a consistent point cloud.

ity and vertex indexing, the solver can reuse precomputed boundary conditions, enabling a continuous simulation workflow. On a mesh with 26K vertices and 52K faces, solving PDEs on the repaired surface requires only 5 ms, whereas recomputing from scratch takes 60 ms, highlighting the efficiency and physical consistency of our topology-preserving repair.

Additive Manufacturing Applications. In 3D printing, the repaired meshes generated by our method exhibit sufficient structural integrity and watertightness to be fabricated directly. Because the surface continuity are preserved during inpainting, the repaired models can be exported as valid models without additional post-processing or remeshing.

Figure 5.10 demonstrates the use of the proposed mesh inpainting method for additive manufacturing. The blue models are the original meshes with missing regions, and the white models are the repaired and printed results. The repaired meshes maintain watertightness and

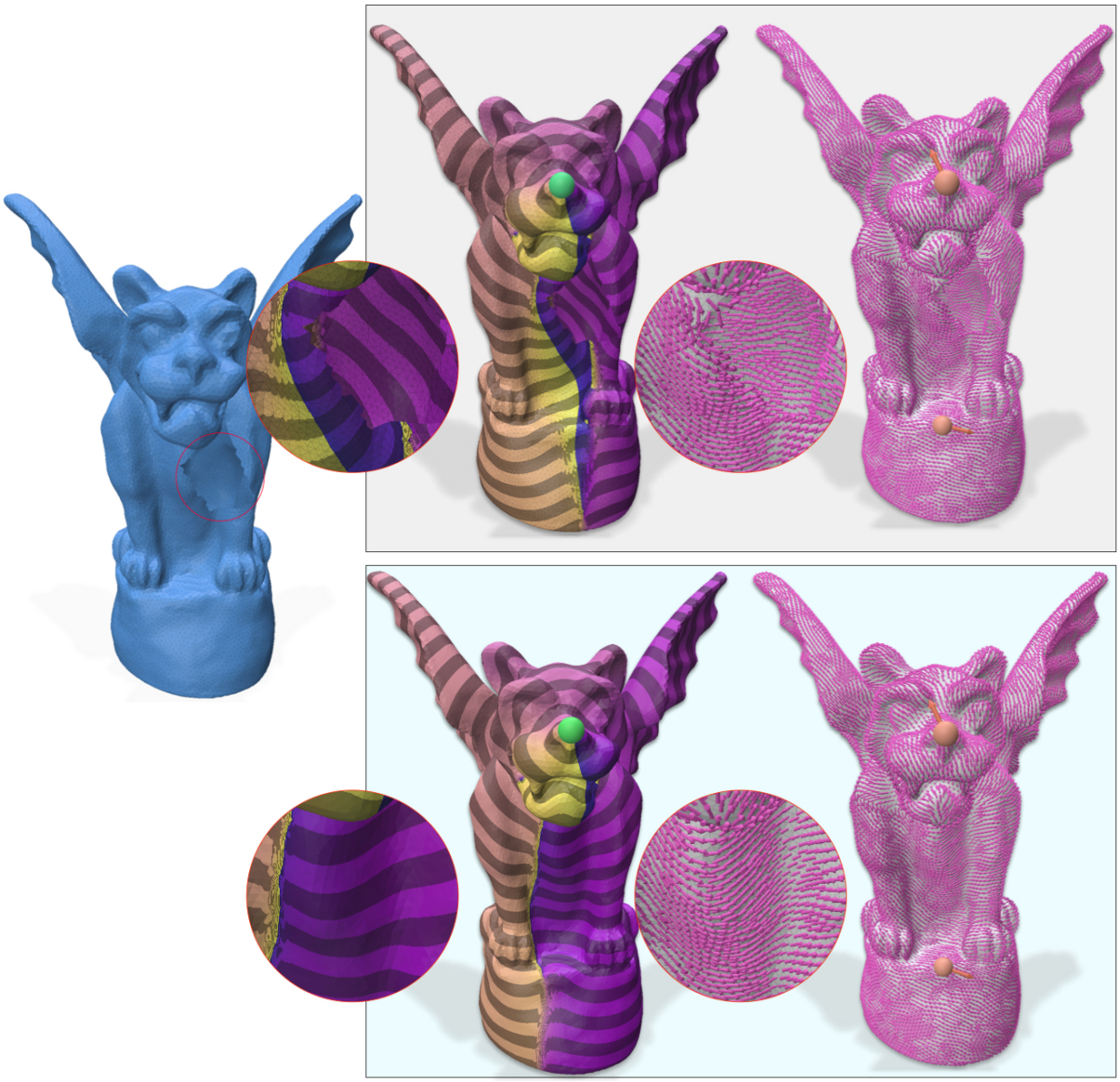


Figure 5.9: Surface PDE continuation on repaired meshes. Top: the input mesh with holes. Bottom: the repaired mesh. Logarithmic map and parallel transport are computed directly on the repaired surface by reusing boundary conditions along the hole edges.

structural stability, allowing direct fabrication without further processing.

5.5 Limitations & Future Work

We introduced a data-free neural mesh inpainting framework that reconstructs missing regions directly on the mesh domain while preserving the original topology and geometric consistency. Our approach integrates a topology-preserving geodesic cut-and-stitch algorithm for hole filling with a self-supervised graph convolutional network that refines vertex positions without external data. Through extensive evaluations, we demonstrated that



Figure 5.10: Repaired meshes from our method are watertight and structurally sound, allowing direct 3D printing without additional post-processing or remeshing.

our method achieves high reconstruction accuracy, strong robustness across diverse geometries, and seamless compatibility with downstream applications such as point cloud completion, physical simulation, and 3D printing. Future work includes extending the cut-and-stitch pipeline to dynamic meshes and integrating physical constraints for simulation-aware repair.

Chapter 6

Conclusion

Geometry processing provides a rich set of tools for analyzing and manipulating 3D surfaces. The intrinsic geometric features explored in this thesis highlight new opportunities to improve geometric learning methods. Incorporating intrinsic surface descriptors enables learning frameworks to represent geometric structure beyond vertex positions, using features such as geodesic distances, curvature, and spectral descriptors.

To manipulate intrinsic triangulations in practice, we extract features from the underlying surface geometry. We begin with Mesh Autoencoder (Chapter 2), where vertex normals support an efficient pooling operator for 3D morphable models, enabling neural networks to operate at multiple resolutions while preserving geometric structure. Furthermore, randomly generated bijective coarse-to-fine mesh counterparts (Chapter 3) allow the network to learn complex mesh shapes, improving the accuracy of reconstructed vertex positions while maintaining the desired topology. Karcher means are used as intrinsic landmarks on near-isometric surfaces (Chapter 4), enabling efficient and robust point-wise correspondence. These triangulations also support a data-free neural mesh inpainting framework (Chapter 5) that operates directly on incomplete meshes while preserving geometric and topological consistency.

