Neural QSLIM for Mesh Autoencoders

Chapter · September 2025

DOI: 10.1007/978-3-032-04555-3_5

5 authors, including:

Haoliang Zhang
University of Oklahoma
4 PUBLICATIONS **60** CITATIONS

SEE PROFILE

Neural QSLIM for Mesh Autoencoders

Haoliang Zhang[1*] ⋆, Xintong Li[2*], Jonghoon Kim[3], Samuel Cheng[1], and Christian El Amm[1]

[1] University of Oklahoma, Norman, OK, USA
[2] Beijing University of Posts and Telecommunications, Beijing, China
[3] Chungnam National University, Daejeon, Republic of Korea

Abstract. 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.

Keywords: Mesh autoencoder · Geometry · QSLIM.

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 [1, 2, 3, 4] due to irregular sampling and complex connectivity. Ranjan et al. [5] 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 [6, 7, 8, 9, 10, 11].

Classical mesh autoencoders rely on pointwise loss, which requires the input and output shapes to have the same number of vertices. While MeshCNN [12] and

⋆ ✉ Corresponding author: mars_zhang@ou.edu
* Contributed equally

Neural Subdivision [13] 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 [14] 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) [15, 16] 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 [17, 18]), 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

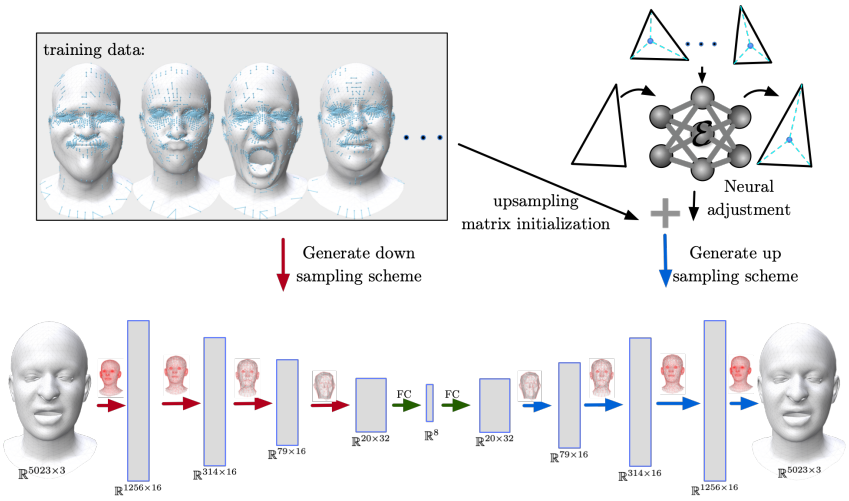

Fig. 1. Unlike standard mesh autoencoders that build down/up matrices from a single mesh, our method employs a novel coarsening scheme and neural unpooling layer on a fixed topology mesh dataset. The resulting transformation matrices or hierarchies can be seamlessly integrated into mesh generative models without altering the user's network architecture.

in Fig. 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.
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.

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.

2.1 Geometric deep learning

CoMA [5] 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 [6, 7, 19, 9, 20, 21, 22, 23] improved mesh convolution networks while inheriting the down-sampling and up-sampling operations. Neural3DMM [6] and SpiralNet++ [7] replace spectral convolution layers with operators that convolve along a fixed spiral serialization [24] of neighboring vertices, outperforming previous approaches. Alternative implementations, such as [14], define a shared weight basis kernel at every vertex location but come at the cost of increased parameters. MeshCNN [12] 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.

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" [25]. However, they are incapable of recuperating lost details or semantic features. Traditional methods like subdivision methods [26, 27, 28] process the mesh by combinatorial update (splitting faces, adding vertices, or/and fipping edges [29]) and vertex smoothing. On the other hand, contemporary non-linear neural subdivision methods [30] and subdivision-based networks (SubdivNet) [31] 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 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.

3.1 Build transformation matrix

Although the dataset shares the same topology, the down-sampling results obtained by QSLIM [15] 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.2). The most common edges are then used to construct the down-sampling transformation matrix.

