Efficient Pooling Operator for 3D Morphable Models

Haoliang Zhang, Samuel Cheng, *Senior Member, IEEE*, Christian El Amm, and Jonghoon Kim, *Senior Member, IEEE*

***Abstract*—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.**

***Index Terms*—Latent representation, mesh reconstruction, morphable model.**

I. 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.

Manuscript received 2 June 2022; revised 7 March 2023; accepted 7 March 2023. Date of publication 13 March 2023; date of current version 26 June 2024. Recommended for acceptance by K. Hormann. *(Corresponding author: Jonghoon Kim.)*

Haoliang Zhang, Samuel Cheng, and Christian El Amm are with the The University of Oklahoma, Norman, OK 73019 USA (e-mail: mars_zhang@ou.edu; samuel.cheng@ou.edu; christianamm@hotmail.com).

Jonghoon Kim is with the Chungnam National University, Daejeon 34134, Republic of Korea (e-mail: qwzxas@hanmail.net).

This article has supplementary downloadable material available at https://doi.org/10.1109/TVCG.2023.3255820, provided by the authors.

Digital Object Identifier 10.1109/TVCG.2023.3255820

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.

II. 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.

A. 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 III, 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.

B. 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 A, available online). 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 (Fig. 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.

C. 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 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 [20], [21], [43], [44], [45], [46]. These methods consider pair vertex features to compute attention weights with different types of kernels or learnable functions. Some methods [25], [47] 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], [22], [50].

III. Method

In this section, we introduce the proposed pooling and unpooling operations. The pooling layers in current autoencoder models [20], [22], [50] 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).

A. Mesh Sampling

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

$$Q^v(\mathrm{v}) = \sum_{\mathrm{f} \in \mathrm{v}} \mathrm{Q}_\mathrm{g}^\mathrm{f}(\mathrm{v}) \tag{1}$$

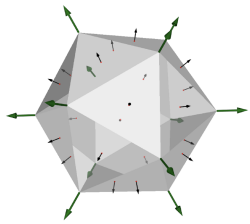


Fig. 1. Vertex and face normals.

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 $Q^{v_1}(\mathrm{v}) + \mathrm{Q}^{\mathrm{v}_2}(\mathrm{v})$. First, we modify the QEM function, as expressed in Eq. (2); each vertex v of the mesh is assigned with the sum of quadrics weighted by its adjacent faces area respectively.

$$Q^v(\mathrm{v}) = \sum_{\mathrm{f} \in \mathrm{v}} \mathrm{area(f)} \mathrm{Q}_\mathrm{g}^\mathrm{f}(\mathrm{v}) \tag{2}$$

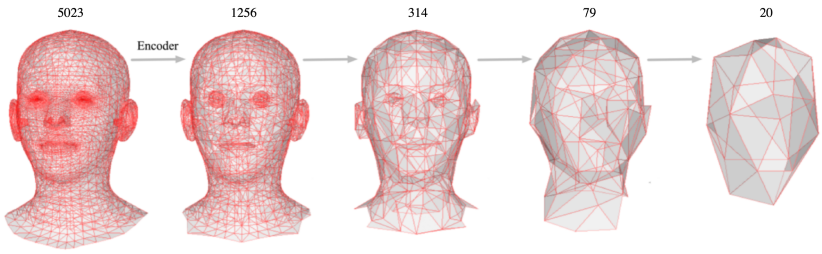
A vertex normal is a directional vector associated with a vertex [51] (see Fig. 1). It also consists of the curvature information of the shape (Appendix A, available in the online supplemental material). The contribution of faces normal to a vertex normal is the ratio of the corner angle in which the 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

$$Q^v\left(\mathrm{v} = \begin{pmatrix} p \\ vn \end{pmatrix}\right) = \sum_{f \in v} area(f) Q_g^f(\mathrm{v}) + \lambda_{\mathrm{vn}} \mathrm{Q}_{\mathrm{vn}}^\mathrm{f}(\mathrm{v}) \tag{3}$$