Across these works, a broader principle emerges: learning does not replace geometry, but becomes more effective when built upon it. Geometric structure provides invariance, consistency, and physical meaning that are difficult to recover from data alone, while learning provides the flexibility to model complex variations beyond hand-designed geometric algorithms. The results of this dissertation show that these two perspectives are not competing alternatives, but complementary tools for geometric processing. Their integration provides a path

toward methods that are more accurate, robust, and less dependent on large-scale supervision.

Looking forward, an important direction for geometric learning is supporting industrial-scale realistic rendering and efficient production pipelines. Large language models may also enable more interactive workflows for 3D modeling, allowing users to specify geometric structures through high-level instructions. At the same time, scaling learning architectures to mesh-based representations remains a significant challenge, particularly for attention-based models when realistic scenes may contain millions of elements. Progress in these directions would allow many geometry processing algorithms to play a larger role in computer graphics, animation production, and photorealistic film generation. Realizing this vision will require advances in both theory and practice. With sustained effort and creativity, geometric learning will remain a vibrant area of research for years to come.

6.1 Future Directions

The methods presented in this dissertation address several challenges in mesh representation, correspondence, and geometry repair, while also suggesting directions for future research. As geometric data grows in scale and diversity, learning systems that reason about geometric structure remain an important challenge.

Geometry-Aware Foundation Models. Foundation models learn general representations from large and diverse datasets. Extending this paradigm to geometric data may enable reusable representations of shape structure that transfer across tasks such as correspondence, reconstruction, segmentation, and geometric editing. Key challenges include representing geometry across modalities and incorporating intrinsic geometric structure into large-scale learning architectures.

Scalable Geometric Learning. Scaling geometric learning methods to large and complex shapes remains a central challenge. Transformer-based architectures perform well in many domains, but their quadratic attention complexity limits their use on high-resolution meshes or large geometric datasets. Moreover, geometric data exhibits irregular connectivity and strong multi-scale structure that are not naturally aligned with standard transformer architectures. Developing scalable architectures for large geometric models therefore remains an important research direction.

Physics-Aware Geometric Learning. Incorporating physical priors—such as elasticity, material properties, and deformation constraints—may allow learning systems to better capture the behavior of real-world shapes. Integrating geometry processing with physics-based modeling remains a promising direction for geometric learning.

Geometry Agents. Agent-based systems combine learning-based reasoning with external computational tools. In geometry processing, such agents could plan and execute sequences of geometric operations rather than directly predicting outputs. A geometry agent may coordinate existing algorithms as modular tools, enabling more flexible and interactive workflows for geometric modeling.

Bibliography

[1] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, p. 436, 2015.

[2] B. Amberg, R. Knothe, and T. Vetter, "Expression invariant 3d face recognition with a morphable model," in *2008 8th IEEE International Conference on Automatic Face & Gesture Recognition*. IEEE, 2008, pp. 1–6.

[3] T. Li, T. Bolkart, M. J. Black, H. Li, and J. Romero, "Learning a model of facial shape and expression from 4d scans." *ACM Trans. Graph.*, vol. 36, no. 6, pp. 194–1, 2017.

[4] A. Tewari, M. Zollhofer, H. Kim, P. Garrido, F. Bernard, P. Perez, and C. Theobalt, "Mofa: Model-based deep convolutional face autoencoder for unsupervised monocular reconstruction," in *Proceedings of the IEEE International Conference on Computer Vision Workshops*, 2017, pp. 1274–1283.

[5] F. Yang, J. Wang, E. Shechtman, L. Bourdev, and D. Metaxas, "Expression flow for 3d-aware face component transfer," in *ACM SIGGRAPH 2011 papers*, 2011, pp. 1–10.

[6] J. Thies, M. Zollhöfer, M. Nießner, L. Valgaerts, M. Stamminger, and C. Theobalt, "Real-time expression transfer for facial reenactment," *ACM Trans. Graph.*, vol. 34, no. 6, nov 2015. [Online]. Available: https://doi.org/10.1145/2816795.2818056

[7] S. Bouaziz, Y. Wang, and M. Pauly, "Online modeling for realtime facial animation," *ACM Transactions on Graphics (ToG)*, vol. 32, no. 4, pp. 1–10, 2013.

[8] H. Li, T. Weise, and M. Pauly, "Example-based facial rigging," *Acm transactions on graphics (tog)*, vol. 29, no. 4, pp. 1–6, 2010.

[9] V. Blanz and T. Vetter, "A morphable model for the synthesis of 3d faces," in *Proceedings*

of the 26th annual conference on Computer graphics and interactive techniques, 1999, pp. 187–194.

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

[11] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf, "Deepface: Closing the gap to human-level performance in face verification," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 1701–1708.

[12] Y. Li, R. Bu, M. Sun, W. Wu, X. Di, and B. Chen, "Pointcnn: Convolution on x-transformed points," *Advances in neural information processing systems*, vol. 31, pp. 820–830, 2018.

[13] Y. Zhou, C. Wu, Z. Li, C. Cao, Y. Ye, J. Saragih, H. Li, and Y. Sheikh, "Fully convolutional mesh autoencoder using efficient spatially varying kernels," *Advances in Neural Information Processing Systems*, vol. 33, pp. 9251–9262, 2020.

[14] T. Bagautdinov, C. Wu, J. Saragih, P. Fua, and Y. Sheikh, "Modeling facial geometry using compositional vaes," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 3877–3886.

[15] A. Sinha, J. Bai, and K. Ramani, "Deep learning 3d shape surfaces using geometry images," in *European conference on computer vision*. Springer, 2016, pp. 223–240.

[16] A. Sinha, A. Unmesh, Q. Huang, and K. Ramani, "Surfnet: Generating 3d shape surfaces using deep residual networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 6040–6049.

[17] H. Maron, M. Galun, N. Aigerman, M. Trope, N. Dym, E. Yumer, V. G. Kim, and Y. Lipman, "Convolutional neural networks on surfaces via seamless toric covers." *ACM Trans. Graph.*, vol. 36, no. 4, pp. 71–1, 2017.

[18] S. Ji, W. Xu, M. Yang, and K. Yu, "3d convolutional neural networks for human action recognition," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 1, pp. 221–231, 2012.

[19] A. Brock, T. Lim, J. M. Ritchie, and N. Weston, "Generative and discriminative voxel modeling with convolutional neural networks," *arXiv preprint arXiv:1608.04236*, 2016.

[20] A. Ranjan, T. Bolkart, S. Sanyal, and M. J. Black, "Generating 3D faces using convolutional mesh autoencoders," in *European Conference on Computer Vision (ECCV)*, 2018, pp. 725–741. [Online]. Available: http://coma.is.tue.mpg.de/

[21] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Curran Associates Inc.*, ser. NIPS'16, Red Hook, NY, USA, 2016, p. 3844–3852.