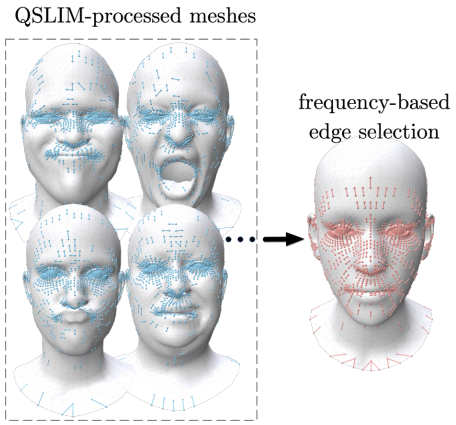

Fig. 2. QSLIM is applied to the meshes, and edges are selected based on their frequency. Red edges show those contracted during down-sampling.

Vertices discarded during down-sampling are mapped onto the surface of the down-sampled mesh using barycentric coordinates. Following [5], 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 [5], 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.

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 [27]. 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. [32] introduced free-boundary conformal parameterizations for a triangular mesh. In the context of an edge collapse, this conformal parameterization [32] can be employed to map the vertices of the 1-ring of the edge into its local UV space, as illustrated in Fig. 3. This UV parameterization serves as a verification tool. A critical aspect to note is the shared boundary vertices between

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);

the contracted point post-edge collapse and the collapsed edge. Given the UV boundary constraints, the post-collapse vertex retains identical UV boundaries.

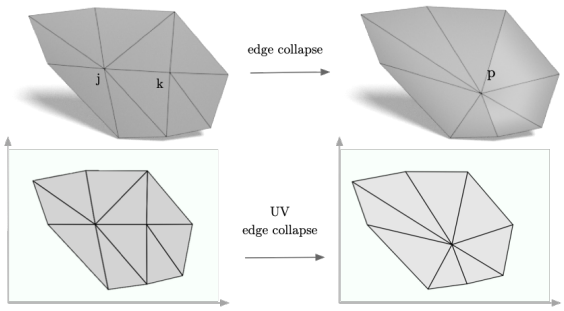

Fig. 3. Utilizing conformal parameterization to map the 1-ring of a collapsed edge to the UV space. Notably, the vertices post and pre-collapse maintain identical boundary vertices.

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.4 illustrates an external vertex 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.

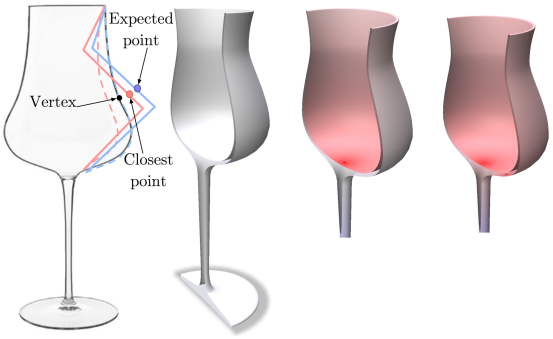

Fig. 4. Heat diffusion illustrated on a wine glass. Through local mapping, inside-out projection is avoided, resulting in an outward heat spread on the surface (center right), as opposed to the incorrect heat flow caused by closest point projection (right).

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 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.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.

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. 6), and $f(v)$ denote the actual barycentric coordinates. We measure the per-vertex loss with the $\mathcal{L} = ||f(v) - \varepsilon(v)||_2$.

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. 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

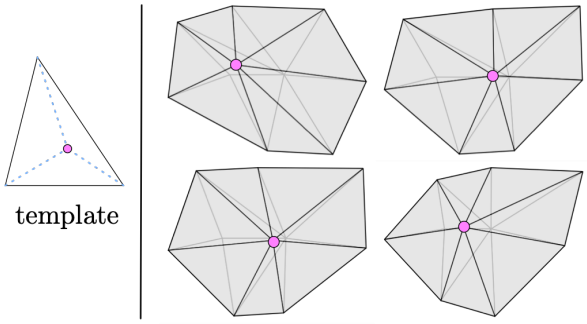

Fig. 5. Barycentric coordinates resulting from mapping discarded vertices to the closest triangle are depicted. Despite contracting the same edge to the same end, the projection point differs depending on the given geometry. Relying solely on the template representation leads to an inaccurate up-sampling matrix.

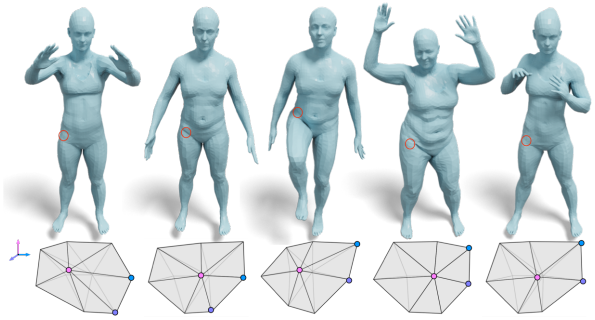

Fig. 6. Registered data allows us to order the three vertexes of the triangle canonically. We feed vertex features into multi-layer perceptrons (MLP).