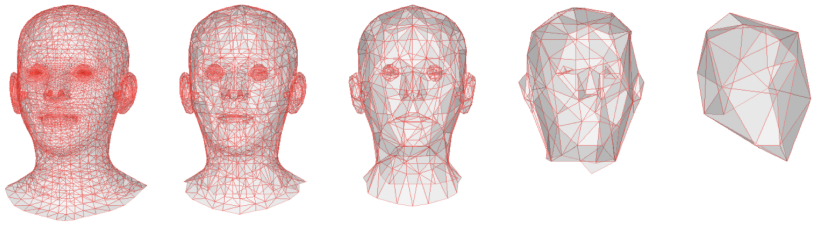
where $Q_g^f(\mathrm{v})$ is the same as obtained using the QEM methods but weighted by the adjacent faces area, and $Q_{vn}^f(v) = (g_{vn}^T p + d_{vn} - vn)^2$ is the squared normal deviation. (g_{vn}, d_{vn}) are computed using Eq. (4)

$$\begin{pmatrix} p_1^T & 1 \\ p_2^T & 1 \\ p_3^T & 1 \\ n^T & 0 \end{pmatrix} \begin{pmatrix} g_j \\ \\ d_j \end{pmatrix} = \begin{pmatrix} vn_1 \\ vn_2 \\ vn_3 \\ 0 \end{pmatrix} \tag{4}$$

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. Fig. 2 shows a comparative analysis, in which the sample rate is four in each layer. As shown in Fig. 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 Fig. 2a; our graph



(a) Our sampling process



(b) Quadric mesh sampling

Fig. 2. Network architecture and comparison for mesh sampling.

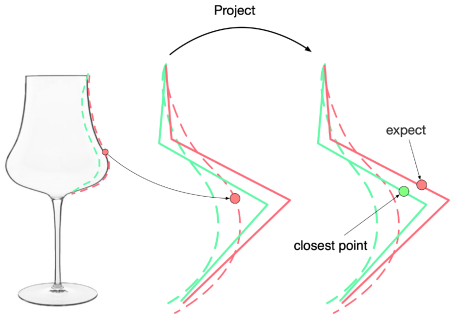


Fig. 3. Closest point projection.

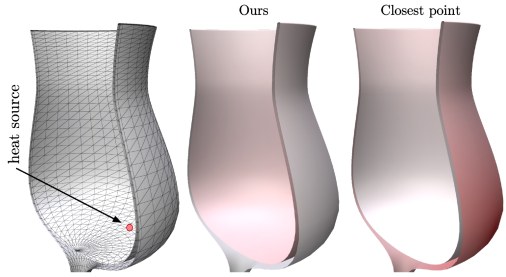


Fig. 4. Heat source is placed inside the wine glass. The closest projection caused the error, where the heat does not diffuse along the surface.

sampling scheme maintains the head shape and detail in the multiscale hierarchy structure.

B. 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], [22], [50] 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.

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 Fig. 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 Fig. 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.

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 IV-D.

IV. 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 IV-D. 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.

A. 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].

B. 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 [20], [22], 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.

C. 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].

We compared our model with two high baseline models: CoMA [20] and Neural3DMM [22]. For both datasets, as clearly illustrated in Fig. 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 Figs. 5 and 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 reconstructions. We use an eight-dimensional latent space to encode the meshes in all models.

D. 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 I, 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 Fig. 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. 8, the closest point projection methods exhibited significant