[22] G. Bouritsas, S. Bokhnyak, S. Ploumpis, M. Bronstein, and S. Zafeiriou, "Neural 3d morphable models: Spiral convolutional networks for 3d shape representation learning and generation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 7213–7222.

[23] I. Lim, A. Dielen, M. Campen, and L. Kobbelt, "A simple approach to intrinsic correspondence learning on unstructured 3d meshes," in *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, 2018, pp. 0–0.

[24] E. Tretschk, A. Tewari, M. Zollhöfer, V. Golyanik, and C. Theobalt, "Demea: Deep mesh autoencoders for non-rigidly deforming objects," in *European Conference on Computer Vision*. Springer, 2020, pp. 601–617.

[25] O. Litany, A. Bronstein, M. Bronstein, and A. Makadia, "Deformable shape completion with graph convolutional autoencoders," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 1886–1895.

[26] F. Bogo, J. Romero, G. Pons-Moll, and M. J. Black, " Dynamic FAUST: Registering Human Bodies in Motion ," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Los Alamitos, CA, USA: IEEE Computer Society, Jul. 2017, pp. 5573–5582. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/CVPR.2017.591

[27] J. Malik, A. Elhayek, F. Nunnari, K. Varanasi, K. Tamaddon, A. Heloir, and D. Stricker, "Deephps: End-to-end estimation of 3d hand pose and shape by learning from synthetic depth," in *2018 International Conference on 3D Vision (3DV)*. IEEE, 2018, pp. 110–119.

[28] M. Loper, N. Mahmood, J. Romero, G. Pons-Moll, and M. J. Black, "Smpl: A skinned

multi-person linear model," *ACM transactions on graphics (TOG)*, vol. 34, no. 6, pp. 1–16, 2015.

[29] J. Romero, D. Tzionas, and M. J. Black, "Embodied hands: Modeling and capturing hands and bodies together," *arXiv preprint arXiv:2201.02610*, 2022.

[30] M. Garland and P. S. Heckbert, "Surface simplification using quadric error metrics," in *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH '97. USA: ACM Press/Addison-Wesley Publishing Co., 1997, p. 209–216. [Online]. Available: https://doi.org/10.1145/258734.258849

[31] H. Hoppe, "New quadric metric for simplifying meshes with appearance attributes," in *Proceedings Visualization'99 (Cat. No. 99CB37067)*. IEEE, 1999, pp. 59–510.

[32] M. Garland and P. S. Heckbert, "Simplifying surfaces with color and texture using quadric error metrics," in *Proceedings Visualization'98 (Cat. No. 98CB36276)*. IEEE, 1998, pp. 263–269.

[33] L. Yao, S. Huang, H. Xu, and P. Li, "Quadratic error metric mesh simplification algorithm based on discrete curvature," *Mathematical Problems in Engineering*, vol. 2015, 2015.

[34] S. Hahner and J. Garcke, "Mesh convolutional autoencoder for semi-regular meshes of different sizes," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2022, pp. 885–894.

[35] C. Tang, X. Sun, A. Gomes, J. Wallner, and H. Pottmann, "Form-finding with polyhedral meshes made simple," *ACM Transactions on Graphics (TOG)*, vol. 33, no. 4, pp. 1–9, 2014.

[36] Y. Liu, H. Pottmann, J. Wallner, Y.-L. Yang, and W. Wang, "Geometric modeling with conical meshes and developable surfaces," in *ACM SIGGRAPH 2006 Papers*, 2006, pp. 681–689.

[37] R. Hanocka, A. Hertz, N. Fish, R. Giryes, S. Fleishman, and D. Cohen-Or, "Meshcnn: a network with an edge," *ACM Transactions on Graphics (TOG)*, vol. 38, no. 4, pp. 1–12, 2019.

[38] F. R. Chung and F. C. Graham, *Spectral graph theory*. American Mathematical Soc., 1997, no. 92.

[39] D. K. Hammond, P. Vandergheynst, and R. Gribonval, "Wavelets on graphs via spectral graph theory," *Applied and Computational Harmonic Analysis*, vol. 30, no. 2, pp. 129–150, 2011.

[40] N. Shervashidze, P. Schweitzer, E. J. Van Leeuwen, K. Mehlhorn, and K. M. Borgwardt, "Weisfeiler-lehman graph kernels." *Journal of Machine Learning Research*, vol. 12, no. 9, 2011.

[41] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," *Advances in neural information processing systems*, vol. 30, 2017.

[42] F. Monti, D. Boscaini, J. Masci, E. Rodola, J. Svoboda, and M. M. Bronstein, "Geometric deep learning on graphs and manifolds using mixture model cnns," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 5115–5124.

[43] J. Atwood and D. Towsley, "Diffusion-convolutional neural networks," in *Advances in neural information processing systems*, 2016, pp. 1993–2001.

[44] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[45] D. Boscaini, J. Masci, E. Rodolà, and M. Bronstein, "Learning shape correspondence with anisotropic convolutional neural networks," in *Advances in Neural Information Processing Systems*, D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, Eds., vol. 29. Curran Associates, Inc., 2016. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2016/file/228499b55310264a8ea0e27b6e7c6ab6-Paper.pdf

[46] J. Masci, D. Boscaini, M. Bronstein, and P. Vandergheynst, "Geodesic convolutional neural networks on riemannian manifolds," in *Proceedings of the IEEE international conference on computer vision workshops*, 2015, pp. 37–45.

[47] Q. Xu, X. Sun, C.-Y. Wu, P. Wang, and U. Neumann, "Grid-gcn for fast and scalable point cloud learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 5661–5670.

[48] N. Verma, E. Boyer, and J. Verbeek, "Feastnet: Feature-steered graph convolutions for 3d shape analysis," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 2598–2606.

[49] I. S. Dhillon, Y. Guan, and B. Kulis, "Weighted graph cuts without eigenvectors a multilevel approach," *IEEE transactions on pattern analysis and machine intelligence*, vol. 29, no. 11, pp. 1944–1957, 2007.

[50] Z. Chen and T.-K. Kim, "Learning feature aggregation for deep 3d morphable models," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 13 164–13 173.

[51] H. Gouraud, "Continuous shading of curved surfaces," *IEEE Transactions on Computers*, vol. C-20, no. 6, pp. 623–629, 1971.

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

[53] S. Gong, L. Chen, M. Bronstein, and S. Zafeiriou, "Spiralnet++: A fast and highly efficient mesh convolution operator," in *Proceedings of the IEEE International Conference on Computer Vision Workshops*, 2019, pp. 0–0.

[54] F. Pichi, B. Moya, and J. S. Hesthaven, "A graph convolutional autoencoder approach to model order reduction for parametrized pdes," *Journal of Computational Physics*, vol. 501, p. 112762, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0021999124000111

[55] A. Almalki and L. J. Latecki, "Self-supervised learning with masked autoencoders for teeth segmentation from intra-oral 3d scans," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2024, pp. 7820–7830.