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 upsampling 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 coor-

dinates 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.

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 [33]. We use Relu activation [34], and the ADAM optimizer [35] with a learning rate of 0.002. We implemented our technique using the Spectra [36], 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. 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.

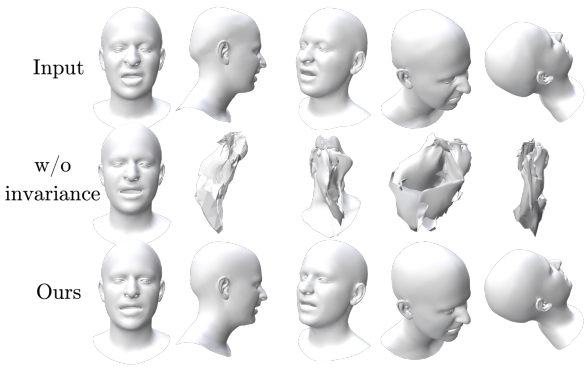

Fig. 7. Employing local coordinates embedded within each mesh as both inputs and outputs, our methodology achieves invariance to rigid motions. This adaptation yields a marked enhancement in quality relative to preceding methods that lack such invariance.

Fig. 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.

We further evaluated successive local mapping on mesh generative tasks. For registered data such as CoMA [5] or FAUST [37], 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. 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.

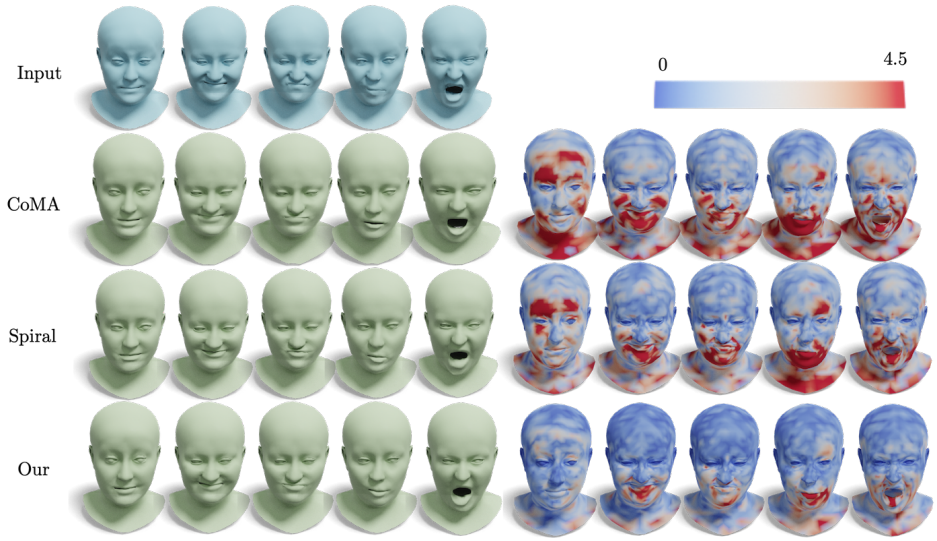

Fig. 9. Qualitative assessment of reconstruction outcomes. Each vertex's Euclidean error is color-coded, with the error values capped at 4.5 millimeters. Warmer hues denote larger errors.

Our method for pooling and unpooling is highly compatible with numerous standard mesh processing pipelines. It is engineered to facilitate local bijective mapping while meticulously 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. 9 and Fig. 10. A comprehensive list of mean errors, accompanied by standard deviations, is presented in Table 1 for an exhaustive comparative analysis.