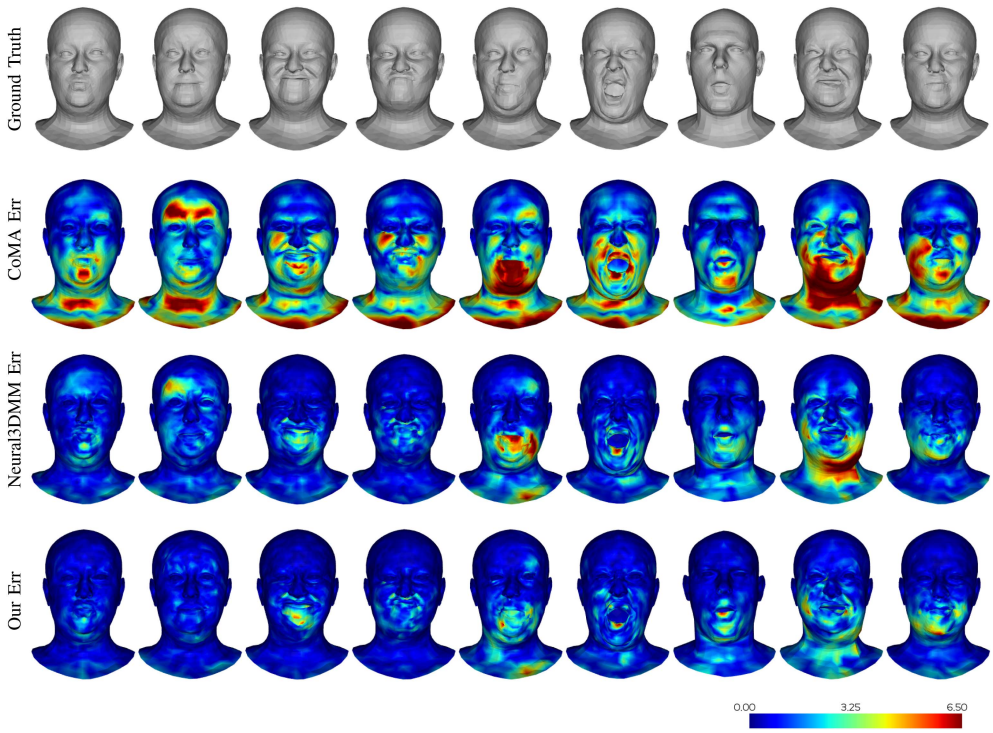


Fig. 5. Qualitative comparison results for interpolation experiment. Colour coding of the per vertex euclidean error of the reconstructions. The error values are saturated at 6.5 (millimeters). Hot colors represent large errors.

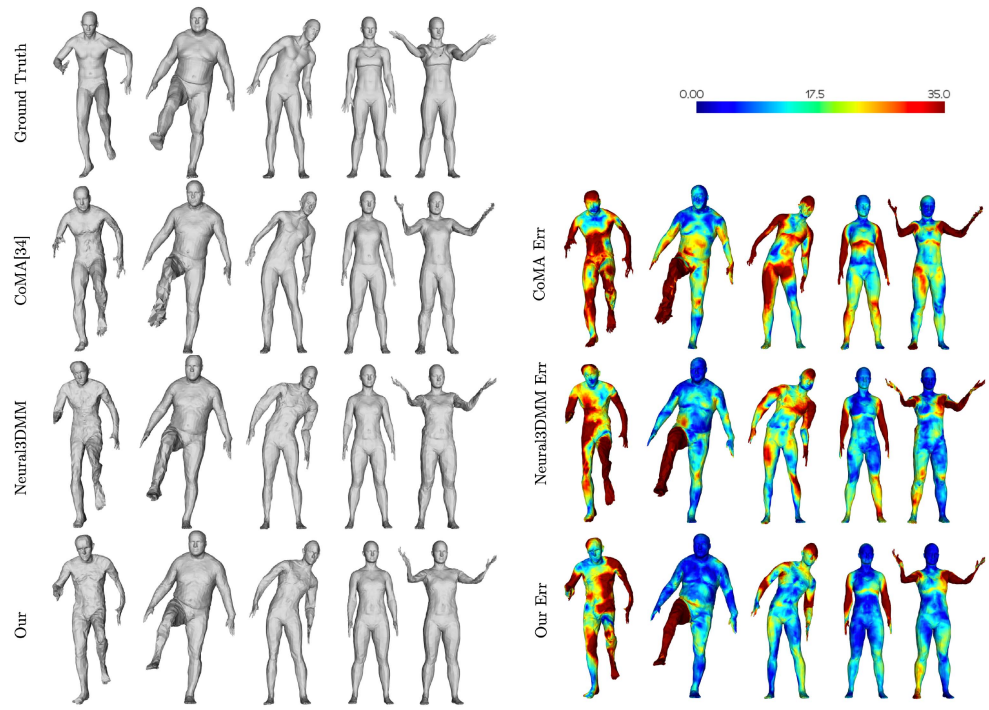


Fig. 6. Colour coding of the per vertex euclidean error of the reconstructions. Top row is ground truth. The error values are saturated at 35 (millimeters).

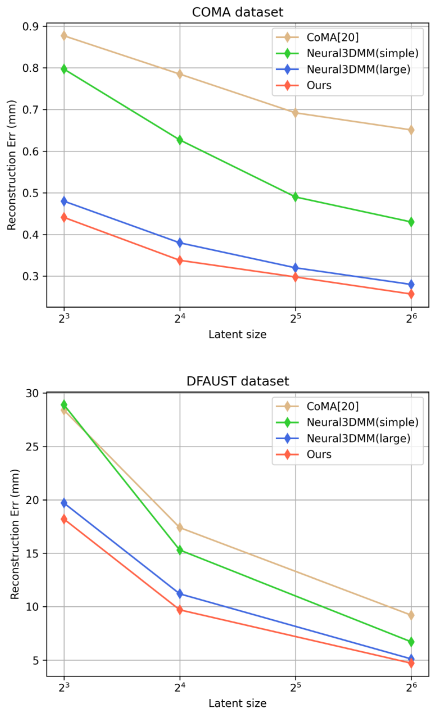


Fig. 7. Reconstruction errors of different methods.