[56] H.-T. D. Liu, V. G. Kim, S. Chaudhuri, N. Aigerman, and A. Jacobson, "Neural subdivision," *ACM Trans. Graph.*, vol. 39, no. 4, Aug. 2020. [Online]. Available: https://doi.org/10.1145/3386569.3392418

[57] J. Cohen, M. Olano, and D. Manocha, "Appearance-preserving simplification," in *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH '98. New York, NY, USA: Association for Computing Machinery, 1998, p. 115–122. [Online]. Available: https://doi.org/10.1145/280814.280832

[58] H. G. Barrow, J. M. Tenenbaum, R. C. Bolles, and H. C. Wolf, "Parametric correspondence and chamfer matching: Two new techniques for image matching," in *Proceedings*

of the 5th International Joint Conference on Artificial Intelligence - Volume 2, ser. IJCAI'77. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1977, p. 659–663.

[59] H. Fan, H. Su, and L. J. Guibas, "A point set generation network for 3d object reconstruction from a single image," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 605–613.

[60] Z. Gao, G. Zhai, J. Zhang, J. Yan, Y. Yang, and X. Yang, "Learning local neighboring structure for robust 3d shape representation," in *AAAI Conference on Artificial Intelligence*, 2020. [Online]. Available: https://api.semanticscholar.org/CorpusID:219402126

[61] H. Bouzid and L. Ballihi, "Spatr: Mocap 3d human action recognition based on spiral auto-encoder and transformer network," *Computer Vision and Image Understanding*, vol. 241, p. 103974, 2024.

[62] D. Zorin, P. Schroder, A. DeRose, L. Kobbelt, A. Levin, and W. Sweldens, *Subdivision for modeling and animation.* ACM Press/Addison-Wesley Publishing Co., 2000.

[63] D. Doo and M. Sabin, "Behaviour of recursive division surfaces near extraordinary points," *Computer-Aided Design*, vol. 10, no. 6, pp. 356–360, 1978.

[64] C. LOOP, "Smooth subdivision surfaces based on triangles," *Master's thesis, University of Utah, Department of Mathematics*, 1987.

[65] E. Catmull and J. Clark, "Recursively generated b-spline surfaces on arbitrary topological meshes," in *Seminal graphics: pioneering efforts that shaped the field.* ACM Press/Addison-Wesley Publishing Co., 1998, pp. 183–188.

[66] L. Kobbelt, "3-subdivision," in *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques.* USA: ACM Press/Addison-Wesley Publishing Co., 2000, p. 103–112.

[67] H.-T. D. Liu, V. G. Kim, S. Chaudhuri, N. Aigerman, and A. Jacobson, "Neural subdivision," *ACM Trans. Graph.*, vol. 39, no. 4, 2020.

[68] S.-M. Hu, Z.-N. Liu, M.-H. Guo, J.-X. Cai, J. Huang, T.-J. Mu, and R. R. Martin, "Subdivision-based mesh convolution networks," *ACM Transactions on Graphics (TOG)*, vol. 41, no. 3, pp. 1–16, 2022.

[69] P. Mullen, Y. Tong, P. Alliez, and M. Desbrun, "Spectral conformal parameterization," in *Computer Graphics Forum*, vol. 27. Wiley Online Library, 2008, pp. 1487–1494.

[70] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga *et al.*, "Pytorch: An imperative style, high-performance deep learning library," *Advances in neural information processing systems*, vol. 32, 2019.

[71] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Icml*, 2010.

[72] Y. Qiu, "Spectra: A header-only c++ library for large scale eigenvalue problems," https://github.com/yixuan/spectra, 2015.

[73] F. Bogo, J. Romero, G. Pons-Moll, and M. J. Black, "Dynamic FAUST: Registering human bodies in motion," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, Jul. 2017.

[74] O. Litany, T. Remez, E. Rodola, A. Bronstein, and M. Bronstein, " Deep Functional Maps: Structured Prediction for Dense Shape Correspondence ," in *2017 IEEE International Conference on Computer Vision (ICCV)*. Los Alamitos, CA, USA: IEEE Computer Society, Oct. 2017, pp. 5660–5668. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/ICCV.2017.603

[75] M. Ovsjanikov, M. Ben-Chen, J. Solomon, A. Butscher, and L. Guibas, "Functional maps: a flexible representation of maps between shapes," *ACM Trans. Graph.*, vol. 31, no. 4, Jul. 2012. [Online]. Available: https://doi.org/10.1145/2185520.2185526

[76] A. M. Bronstein, M. M. Bronstein, and R. Kimmel, "Generalized multidimensional scaling: A framework for isometry-invariant partial surface matching," *Proceedings of the National Academy of Sciences*, vol. 103, no. 5, pp. 1168–1172, 2006. [Online]. Available: https://www.pnas.org/doi/abs/10.1073/pnas.0508601103

[77] Q.-X. Huang, B. Adams, M. Wicke, and L. J. Guibas, "Non-rigid registration under isometric deformations," in *Proceedings of the Symposium on Geometry Processing*, ser. SGP '08. Goslar, DEU: Eurographics Association, 2008, p. 1449–1457.

[78] M. Ovsjanikov, Q. Mérigot, F. Mémoli, and L. Guibas, "One point isometric matching with the heat kernel," *Computer Graphics Forum*, vol. 29, no. 5, pp. 1555–1564, 2010.

[Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1467-8659.2010.01764.x

[79] O. van Kaick, H. Zhang, G. Hamarneh, and D. Cohen-Or, "A survey on shape correspondence," *Computer Graphics Forum*, vol. 30, no. 6, pp. 1681–1707, 2011. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1467-8659.2011.01884.x

[80] G. K. Tam, Z.-Q. Cheng, Y.-K. Lai, F. Langbein, Y. Liu, A. D. Marshall, R. Martin, X. Sun, and P. Rosin, "Registration of 3d point clouds and meshes: A survey from rigid to nonrigid," *IEEE Transactions on Visualization and Computer Graphics*, vol. 19, no. 7, p. 1199–1217, Jul. 2013. [Online]. Available: https://doi.org/10.1109/TVCG.2012.310

[81] S. Biasotti, A. Cerri, A. Bronstein, and M. Bronstein, "Recent trends, applications, and perspectives in 3d shape similarity assessment," *Comput. Graph. Forum*, vol. 35, no. 6, p. 87–119, Sep. 2016. [Online]. Available: https://doi.org/10.1111/cgf.12734

[82] B. Deng, Y. Yao, R. M. Dyke, and J. Zhang, "A survey of non-rigid 3d registration," *Computer Graphics Forum*, vol. 41, no. 2, pp. 559–589, 2022. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.14502

[83] Y. Sahillioğlu and Y. Yemez, "3d shape correspondence by isometry-driven greedy optimization," *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 453–458, 2010. [Online]. Available: https://api.semanticscholar.org/CorpusID:6988633