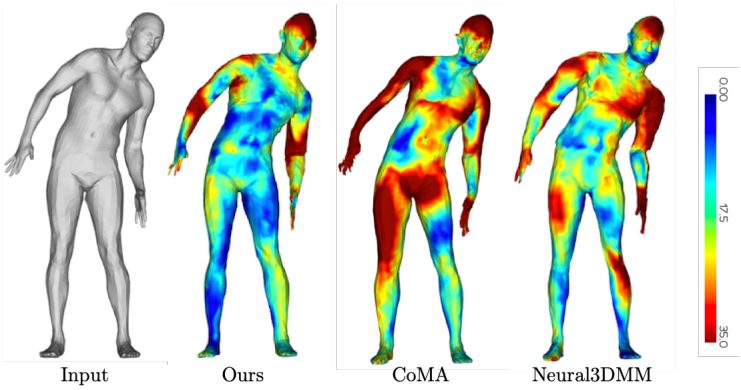

Fig. 10. Colour coding of the per vertex euclidean error of the reconstructions. The error values are saturated at 35 (millimeters)

Table 1. Comparison with CoMA and Neural3DMM. Errors are in millimeters. Simplified Neural3DMM with filter size [16,16,16,32].

	Mean Error	#Parameters
Ours	$\mathbf{0.541 \pm 0.588}$	37,563
Neural3DMM [6]	0.804 ± 0.877	48,067
CoMA [5]	0.845 ± 0.994	33,856
PCA	1.639 ± 1.638	120,552

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 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.

Bibliography

[1] Timur Bagautdinov, Chenglei Wu, Jason Saragih, Pascal Fua, and Yaser Sheikh. Modeling facial geometry using compositional vaes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3877–3886, 2018.

[2] Ayan Sinha, Jing Bai, and Karthik Ramani. Deep learning 3d shape surfaces using geometry images. In *European conference on computer vision*, pages 223–240. Springer, 2016.

[3] Ayan Sinha, Asim Unmesh, Qixing Huang, and Karthik Ramani. Surfnet: Generating 3d shape surfaces using deep residual networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6040–6049, 2017.

[4] Haggai Maron, Meirav Galun, Noam Aigerman, Miri Trope, Nadav Dym, Ersin Yumer, Vladimir G Kim, and Yaron Lipman. Convolutional neural networks on surfaces via seamless toric covers. *ACM Trans. Graph.*, 36(4): 71–1, 2017.

[5] Anurag Ranjan, Timo Bolkart, Soubhik Sanyal, and Michael J. Black. Generating 3D faces using convolutional mesh autoencoders. In *European Conference on Computer Vision (ECCV)*, pages 725–741, 2018.

[6] Giorgos Bouritsas, Sergiy Bokhnyak, Stylianos Ploumpis, Michael Bronstein, and Stefanos 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*, pages 7213–7222, 2019.

[7] Shunwang Gong, Lei Chen, Michael Bronstein, and Stefanos Zafeiriou. Spiralnet++: A fast and highly efficient mesh convolution operator. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 0–0, 2019.

[8] Edgar Tretschk, Ayush Tewari, Michael Zollhöfer, Vladislav Golyanik, and Christian Theobalt. Demea: Deep mesh autoencoders for non-rigidly deforming objects. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part IV 16*, pages 601–617. Springer, 2020.

[9] Zhixiang Chen and Tae-Kyun Kim. Learning feature aggregation for deep 3d morphable models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13164–13173, 2021.

[10] Federico Pichi, Beatriz Moya, and Jan S. Hesthaven. A graph convolutional autoencoder approach to model order reduction for parametrized pdes. *Journal of Computational Physics*, 501:112762, 2024. ISSN 0021-9991. https://doi.org/https://doi.org/10.1016/j.jcp.2024.112762. URL https://www.sciencedirect.com/science/article/pii/S0021999124000111.

[11] Amani Almalki and Longin Jan 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, pages 7820–7830, 2024.

[12] Rana Hanocka, Amir Hertz, Noa Fish, Raja Giryes, Shachar Fleishman, and Daniel Cohen-Or. Meshcnn: A network with an edge. *ACM Transactions on Graphics (TOG)*, 38(4):90:1–90:12, 2019.

[13] Hsueh-Ti Derek Liu, Vladimir G. Kim, Siddhartha Chaudhuri, Noam Aigerman, and Alec Jacobson. Neural subdivision. *ACM Trans. Graph.*, 39(4), August 2020. ISSN 0730-0301. https://doi.org/10.1145/3386569.3392418. URL https://doi.org/10.1145/3386569.3392418.

[14] Yi Zhou, Chenglei Wu, Zimo Li, Chen Cao, Yuting Ye, Jason Saragih, Hao Li, and Yaser Sheikh. Fully convolutional mesh autoencoder using efficient spatially varying kernels. In *Advances in neural information processing systems*, 2020.

