Generating 3D Faces Using Convolutional Mesh Autoencoders

Anurag Ranjan(✉), Timo Bolkart, Soubhik Sanyal, and Michael J. Black

Max Planck Institute for Intelligent Systems, Tübingen, Germany
{aranjan,tbolkart,ssanyal,black}@tuebingen.mpg.de

Abstract. Learned 3D representations of human faces are useful for computer vision problems such as 3D face tracking and reconstruction from images, as well as graphics applications such as character generation and animation. Traditional models learn a latent representation of a face using linear subspaces or higher-order tensor generalizations. Due to this linearity, they can not capture extreme deformations and non-linear expressions. To address this, we introduce a versatile model that learns a non-linear representation of a face using spectral convolutions on a mesh surface. We introduce mesh sampling operations that enable a hierarchical mesh representation that captures non-linear variations in shape and expression at multiple scales within the model. In a variational setting, our model samples diverse realistic 3D faces from a multivariate Gaussian distribution. Our training data consists of 20,466 meshes of extreme expressions captured over 12 different subjects. Despite limited training data, our trained model outperforms state-of-the-art face models with 50% lower reconstruction error, while using 75% fewer parameters. We show that, replacing the expression space of an existing state-of-the-art face model with our model, achieves a lower reconstruction error. Our data, model and code are available at http://coma.is.tue.mpg.de/.

1 Introduction

The human face is highly variable in shape as it is affected by many factors such as age, sex, ethnicity, etc., and deforms significantly with expressions. The existing state of the art 3D face representations mostly use linear transformations [28,41,42] or higher-order tensor generalizations [12,14,46]. These 3D face models have several applications including face recognition [40], generating and animating faces [28] and monocular 3D face reconstruction [44]. Since these models are linear, they do not capture the non-linear deformations due to extreme facial expressions. These expressions are crucial to capture the realism of a 3D face.

Meanwhile, convolutional neural networks (CNNs) have emerged as rich models for generating images [22,35], audio [34], etc. One of the reasons for

Electronic supplementary material The online version of this chapter (https://doi.org/10.1007/978-3-030-01219-9_43) contains supplementary material, which is available to authorized users.

V. Ferrari et al. (Eds.): ECCV 2018, LNCS 11207, pp. 725–741, 2018.
https://doi.org/10.1007/978-3-030-01219-9_43

their success is attributed to the multi-scale hierarchical structure of CNNs that allows them to learn translational-invariant localized features. Recent works have explored volumetric convolutions [8] for 3D representations. However, volumetric operations require a lot of memory and have been limited to low resolution 3D volumes. Modeling convolutions on 3D meshes can be memory efficient and allows for processing high resolution 3D structures. However, CNNs have mostly been successful in Euclidean domains with grid-based structured data and the generalization of CNNs to meshes is not trivial. Extending CNNs to graph structures and meshes has only recently drawn significant attention [10,11,17]. Hierarchical operations in CNNs such as max-pooling and upsampling have not been adapted to meshes. Moreover, training CNNs on 3D facial data is challenging due to the limited size of current 3D datasets. Existing large scale datasets [14,16,38,49,50] do not contain high resolution extreme facial expressions.

To address these problems, we introduce a Convolutional Mesh Autoencoder (CoMA) with novel mesh sampling operations, which preserve the topological structure of the mesh features at different scales in a neural network. We follow the work of Defferrard et al. [17] on generalizing the convolution on graphs using fast Chebyshev filters, and use their formulation for convolving over our facial mesh. We perform spectral decomposition of meshes and apply convolutions directly in frequency space. This makes convolutions memory efficient and feasible to process high resolution meshes. We combine the convolutions and sampling operations to construct our model in the form of a Convolutional Mesh Autoencoder. We show that CoMA performs much better than state of the art face models at capturing highly non-linear extreme facial expressions with fewer model parameters. Having fewer parameters in our model makes it more compact, and easier to train. This reduction in parameters is attributed to the locally invariant convolutional filters that can be shared over the mesh surface.

We address the problem of data limitation by capturing 20,466 high resolution meshes with extreme facial expressions in a multi-camera active stereo system. Our dataset spans 12 subjects performing 12 different expressions. The expressions are chosen to be complex and asymmetric, with significant deformation in the facial tissue.