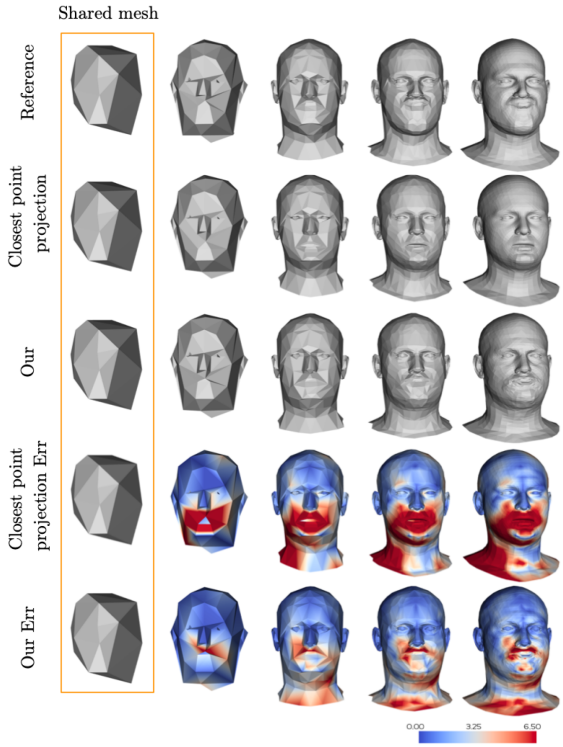


Fig. 8. Visualization of unpooling meshes with error: The first row lists the pooling meshes in reverse order. The two unpooling methods share the same initial coarse mesh. The second and third rows present the results of the closest point projection [20], [22] and our proposed unpooling method, respectively. The error values are saturated at 6.5 (millimeters).

TABLE I
COMPARISON WITH ORIGINAL POOLING AND UNPOOLING: INTERPOLATION EXPERIMENTS. 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)	$\mathbf{0.749 \pm 0.888}$	0.443

The best results are highlighted in bold.

template overfit. However, the proposed method provided an effective projection during the unpooling of the layers.

E. Comparison of Network Hyperparameters

In this section, we compare different network hyperparameters with our operators. Based on the architecture defined in Section IV-B, 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 [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 II 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 IV-B, Table II lists the architectural designs, the errors, and the parameter count.

TABLE II
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]. ERRORS ARE IN MILLIMETERS

	Mean Error	#Parameters
Ours ([64,64,64,128])	$\mathbf{0.441 \pm 0.558}$	489,995
Ours ([32,32,32,64])	$\mathbf{0.547 \pm 0.641}$	134,411
Ours ([16,16,32,32])	$\mathbf{0.692 \pm 0.804}$	53,419
Ours ([16,16,16,32])	$\mathbf{0.745 \pm 0.874}$	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

The best results per architecture appear in bold.

As listed in Table II, 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.

V. 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.

References

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

[2] B. Amberg, R. Knothe, and T. Vetter, "Expression invariant 3D face recognition with a morphable model," in *Proc. IEEE 8th Int. Conf. Autom. Face Gesture Recognit.*, 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 et al., "MoFA: Model-based deep convolutional face autoencoder for unsupervised monocular reconstruction," in *Proc. IEEE Int. Conf. Comput. Vis. 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 *Proc. ACM SIGGRAPH*, 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, doi: 10.1145/2816795.2818056.

[7] S. Bouaziz, Y. Wang, and M. Pauly, "Online modeling for realtime facial animation," *ACM Trans. Graph.*, vol. 32, no. 4, pp. 1–10, 2013.

[8] H. Li, T. Weise, and M. Pauly, "Example-based facial rigging," *ACM Trans. Graph.*, vol. 29, no. 4, pp. 1–6, 2010.

[9] V. Blanz and T. Vetter, "A morphable model for the synthesis of 3D faces," in *Proc. 26th Annu. Conf. Comput. Graph. Interactive Techn.*, 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 *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 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 *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2014, pp. 1701–1708.