[84] A. Tevs, M. Bokeloh, M. Wand, A. Schilling, and H.-P. Seidel, "Isometric registration of ambiguous and partial data," *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1185–1192, 2009. [Online]. Available: https://api.semanticscholar.org/CorpusID:15366827

[85] Y. Sahillioğlu, "A genetic isometric shape correspondence algorithm with adaptive sampling," *ACM Trans. Graph.*, vol. 37, no. 5, Oct. 2018. [Online]. Available: https://doi.org/10.1145/3243593

[86] Y. Lipman and T. Funkhouser, "Möbius voting for surface correspondence," *ACM Trans. Graph.*, vol. 28, no. 3, Jul. 2009. [Online]. Available: https://doi.org/10.1145/1531326.1531378

[87] V. G. Kim, Y. Lipman, and T. Funkhouser, "Blended intrinsic maps," *ACM Trans. Graph.*, vol. 30, no. 4, Jul. 2011. [Online]. Available: https://doi.org/10.1145/2010324.1964974

[88] N. Aigerman, R. Poranne, and Y. Lipman, "Seamless surface mappings," *ACM Trans. Graph.*, vol. 34, no. 4, Jul. 2015. [Online]. Available: https://doi.org/10.1145/2766921

[89] H. Maron, N. Dym, I. Kezurer, S. Kovalsky, and Y. Lipman, "Point registration via efficient convex relaxation," *ACM Trans. Graph.*, vol. 35, no. 4, Jul. 2016. [Online]. Available: https://doi.org/10.1145/2897824.2925913

[90] I. Kezurer, S. Z. Kovalsky, R. Basri, and Y. Lipman, "Tight relaxation of quadratic matching," *Computer Graphics Forum*, vol. 34, no. 5, pp. 115–128, 2015. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.12701

[91] N. Dym, H. Maron, and Y. Lipman, "DS++: a flexible, scalable and provably tight relaxation for matching problems," *ACM Trans. Graph.*, vol. 36, no. 6, Nov. 2017. [Online]. Available: https://doi.org/10.1145/3130800.3130826

[92] A. Kovnatsky, M. M. Bronstein, A. M. Bronstein, K. Glashoff, and R. Kimmel, "Coupled quasi-harmonic bases," *Computer Graphics Forum*, vol. 32, no. 2pt4, pp. 439–448, 2013. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.12064

[93] J. Ren, A. Poulenard, P. Wonka, and M. Ovsjanikov, "Continuous and orientation-preserving correspondences via functional maps," *ACM Trans. Graph.*, vol. 37, no. 6, Dec. 2018. [Online]. Available: https://doi.org/10.1145/3272127.3275040

[94] J. Ren, S. Melzi, M. Ovsjanikov, and P. Wonka, "Maptree: recovering multiple solutions in the space of maps," *ACM Trans. Graph.*, vol. 39, no. 6, Nov. 2020. [Online]. Available: https://doi.org/10.1145/3414685.3417800

[95] J. Ren, S. Melzi, P. Wonka, and M. Ovsjanikov, "Discrete optimization for shape matching," *Computer Graphics Forum*, vol. 40, no. 5, pp. 81–96, 2021. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.14359

[96] D. Ezuz and M. Ben-Chen, "Deblurring and denoising of maps between shapes," *Comput. Graph. Forum*, vol. 36, no. 5, p. 165–174, Aug. 2017. [Online]. Available: https://doi.org/10.1111/cgf.13254

[97] E. Rodolà, L. Cosmo, M. M. Bronstein, A. Torsello, and D. Cremers, "Partial functional

correspondence," *Computer Graphics Forum*, vol. 36, no. 1, pp. 222–236, 2017. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.12797

[98] M. Vestner, R. Litman, E. Rodolà, A. M. Bronstein, and D. Cremers, "Product manifold filter: Non-rigid shape correspondence via kernel density estimation in the product space," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6681–6690, 2017. [Online]. Available: https://api.semanticscholar.org/CorpusID:10436114

[99] R. Huang and M. Ovsjanikov, "Adjoint map representation for shape analysis and matching," *Computer Graphics Forum*, vol. 36, no. 5, pp. 151–163, 2017. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.13253

[100] R. Magnet and M. Ovsjanikov, "Scalable and efficient functional map computations on dense meshes," *Computer Graphics Forum*, vol. 42, no. 2, pp. 89–101, 2023. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.14746

[101] R. M. Rustamov, "Laplace-Beltrami Eigenfunctions for Deformation Invariant Shape Representation," in *Geometry Processing*, A. Belyaev and M. Garland, Eds. The Eurographics Association, 2007.

[102] J. Sun, M. Ovsjanikov, and L. Guibas, "A Concise and Provably Informative Multi-Scale Signature Based on Heat Diffusion," *Computer Graphics Forum*, 2009.

[103] M. Aubry, U. Schlickewei, and D. Cremers, " The wave kernel signature: A quantum mechanical approach to shape analysis ," in *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*. Los Alamitos, CA, USA: IEEE Computer Society, Nov. 2011, pp. 1626–1633. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/ICCVW.2011.6130444

[104] F. Tombari, S. Salti, and L. Di Stefano, "Unique signatures of histograms for local surface description," in *Proceedings of the 11th European Conference on Computer Vision Conference on Computer Vision: Part III*, ser. ECCV'10. Berlin, Heidelberg: Springer-Verlag, 2010, p. 356–369.

[105] G. Shamai and R. Kimmel, " Geodesic Distance Descriptors ," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Los Alamitos,

CA, USA: IEEE Computer Society, Jul. 2017, pp. 3624–3632. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/CVPR.2017.386

[106] R. Litman and A. M. Bronstein, " Learning Spectral Descriptors for Deformable Shape Correspondence ," *IEEE Transactions on Pattern Analysis & Machine Intelligence*, vol. 36, no. 01, pp. 171–180, Jan. 2014. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/TPAMI.2013.148

[107] N. Sharp, S. Attaiki, K. Crane, and M. Ovsjanikov, "Diffusionnet: Discretization agnostic learning on surfaces," *ACM Transactions on Graphics (TOG)*, vol. 41, no. 3, pp. 1–16, 2022.

[108] I. Lang, D. Ginzburg, S. Avidan, and D. Raviv, "Dpc: Unsupervised deep point correspondence via cross and self construction," *2021 International Conference on 3D Vision (3DV)*, pp. 1442–1451, 2020. [Online]. Available: https://api.semanticscholar.org/CorpusID:239016800

[109] S. Attaiki and M. Ovsjanikov, "Ncp: neural correspondence prior for effective unsupervised shape matching," in *Proceedings of the 36th International Conference on Neural Information Processing Systems*, ser. NIPS '22. Red Hook, NY, USA: Curran Associates Inc., 2022.