In summary, our work introduces a representation that models variations on the mesh surface using a hierarchical multi-scale approach and can generalize to other 3D mesh processing applications. Our main contributions are: (1) we introduce a Convolutional Mesh Autoencoder consisting of mesh downsampling and mesh upsampling layers with fast localized convolutional filters defined on the mesh surface; (2) we show that our model accurately represents 3D faces in a low-dimensional latent space performing 50% better than a PCA model that is used in state of the art face models such as [1,7,28,41,47]; (3) our autoencoder uses up to 75% fewer parameters than linear PCA models, while being more accurate in terms of reconstruction error; (4) we show that replacing the expression space of a state of the art face model, FLAME [28], by CoMA improves its reconstruction accuracy; (5) we show that our model can be used in a variational setting to sample a diversity of facial meshes from a known

Gaussian distribution; (6) we provide 20,466 frames of complex 3D head meshes from 12 different subjects for a range of extreme facial expressions along with our code and trained models for research purposes.

2 Related Work

Face Representations. Blanz and Vetter [2] introduced the *morphable model*; the first generic representation for 3D faces based on principal component analysis (PCA) to describe facial shape and texture variations. We also refer the reader to Brunton et al. [13] for a comprehensive overview of 3D face representations. To date, the Basel Face Model (BFM) [36], i.e. the publicly available variant of the morphable model, is the most widely used representation for 3D face shape in a neutral expression. Booth et al. [3] recently proposed another linear neutral expression 3D face model learned from almost 10,000 face scans of more diverse subjects.

Representing facial expressions with linear spaces, or higher-order generalizations thereof, remains the state-of-the-art. The linear expression basis vectors are either computed using PCA [1,7,28,41,47], or are manually defined using linear blendshapes (e.g. [6,27,42]). Yang et al. [47] use multiple PCA models, one per expression, Amberg et al. [1] combine a neutral shape PCA model with a PCA model on the expression residuals from the neutral shape. A similar model with an additional albedo model was used within the Face2Face framework [43]. The recently published FLAME model [28] additionally models head rotation, and yaw motion with linear blendskinning and achieves state-of-the-art results. Vlasic et al. [46] introduce multilinear models, i.e., a higher-order generalization of PCA to model expressive 3D faces. Recently, Abrevaya et al. [18] propose an autoencoder with a CNN-based encoder and a multilinear model as a decoder. Opposed to our mesh autoencoder, their encoder operates on depth images rather than directly on meshes. For all these methods, the model parameters globally influence the shape; i.e. each parameter affects all the vertices of the face mesh. Our convolutional mesh autoencoder however models localized variations due to the hierarchical multiscale nature of the convolutions combined with the down- and up-sampling.

To capture localized facial details, Neumann et al. [33] and Ferrari et al. [19] use sparse linear models. Brunton et al. [12] use a hierarchical multiscale approach by computing localized multilinear models on wavelet coefficients. While Brunton et al. [12] also used a hierarchical multi-scale representation, their method does not use shared parameters across the entire domain. Note that sampling in localized low-dimensional spaces [12] is difficult due to the locality of the facial features; combinations of localized facial features are unlikely to form plausible global face shapes. One goal of our work is to generate new face meshes by sampling the latent space, thus we design our autoencoder to use a single low-dimensional latent space.

Jackson et al. [25] use a volumetric face representation in their CNN-based framework. In contrast to existing face representation methods, our mesh

autoencoder uses convolutional layers to represent faces with significantly fewer parameters. Since it is defined completely on the mesh space, we do not have memory constraints which affect volumetric convolutional methods for representing 3D models.

Convolutional Networks. Bronstein et al. [10] give a comprehensive overview of generalizations of CNNs on non-Euclidean domains, including meshes and graphs. Masci et al. [31] define the first mesh convolutions by locally parameterizing the surface around each point using geodesic polar coordinates, and defining convolutions on the resulting angular bins. In a follow-up work, Boscaini et al. [5] parametrize local intrinsic patches around each point using anisotropic heat kernels. Monti et al. [32] introduce d-dimensional pseudo-coordinates that define a local system around each point with weight functions. This method resembles the intrinsic mesh convolution of [31] and [5] for specific choices of the weight functions. In contrast, Monti el al. [32] use Gaussian kernels with a trainable mean vector and covariance matrix as weight functions.