[12] Y. Li, R. Bu, M. Sun, W. Wu, X. Di, and B. Chen, "PointCNN: Convolution on X-transformed points," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 820–830.

[13] Y. Zhou et al., "Fully convolutional mesh autoencoder using efficient spatially varying kernels," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, pp. 9251–9262.

[14] T. Bagautdinov, C. Wu, J. Saragih, P. Fua, and Y. Sheikh, "Modeling facial geometry using compositional VAEs," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 3877–3886.

[15] A. Sinha, J. Bai, and K. Ramani, "Deep learning 3D shape surfaces using geometry images," in *Proc. Eur. Conf. Comput. Vis.*, 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 *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 6040–6049.

[17] H. Maron et al., "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 Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 1, pp. 221–231,Jan. 2013.

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

[20] A. Ranjan, T. Bolkart, S. Sanyal, and M. J. Black, "Generating 3D faces using convolutional mesh autoencoders," in *Proc. Eur. Conf. Comput. Vis.*, 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 *Proc. Neural Inf. Process. Syst.*, Red Hook, NY, USA, 2016, pp. 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 *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 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 *Proc. Eur. Conf. Comput. Vis. Workshops*, 2018, pp. 349–362.

[24] E. Tretschk, A. Tewari, M. Zollhöfer, V. Golyanik, and C. Theobalt, "DEMEA: Deep mesh autoencoders for non-rigidly deforming objects," in *Proc. Eur. Conf. Comput. Vis.*, Springer, 2020, pp. 601–617.

[25] O. Litany, A. Bronstein, M. Bronstein, and A. Makadia, "Deformable shape completion with graph convolutional autoencoders," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 1886–1895.

[26] F. Bogo, J. Romero, G. Pons-Moll, and M. J. Black, "Dynamic faust: Registering human bodies in motion," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 6233–6242.

[27] J. Malik et al., "DeepHPS: End-to-end estimation of 3D hand pose and shape by learning from synthetic depth," in *Proc. IEEE Int. Conf. 3D Vis.*, 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 Trans. Graph.*, 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," 2022, *arXiv:2201.02610*.

[30] M. Garland and P. S. Heckbert, "Surface simplification using quadric error metrics," in *Proc. 24th Annu. Conf. Comput. Graph. Interactive Techn.*, 1997, pp. 209–216.

[31] H. Hoppe, "New quadric metric for simplifying meshes with appearance attributes," in *Proc. IEEE Conf. Vis.*, 1999, pp. 59–510.

[32] M. Garland and P. S. Heckbert, "Simplifying surfaces with color and texture using quadric error metrics," in *Proc. IEEE Conf. Vis.*, 1998, pp. 263–269.

[33] L. Yao, S. Huang, H. Xu, and P. Li, "Quadratic error metric mesh simplification algorithm based on discrete curvature," *Math. Problems Eng.*, vol. 2015, pp. 1–7, 2015.

[34] S. Hahner and J. Garcke, "Mesh convolutional autoencoder for semi-regular meshes of different sizes," in *Proc. IEEE/CVF Winter Conf. Appl. Comput. Vis.*, 2022, pp. 885–894.

[35] C. Tang, X. Sun, A. Gomes, J. Wallner, and H. Pottmann, "Form-finding with polyhedral meshes made simple," *ACM Trans. Graph.*, 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 *Proc. ACM SIGGRAPH Papers*, 2006, pp. 681–689.

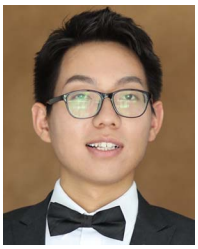
[37] R. Hanocka, A. Hertz, N. Fish, R. Giryes, S. Fleishman, and D. Cohen-Or, "MeshCNN: A network with an edge," *ACM Trans. Graph.*, vol. 38, no. 4, pp. 1–12, 2019.

[38] F. R. Chung and F. C. Graham, *Spectral Graph Theory*.Providence, RI, USA: American Mathematical Soc., 1997.

[39] D. K. Hammond, P. Vandergheynst, and R. Gribonval, "Wavelets on graphs via spectral graph theory," *Appl. Comput. Harmon. Anal.*, 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," *J. Mach. Learn. Res.*, vol. 12, no. 9, pp. 2539–2561, 2011.