[110] S. Melzi, J. Ren, E. Rodolà, A. Sharma, P. Wonka, and M. Ovsjanikov, "Zoomout: spectral upsampling for efficient shape correspondence," *ACM Trans. Graph.*, vol. 38, no. 6, Nov. 2019. [Online]. Available: https://doi.org/10.1145/3355089.3356524

[111] R. Huang, J. Ren, P. Wonka, and M. Ovsjanikov, "Consistent zoomout: Efficient spectral map synchronization," *Computer Graphics Forum*, vol. 39, no. 5, pp. 265–278, 2020. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.14084

[112] L. Li, N. Donati, and M. Ovsjanikov, "Learning multi-resolution functional maps with spectral attention for robust shape matching," *Advances in Neural Information Processing Systems*, vol. 35, pp. 29 336–29 349, 2022.

[113] M. Ovsjanikov, E. Corman, M. Bronstein, E. Rodolà, M. Ben-Chen, L. Guibas, F. Chazal, and A. Bronstein, "Computing and processing correspondences with functional maps," in *ACM SIGGRAPH 2017 Courses*, ser. SIGGRAPH '17. New

York, NY, USA: Association for Computing Machinery, 2017. [Online]. Available: https://doi.org/10.1145/3084873.3084877

[114] H.-T. D. Liu, A. Jacobson, and M. Ovsjanikov, "Spectral coarsening of geometric operators," *ACM Trans. Graph.*, vol. 38, no. 4, Jul. 2019. [Online]. Available: https://doi.org/10.1145/3306346.3322953

[115] R. Schmidt, C. Grimm, and B. Wyvill, "Interactive decal compositing with discrete exponential maps," *ACM Trans. Graph.*, vol. 25, no. 3, p. 605–613, Jul. 2006. [Online]. Available: https://doi.org/10.1145/1141911.1141930

[116] K. Crane, C. Weischedel, and M. Wardetzky, "The heat method for distance computation," *Commun. ACM*, vol. 60, no. 11, p. 90–99, Oct. 2017. [Online]. Available: https://doi.org/10.1145/3131280

[117] D. Nogneng and M. Ovsjanikov, "Informative descriptor preservation via commutativity for shape matching," *Comput. Graph. Forum*, vol. 36, no. 2, p. 259–267, May 2017. [Online]. Available: https://doi.org/10.1111/cgf.13124

[118] M. Ovsjanikovs, *Spectral methods for isometric shape matching and symmetry detection.* Stanford University, 2011.

[119] F. Bogo, J. Romero, M. Loper, and M. J. Black, "Faust: Dataset and evaluation for 3d mesh registration," *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3794–3801, 2014. [Online]. Available: https://api.semanticscholar.org/CorpusID:206592437

[120] A. Bronstein, M. Bronstein, and R. Kimmel, *Numerical Geometry of Non-Rigid Shapes*, 1st ed. Springer Publishing Company, Incorporated, 2008.

[121] D. Anguelov, P. Srinivasan, D. Koller, S. Thrun, J. Rodgers, and J. Davis, "Scape: shape completion and animation of people," *ACM Trans. Graph.*, vol. 24, no. 3, p. 408–416, Jul. 2005. [Online]. Available: https://doi.org/10.1145/1073204.1073207

[122] G. Barequet and M. Sharir, "Filling gaps in the boundary of a polyhedron," *Computer Aided Geometric Design*, vol. 12, no. 2, pp. 207–229, 1995. [Online]. Available: https://www.sciencedirect.com/science/article/pii/016783969400011G

[123] J.-P. Pernot, G. Moraru, and P. Véron, "Filling holes in meshes using a mechanical

model to simulate the curvature variation minimization," *Computers & Graphics*, vol. 30, no. 6, pp. 892–902, 2006. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0097849306001452

[124] W. Zhao, S. Gao, and H. Lin, "A robust hole-filling algorithm for triangular mesh," in *2007 10th IEEE International Conference on Computer-Aided Design and Computer Graphics*, 2007, pp. 22–22.

[125] A. Sharf, M. Alexa, and D. Cohen-Or, "Context-based surface completion," *ACM Trans. Graph.*, vol. 23, no. 3, p. 878–887, Aug. 2004. [Online]. Available: https://doi.org/10.1145/1015706.1015814

[126] G. Harary, A. Tal, and E. Grinspun, "Context-based coherent surface completion," *ACM Trans. Graph.*, vol. 33, no. 1, Feb. 2014. [Online]. Available: https://doi.org/10.1145/2532548

[127] U. Clarenz, U. Diewald, G. Dziuk, M. Rumpf, and R. Rusu, "A finite element method for surface restoration with smooth boundary conditions," *Computer Aided Geometric Design*, vol. 21, no. 5, pp. 427–445, 2004. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167839604000202

[128] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao, "3d shapenets: A deep representation for volumetric shapes," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 1912–1920.

[129] A. Dai, C. R. Qi, and M. Nießner, "Shape completion using 3d-encoder-predictor cnns and shape synthesis," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 6545–6554.

[130] W. Yuan, T. Khot, D. Held, C. Mertz, and M. Hebert, "Pcn: Point completion network," in *2018 International Conference on 3D Vision (3DV)*, 2018, pp. 728–737.

[131] X. Wen, T. Li, Z. Han, and Y.-S. Liu, " Point Cloud Completion by Skip-Attention Network With Hierarchical Folding ," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Los Alamitos, CA, USA: IEEE Computer Society, Jun. 2020, pp. 1936–1945. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/CVPR42600.2020.00201

[132] X. Wen, P. Xiang, Y. Cao, P. Wan, W. Zheng, and Y.-S. Liu, "Pmp-net++: Point cloud completion by transformer-enhanced multi-step point moving paths," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, pp. 852–867, 2022. [Online]. Available: https://api.semanticscholar.org/CorpusID:247011998

[133] L. Wang, J. Li, S. Guo, and S. Han, "A cascaded graph convolutional network for point cloud completion," *The Visual Computer*, vol. 41, no. 1, pp. 659–674, 2025. [Online]. Available: https://doi.org/10.1007/s00371-024-03354-x

[134] J. Li, J. Zhang, X. Zhang, and M. Chen, "Edge-guided generative network with attention for point cloud completion," *The Visual Computer*, vol. 41, no. 2, pp. 785–798, 2025. [Online]. Available: https://doi.org/10.1007/s00371-024-03364-9

[135] Y. Siddiqui, A. Alliegro, A. Artemov, T. Tommasi, D. Sirigatti, V. Rosov, A. Dai, and M. Nießner, "Meshgpt: Generating triangle meshes with decoder-only transformers," *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 19 615–19 625, 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID:265457242

[136] Y. Chen, T. He, D. Huang, W. Ye, S. Chen, J. Tang, X. Chen, Z. Cai, L. Yang, G. Yu, G. Lin, and C. Zhang, "Meshanything: Artist-created mesh generation with autoregressive transformers," *ArXiv*, vol. abs/2406.10163, 2024. [Online]. Available: https://api.semanticscholar.org/CorpusID:270521478

[137] J. Tang, Z. Li, Z. Hao, X. Liu, G. Zeng, M.-Y. Liu, and Q. Zhang, "Edgerunner: Auto-regressive auto-encoder for artistic mesh generation," *arXiv preprint arXiv:2409.18114*, 2024.