Verma et al. [45] presente dynamic filtering on graphs where the filter weights depend on the inputs. This work does not focus on reducing the dimensionality of graphs or meshes. Yi et al. [48] also present a spectral CNN for labeling nodes but does not involve any mesh dimensionality reduction. Sinha et al. [39] and Maron et al. [30] embed mesh surfaces into planar images to apply conventional CNNs. Sinha et al. use a robust spherical parametrization to project the surface onto an octahedron, which is then cut and unfolded to form a square image. Maron et al. [30] introduce a conformal mapping from the mesh surface into a flat torus. Litany et al. [29] use graph convolutions for shape completion.

Although, the above methods presented generalizations of convolutions on meshes, they do not use a structure to reduce the meshes to a low dimensional space. Our proposed autoencoder efficiently handles these problems by combining the mesh convolutions with efficient mesh-downsampling and mesh-upsampling operators.

Bruna et al. [11] propose the first generalization of CNNs on graphs by exploiting the connection of the graph Laplacian and the Fourier basis (see Sect. 3 for more details). This leads to spectral filters that generalize graph convolutions. Boscaini et al. [4] extend this using a windowed Fourier transform to localize in frequency space. Henaff et al. [24] build upon the work of Bruna et al. by adding a procedure to estimate the structure of the graph. To reduce the computational complexity of the spectral graph convolutions, Defferrard et al. [17] approximate the spectral filters by truncated Chebyshev poynomials, which avoids explicitly computing the Laplacian eigenvectors, and introduce an efficient pooling operator for graphs. Kipf and Welling [26] simplify this using only first-order Chebyshev polynomials.

However, these graph CNNs are not directly applied to 3D meshes. CoMA uses truncated Chebyshev polynomials [17] as mesh convolutions. In addition, we define mesh down-sampling and up-sampling layers to obtain a complete mesh autoencoder structure to represent highly complex 3D faces, obtaining state of the art results in 3D face modeling.

3 Mesh Operators

We define a 3D facial mesh as a set of vertices and edges, $\mathcal{F} = (\mathcal{V}, A)$, with $|\mathcal{V}| = n$ vertices that lie in 3D Euclidean space, $\mathcal{V} \in \mathbb{R}^{n\times 3}$. The sparse adjacency matrix $A \in \{0,1\}^{n\times n}$ represents the edge connections, where $A_{ij} = 1$ denotes an edge connecting vertices i and j, and $A_{ij} = 0$ otherwise. The non-normalized graph Laplacian [15] is defined as $L = D - A$, with the diagonal matrix D that represents the degree of each vertex in $\mathcal{V}$ as $D_{ii} = \sum_j A_{ij}$.

The Laplacian is diagonalized by the Fourier basis $U \in \mathbb{R}^{n\times n}$ (as L is a real symmetric matrix) as $L = U\Lambda U^T$, where the columns of $U = [u_0, u_1, ..., u_{n-1}]$ are the orthogonal eigenvectors of L, and $\Lambda = diag([\lambda_0, \lambda_1, ..., \lambda_{n-1}]) \in \mathbb{R}^{n\times n}$ is a diagonal matrix with the associated real, non-negative eigenvalues. The graph Fourier transform [15] of the mesh vertices $x \in \mathbb{R}^{n\times 3}$ is then defined as $x_\omega = U^T x$, and the inverse Fourier transform as $x = U x_\omega$.

3.1 Fast Spectral Convolutions

The convolution operator $*$ can be defined in Fourier space as a Hadamard product, $x * y = U((U^T x) \odot (U^T y))$. This is computationally expensive with large numbers of vertices, since U is not sparse. The problem is addressed by formulating mesh filtering with a kernel g_θ using a recursive Chebyshev polynomial [17,23]. The filter g_θ is parametrized as a Chebyshev polynomial of order K given by

$$g_\theta(L) = \sum_{k=0}^{K-1} \theta_k T_k(\tilde{L}), \tag{1}$$

where $\tilde{L} = 2L/\lambda_{max} - I_n$ is the scaled Laplacian, the parameter $\theta \in \mathbb{R}^K$ is a vector of Chebyshev coefficients, and $T_k \in \mathbb{R}^{n\times n}$ is the Chebyshev polynomial of order k that can be computed recursively as $T_k(x) = 2xT_{k-1}(x) - T_{k-2}(x)$ with $T_0 = 1$ and $T_1 = x$. The spectral convolution can then be defined as in [17]

$$y_j = \sum_{i=1}^{F_{in}} g_{\theta_{i,j}}(L) x_i \in \mathbb{R}^n, \tag{2}$$

where y_j computes the j^{th} feature of $y \in \mathbb{R}^{n\times F_{out}}$. The input $x \in \mathbb{R}^{n\times F_{in}}$ has F_{in} features. The input face mesh has $F_{in} = 3$ features corresponding to its 3D vertex positions. Each convolutional layer has $F_{in} \times F_{out}$ vectors of Chebyshev coefficients, $\theta_{i,j} \in \mathbb{R}^K$, as trainable parameters.