[41] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 1025–1035.
[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 *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 5115–5124.
[43] J. Atwood and D. Towsley, "Diffusion-convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 1993–2001.
[44] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," 2016, *arXiv:1609.02907*.
[45] D. Boscaini, J. Masci, E. Rodoià, and M. Bronstein, "Learning shape correspondence with anisotropic convolutional neural networks," in *Proc. 30th Int. Conf. Neural Inf. Process. Syst.*, 2016, pp. 3197–3205.
[46] J. Masci, D. Boscaini, M. Bronstein, and P. Vandergheynst, "Geodesic convolutional neural networks on Riemannian manifolds," in *Proc. IEEE Int. Conf. Comput. Vis. 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 *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 5661–5670.
[48] N. Verma, E. Boyer, and J. Verbeek, "FeastNet: Feature-steered graph convolutions for 3D shape analysis," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 2598–2606.
[49] I. S. Dhillon, Y. Guan, and B. Kulis, "Weighted graph cuts without eigenvectors a multilevel approach," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 11, pp. 1944–1957,Nov. 2007.
[50] Z. Chen and T.-K. Kim, "Learning feature aggregation for deep 3D morphable models," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 13 164–13 173.
[51] H. Gouraud, "Continuous shading of curved surfaces," *IEEE Trans. Comput.*, vol. C-20, no. 6, pp. 623–629,Jun. 1971.
[52] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*.



Haoliang Zhang is currently working toward the PhD degree in Electrical and Computer Engineering, University of Oklahoma. His research interests include machine learning and discrete differential geometry.

Samuel Cheng (Senior Member, IEEE) received the BS degree in electrical and electronic engineering from the University of Hong Kong, and the MPhil degree in physics and the MS degree in electrical engineering from the Hong Kong University of Science and Technology and the University of Hawaii, Honolulu, respectively, and the PhD degree in electrical engineering from Texas A&M University in 2004. He worked with Microsoft Asia, China, and Panasonic Technologies Company, New Jersey, in the areas of texture compression and digital watermarking during the summers of 2000 and 2001. In 2004, he joined Advanced Digital Imaging Research, a research company based near Houston, Texas, as a research engineer to perform biomedical imaging research and was promoted to senior research engineer the next year. Since 2006, he joined the School of Electrical and Computer Engineering with the University of Oklahoma and is currently an associate professor and the director of OU-Tulsa Lab of Information and Image Processing. He has been awarded six US patents in miscellaneous areas of signal processing and is the co-recipient of the 2007 IEEE Signal Processing Magazine Best Paper Award. He is a member of ACM.

Christian El Amm is a professor with the Department of Surgery at the University of Oklahoma. He specializes in plastic and reconstructive surgery, and pediatric plastic and craniofacial surgery. His areas of expertise include: cleft lip and palate, craniosynostosis, Pierre Robin Sequence, midface hypoplasia, fat grafting and liposuction, rhinoplasty, secondary rhinoplasty, facial rejuvenation, body contouring, and breast augmentation or reconstruction. He has practiced at OU Health since 2005 and has received recognition by Castle Connolly Medical Ltd. as a top doctor in 2016 and 2017.

Jonghoon Kim (Senior Member, IEEE) received the BS degree from Chungnam National University, Daejeon, South Korea, in 2005 and the PhD degree in electrical engineering and computer science from Seoul National University, Seoul, South Korea, in 2012. From 2012 to 2013, he was a senior research engineer with the Energy Storage System Development Group, Energy Solution Division, Samsung SDI, Cheonan, South Korea. From 2013 to 2016, he was an assistant professor with the Department of Electrical Engineering, Chosun University, Gwangju, South Korea. Since 2016, he has been an associate professor with the Department of Electrical Engineering, Chungnam National University. Since 2018, he has been an adjunct professor with Eco-Friendly Smart Car Research Center, Korea Advanced Institute of Science and Technology, Daejeon, South Korea. His main research interests include battery management system (SOx estimation, prediction algorithms, equalization, and screening), next-generation batteries (lithium-air battery, lithium-sulfur battery, vanadium redox flow battery, solid-state battery, aluminum-ion battery, and sodium-ion/sulfur batteries), xEV retired battery (second-use), energy storage system (ESS), fault diagnosis, thermal management, artificial intelligence, power electronics circuits, and fuel cell system (ripple current analysis and energy management system).