[138] D. Perez, Y. Shen, and J. Li, "Mesh convolutional networks with face and vertex feature operators," *IEEE Transactions on Visualization and Computer Graphics*, vol. 29, no. 3, pp. 1678–1690, 2023.

[139] H. Zhang, X. Li, J. Kim, S. Cheng, and C. El Amm, "Neural qslim for mesh autoencoders," in *International Conference on Artificial Neural Networks*. Springer, 2025, pp. 51–65.

[140] S. Koch, A. Matveev, Z. Jiang, F. Williams, A. Artemov, E. Burnaev, M. Alexa, D. Zorin,

and D. Panozzo, "Abc: A big cad model dataset for geometric deep learning," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[141] Q. Zhou and A. Jacobson, "Thingi10k: A dataset of 10,000 3d-printing models," *arXiv preprint arXiv:1605.04797*, 2016.

[142] T. Shen, Z. Li, M. Law, M. Atzmon, S. Fidler, J. Lucas, J. Gao, and N. Sharp, "Spacemesh: A continuous representation for learning manifold surface meshes," in *SIGGRAPH Asia 2024 Conference Papers (SA Conference Papers '24)*. New York, NY, USA: ACM, 2024, p. 11. [Online]. Available: https://doi.org/10.1145/3680528.3687634

[143] R. Hanocka, G. Metzer, R. Giryes, and D. Cohen-Or, "Point2mesh: a self-prior for deformable meshes," *ACM Trans. Graph.*, vol. 39, no. 4, Aug. 2020. [Online]. Available: https://doi.org/10.1145/3386569.3392415

[144] X. Wei, Z. Chen, Y. Fu, Z. Cui, and Y. Zhang, "Deep hybrid self-prior for full 3d mesh generation," in *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021, pp. 5785–5794.

[145] X. Yan, L. Lin, N. J. Mitra, D. Lischinski, D. Cohen-Or, and H. Huang, "Shapeformer: Transformer-based shape completion via sparse representation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022.

[146] L. Chu, H. Pan, and W. Wang, "Unsupervised shape completion via deep prior in the neural tangent kernel perspective," *ACM Transactions on Graphics (TOG)*, vol. 40, pp. 1 – 17, 2021. [Online]. Available: https://api.semanticscholar.org/CorpusID:233296378

[147] H. Mittal, B. Okorn, A. Jangid, and D. Held, "Self-supervised point cloud completion via inpainting," in *Proceedings of British Machine Vision Conference (BMVC)*, 2021.

[148] J. Chibane, T. Alldieck, and G. Pons-Moll, "Implicit functions in feature space for 3d shape reconstruction and completion," *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6968–6979, 2020. [Online]. Available: https://api.semanticscholar.org/CorpusID:211817828

[149] P. Liepa, "Filling Holes in Meshes," in *Eurographics Symposium on Geometry Processing*, L. Kobbelt, P. Schroeder, and H. Hoppe, Eds. The Eurographics Association, 2003.

[150] V. Kraevoy and A. Sheffer, "Template-Based Mesh Completion," in *Eurographics Sympo-*

sium on Geometry Processing 2005, M. Desbrun and H. Pottmann, Eds. The Eurographics Association, 2005.

[151] J. Davis, S. Marschner, M. Garr, and M. Levoy, "Filling holes in complex surfaces using volumetric diffusion," in *Proceedings. First International Symposium on 3D Data Processing Visualization and Transmission*, 2002, pp. 428–441.

[152] J. Verdera, V. Caselles, M. Bertalmio, and G. Sapiro, "Inpainting surface holes," in *Proceedings 2003 International Conference on Image Processing (Cat. No. 03CH37429)*, vol. 2. IEEE, 2003, pp. II–903.

[153] T.-q. Guo, J.-j. Li, J.-g. Weng, and Y.-t. Zhuang, "Filling holes in complex surfaces using oriented voxel diffusion," in *2006 International Conference on Machine Learning and Cybernetics*, 2006, pp. 4370–4375.

[154] S. Bischoff, D. Pavic, and L. Kobbelt, "Automatic restoration of polygon models," *ACM Trans. Graph.*, vol. 24, no. 4, p. 1332–1352, Oct. 2005. [Online]. Available: https://doi.org/10.1145/1095878.1095883

[155] J. Varley, C. DeChant, A. Richardson, J. Ruales, and P. Allen, "Shape completion enabled robotic grasping," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE Press, 2017, p. 2442–2447. [Online]. Available: https://doi.org/10.1109/IROS.2017.8206060

[156] X. Han, Z. Li, H. Huang, E. Kalogerakis, and Y. Yu, " High-Resolution Shape Completion Using Deep Neural Networks for Global Structure and Local Geometry Inference ," in *2017 IEEE International Conference on Computer Vision (ICCV)*. Los Alamitos, CA, USA: IEEE Computer Society, Oct. 2017, pp. 85–93. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/ICCV.2017.19

[157] X. Chen, B. Chen, and N. J. Mitra, "Unpaired point cloud completion on real scans using adversarial training," *ArXiv*, vol. abs/1904.00069, 2019. [Online]. Available: https://api.semanticscholar.org/CorpusID:90258365

[158] M. Sarmad, H. J. Lee, and Y. M. Kim, "Rl-gan-net: A reinforcement learning agent controlled gan network for real-time point cloud shape completion," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[159] X. Wen, P. Xiang, Z. Han, Y.-P. Cao, P. Wan, W. Zheng, and Y.-S. Liu, "Pmp-net: Point cloud completion by learning multi-step point moving paths," *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7439–7448, 2020. [Online]. Available: https://api.semanticscholar.org/CorpusID:227336293

[160] J. J. Park, P. Florence, J. Straub, R. Newcombe, and S. Lovegrove, " DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation ," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Los Alamitos, CA, USA: IEEE Computer Society, Jun. 2019, pp. 165–174. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/CVPR.2019.00025

[161] Y. Rao, Y. Nie, and A. Dai, "Patchcomplete: learning multi-resolution patch priors for 3d shape completion on unseen categories," in *Proceedings of the 36th International Conference on Neural Information Processing Systems*, ser. NIPS '22. Red Hook, NY, USA: Curran Associates Inc., 2022.

[162] P. Mittal, Y.-C. Cheng, M. Singh, and S. Tulsiani, "Autosdf: Shape priors for 3d completion, reconstruction and generation," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 306–315.