3.2 Mesh Sampling

In order to capture both global and local context, we seek a hierarchical multi-scale representation of the mesh. This allows convolutional kernels to capture local context in the shallow layers and global context in the deeper layers of the network. In order to address this representation problem, we introduce mesh

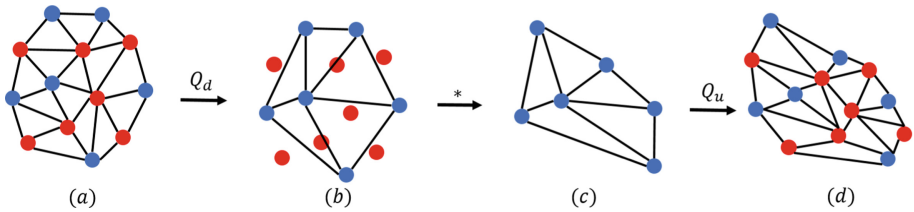

Fig. 1. Mesh sampling operations: a mesh feature (a) is down-sampled by removing red vertices that minimize quadric error [20]. We store the barycentric coordinates of the red vertices w.r.t. the down-sampled mesh (b). The down-sampled mesh can then be transformed using convolutional operations to obtain the transformed mesh (c). The contracted vertices are then added at the barycentric locations (d). (Color figure online)

sampling operators that define the down-sampling and up-sampling of a mesh feature in a neural network. A mesh feature with n vertices can be represented using a $n \times F$ tensor, where F is the dimensionality of each vertex. A 3D mesh is represented with $F = 3$. However, applying convolutions to the mesh can result in features with different dimensionality. The mesh sampling operations define a new topological structure at each layer and maintain the context on neighborhood vertices. We now describe our sampling method with an overview as shown in Fig. 1.

We perform the in-network down-sampling of a mesh with m vertices using transform matrices $Q_d \in \{0,1\}^{n \times m}$, and up-sampling using $Q_u \in \mathbb{R}^{m \times n}$ where $m > n$. The down-sampling is obtained by contracting vertex pairs iteratively that maintain surface error approximations using quadric matrices [20]. In Fig. 1(a), the red vertices are contracted during the down-sampling operation. The (blue) vertices after down-sampling are a subset of the original mesh vertices $\mathcal{V}_d \subset \mathcal{V}$. Each weight $Q_d(p,q) \in \{0,1\}$ denotes whether the q-th vertex is kept during down-sampling, $Q_d(p,q) = 1$, or discarded where $Q_d(p,q) = 0$, $\forall p$.

Since a loss-less down-sampling and up-sampling is not feasible for general surfaces, the up-sampling matrix is built during down-sampling. Vertices retained during down-sampling (blue) undergo convolutional transformations, see Fig. 1(c). These (blue) vertices are retained during up-sampling $Q_u(q,p) = 1$ iff $Q_d(p,q) = 1$. Vertices $v_q \in \mathcal{V}$ discarded during down-sampling (red vertices) where $Q_d(p,q) = 0$ $\forall p$, are mapped into the down-sampled mesh surface using barycentric coordinates. As shown in Figs. 1(b)–(d), this is done by projecting v_q into the closest triangle (i,j,k) in the down-sampled mesh, denoted by $\widetilde{v}_p$, and computing the barycentric coordinates, $\widetilde{v}_p = w_i v_i + w_j v_j + w_k v_k$, such that $v_i, v_j, v_k \in \mathcal{V}_d$ and $w_i + w_j + w_k = 1$. The weights are then updated in Q_u as $Q_u(q,i) = w_i$, $Q_u(q,j) = w_j$, and $Q_u(q,k) = w_k$, and $Q_u(q,l) = 0$ otherwise. The up-sampled mesh with vertices $\mathcal{V}_u$ is obtained using sparse matrix multiplication, $\mathcal{V}_u = Q_u \mathcal{V}_d$.