[15] Michael Garland and Paul S Heckbert. Surface simplification using quadric error metrics. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 209–216, 1997.

[16] Jonathan Cohen, Marc Olano, and Dinesh Manocha. Appearance-preserving simplification. In *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '98, page 115–122, New York, NY, USA, 1998. Association for Computing Machinery. ISBN 0897919998. https://doi.org/10.1145/280814.280832. URL https://doi.org/10.1145/280814.280832.

[17] 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*, IJCAI'77, page 659–663, San Francisco, CA, USA, 1977. Morgan Kaufmann Publishers Inc.

[18] Haoqiang Fan, Hao Su, and Leonidas 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*, pages 605–613, 2017.

[19] Zhongpai Gao, Guangtao Zhai, Juyong Zhang, Junchi Yan, Yiyan Yang, and Xiaokang Yang. Learning local neighboring structure for robust 3d shape representation. In *AAAI Conference on Artificial Intelligence*, 2020. URL https://api.semanticscholar.org/CorpusID:219402126.

[20] Hamza Bouzid and Lahoucine Ballihi. Spatr: Mocap 3d human action recognition based on spiral auto-encoder and transformer network. *Computer Vision and Image Understanding*, 241:103974, 2024.

[21] Haoliang Zhang, Samuel Cheng, Christian El Amm, and Jonghoon Kim. Efficient pooling operator for 3d morphable models. *IEEE Transactions on Visualization and Computer Graphics*, 30(7):4225–4233, 2024. https://doi.org/10.1109/TVCG.2023.3255820.

[22] Haoliang Zhang, Woonki Na, and Jonghoon Kim. State-of-charge estimation of the lithium-ion battery using neural network based on an improved thevenin circuit model. In *2018 IEEE Transporta-*

tion Electrification Conference and Expo (ITEC), pages 342–346, 2018. https://doi.org/10.1109/ITEC.2018.8450162.

[23] Haoliang Zhang, Wei Tang, Woonki Na, Pyeong-Yeon Lee, and Jonghoon Kim. Implementation of generative adversarial network-cls combined with bidirectional long short-term memory for lithium-ion battery state prediction. *Journal of Energy Storage*, 31:101489, 2020. https://doi.org/https://doi.org/10.1016/j.est.2020.101489. URL https://www.sciencedirect.com/science/article/pii/S2352152X19306930.

[24] Isaak Lim, Alexander Dielen, Marcel Campen, and Leif Kobbelt. A simple approach to intrinsic correspondence learning on unstructured 3d meshes. In *Proceedings of the European conference on computer vision (ECCV) workshops*, pages 0–0, 2018.

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

[26] Daniel Doo and Malcolm Sabin. Behaviour of recursive division surfaces near extraordinary points. *Computer-Aided Design*, 10(6):356–360, 1978.

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

[28] Edwin Catmull and James Clark. Recursively generated b-spline surfaces on arbitrary topological meshes. In *Seminal graphics: pioneering efforts that shaped the field*, pages 183–188. ACM Press/Addison-Wesley Publishing Co., 1998.

[29] Leif Kobbelt. 3-subdivision. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*, page 103–112, USA, 2000. ACM Press/Addison-Wesley Publishing Co. ISBN 1581132085.

[30] Hsueh-Ti Derek Liu, Vladimir G. Kim, Siddhartha Chaudhuri, Noam Aigerman, and Alec Jacobson. Neural subdivision. *ACM Trans. Graph.*, 39(4), 2020. ISSN 0730-0301.

[31] Shi-Min Hu, Zheng-Ning Liu, Meng-Hao Guo, Jun-Xiong Cai, Jiahui Huang, Tai-Jiang Mu, and Ralph R Martin. Subdivision-based mesh convolution networks. *ACM Transactions on Graphics (TOG)*, 41(3):1–16, 2022.

[32] Patrick Mullen, Yiying Tong, Pierre Alliez, and Mathieu Desbrun. Spectral conformal parameterization. In *Computer Graphics Forum*, volume 27, pages 1487–1494. Wiley Online Library, 2008.

[33] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

[34] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Icml*, 2010.

[35] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017. URL https://arxiv.org/abs/1412.6980.

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

[37] Federica Bogo, Javier Romero, Gerard Pons-Moll, and Michael J. Black. Dynamic FAUST: Registering human bodies in motion. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, July 2017.