[163] S. Hattori, T. Yatagawa, Y. Ohtake, and H. Suzuki, "Learning self-prior for mesh inpainting using self-supervised graph convolutional networks," *IEEE Transactions on Visualization and Computer Graphics*, vol. 31, no. 2, pp. 1448–1464, 2024.

[164] M. Attene, "A lightweight approach to repairing digitized polygon meshes," *Vis. Comput.*, vol. 26, no. 11, p. 1393–1406, Nov. 2010. [Online]. Available: https://doi.org/10.1007/s00371-010-0416-3

[165] L. Chu, H. Pan, and W. Wang, "Unsupervised shape completion via deep prior in the neural tangent kernel perspective," *ACM Transactions on Graphics (TOG)*, vol. 40, no. 3, pp. 1–17, 2021.

[166] A. X. Chang, T. A. Funkhouser, L. J. Guibas, P. Hanrahan, Q.-X. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, J. Xiao, L. Yi, and F. Yu, "Shapenet: An information-rich 3d model repository," *ArXiv*, vol. abs/1512.03012, 2015. [Online]. Available: https://api.semanticscholar.org/CorpusID:2554264

[167] J. Behley, M. Garbade, A. Milioto, J. Quenzel, S. Behnke, C. Stachniss, and J. Gall, "Semantickitti: A dataset for semantic scene understanding of lidar sequences," *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 9296–9306, 2019. [Online]. Available: https://api.semanticscholar.org/CorpusID:199441943

[168] F. Williams, T. Schneider, C. Silva, D. Zorin, J. Bruna, and D. Panozzo, "Deep geometric prior for surface reconstruction," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 10 130–10 139.

[169] X. Wei, Z. Chen, Y. Fu, Z. Cui, and Y. Zhang, "Deep hybrid self-prior for full 3d mesh generation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 5805–5814.

[170] B. Ma, Y.-S. Liu, and Z. Han, " Reconstructing Surfaces for Sparse Point Clouds with On-Surface Priors ," in *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Los Alamitos, CA, USA: IEEE Computer Society, Jun. 2022, pp. 6305–6315. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/CVPR52688.2022.00621

[171] C. Chen, Y.-S. Liu, and Z. Han, "Unsupervised inference of signed distance functions from single sparse point clouds without learning priors," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 17 712–17 723.

[172] G. Metzer, R. Hanocka, R. Giryes, and D. Cohen-Or, "Self-sampling for neural point cloud consolidation," *ACM Transactions on Graphics (TOG)*, vol. 40, no. 5, pp. 1–14, 2021.

[173] N. Feng and K. Crane, "A heat method for generalized signed distance," *ACM Trans. Graph.*, vol. 43, no. 4, Jul. 2024. [Online]. Available: https://doi.org/10.1145/3658220

[174] W. E. Lorensen and H. E. Cline, "Marching cubes: A high resolution 3d surface construction algorithm," in *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH '87. New York, NY, USA: Association for Computing Machinery, 1987, p. 163–169. [Online]. Available: https://doi.org/10.1145/37401.37422

[175] M. Garland and P. S. Heckbert, "Surface simplification using quadric error metrics," in *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive*

Techniques, ser. SIGGRAPH '97. USA: ACM Press/Addison-Wesley Publishing Co., 1997, p. 209–216. [Online]. Available: https://doi.org/10.1145/258734.258849

[176] M. Kazhdan, M. Bolitho, and H. Hoppe, "Poisson surface reconstruction," in *Proceedings of the Fourth Eurographics Symposium on Geometry Processing*, ser. SGP '06. Goslar, DEU: Eurographics Association, 2006, p. 61–70.

[177] P. Alliez, E. C. d. Verdière, O. Devillers, and M. Isenburg, "Isotropic surface remeshing," in *Proceedings of the Shape Modeling International 2003*, ser. SMI '03. USA: IEEE Computer Society, 2003, p. 49.

[178] N. Sharp, M. Gillespie, and K. Crane, "Geometry processing with intrinsic triangulations," in *ACM SIGGRAPH 2021 Courses*, ser. SIGGRAPH '21. New York, NY, USA: Association for Computing Machinery, 2021. [Online]. Available: https://doi.org/10.1145/3450508.3464592

[179] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, no. 1, pp. 269–271, 1959. [Online]. Available: https://doi.org/10.1007/BF01386390

[180] K. Polthier and M. Schmies, "Straightest geodesics on polyhedral surfaces," in *ACM SIGGRAPH 2006 Courses*, ser. SIGGRAPH '06. New York, NY, USA: Association for Computing Machinery, 2006, p. 30–38. [Online]. Available: https://doi.org/10.1145/1185657.1185664

[181] V. Surazhsky, T. Surazhsky, D. Kirsanov, S. J. Gortler, and H. Hoppe, "Fast exact and approximate geodesics on meshes," *ACM Trans. Graph.*, vol. 24, no. 3, p. 553–560, Jul. 2005. [Online]. Available: https://doi.org/10.1145/1073204.1073228

[182] N. Sharp and K. Crane, "You can find geodesic paths in triangle meshes by just flipping edges," *ACM Trans. Graph.*, vol. 39, no. 6, Nov. 2020. [Online]. Available: https://doi.org/10.1145/3414685.3417839

[183] K. J. Weiler, *Topological structures for geometric modeling (Boundary representation, manifold, radial edge structure).* Rensselaer Polytechnic Institute, 1986.

[184] I. Lim, A. Dielen, M. Campen, and L. Kobbelt, "A simple approach to intrinsic correspondence learning on unstructured 3d meshes," in *Computer Vision –*

ECCV 2018 Workshops: Munich, Germany, September 8-14, 2018, Proceedings, Part III. Berlin, Heidelberg: Springer-Verlag, 2018, p. 349–362. [Online]. Available: https://doi.org/10.1007/978-3-030-11015-4_26

[185] G. Bouritsas, S. Bokhnyak, S. Ploumpis, S. Zafeiriou, and M. Bronstein, "Neural 3d morphable models: Spiral convolutional networks for 3d shape representation learning and generation," in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. IEEE, Oct. 2019, p. 7212–7221. [Online]. Available: http://dx.doi.org/10.1109/ICCV.2019.00731

[186] H. Zhang, S. Cheng, C. E. Amm, and J. Kim, "Efficient pooling operator for 3d morphable models," *IEEE Transactions on Visualization and Computer Graphics*, vol. 30, no. 7, pp. 4225–4233, 2024.

[187] Y. Zheng, H. Fu, O. K.-C. Au, and C.-L. Tai, "Bilateral normal filtering for mesh denoising," *IEEE Transactions on Visualization and Computer Graphics*, vol. 17, no. 10, pp. 1521–1530, 2011.

[188] K. Crane, M. Desbrun, and P. Schröder, "Trivial connections on discrete surfaces," *Computer Graphics Forum*, vol. 29, no. 5, pp. 1525–1533, 2010. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1467-8659.2010.01761.x