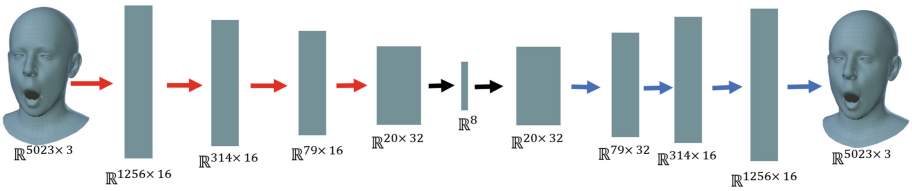

Fig. 2. Convolutional mesh autoencoder: the red and blue arrows indicate down-sampling and up-sampling layers respectively. (Color figure online)

4 Mesh Autoencoder

Network Architecture. Our autoencoder consists of an encoder and a decoder. The structure of the encoder is shown in Table 1. The encoder consists of 4 Chebyshev convolutional filters with $K = 6$ Chebyshev polynomials. Each of the convolutions is followed by a biased ReLU [21]. The down-sampling layers are interleaved between convolutional layers. Each of the down-sampling layers reduce the number of mesh vertices by approximately 4 times. The encoder transforms the face mesh from $\mathbb{R}^{n\times 3}$ to an 8 dimensional latent vector using a fully connected layer at the end.

The structure of the decoder is shown in Table 2. The decoder similarly consists of a fully connected layer that transforms the latent vector from $\mathbb{R}^8$ to $\mathbb{R}^{20\times 32}$ that can be further up-sampled to reconstruct the mesh. Following the decoder's fully connected layer, 4 convolutional layers with interleaved up-sampling layers generate a 3D mesh in $\mathbb{R}^{5023\times 3}$. Each of the convolutions is followed by a biased ReLU similar to the encoder network. Each up-sampling layer increases the numbers of vertices by approximately 4 times. Figure 2 shows the complete structure of our mesh autoencoder.

Table 1. Encoder architecture

Layer	Input size	Output size
Convolution	5023×3	5023×16
Down-sampling	5023×16	1256×16
Convolution	1256×16	1256×16
Down-sampling	1256×16	314×16
Convolution	314×16	314×16
Down-sampling	314×16	79×16
Convolution	79×16	79×32
Down-sampling	79×32	20×32
Fully connected	20×32	8

Table 2. Decoder architecture

Layer	Input size	Output size
Fully connected	8	20×32
Up-sampling	20×32	79×32
Convolution	79×32	79×32
Up-sampling	79×32	314×32
Convolution	314×32	314×16
Up-sampling	314×16	1256×16
Convolution	1256×16	1256×16
Up-sampling	1256×16	5023×16
Convolution	5023×16	5023×3

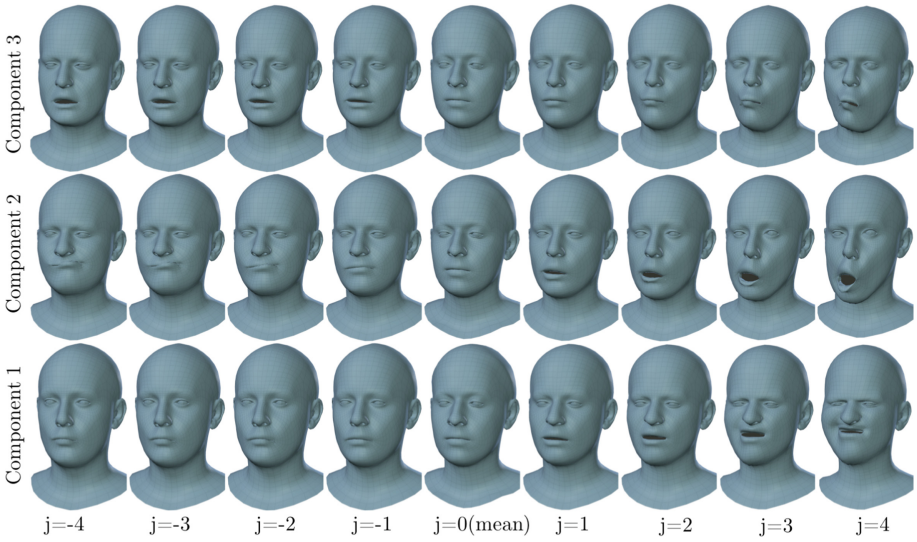

Fig. 3. Sampling from the latent space of the mesh autoencoder around the mean face $j = 0$ along 3 different components.

Training Details. We train our autoencoder for 300 epochs with a learning rate of 8e-3 and a learning rate decay of 0.99 every epoch. We use stochastic gradient descent with a momentum of 0.9 to optimize the L1 loss between predicted mesh vertices and the ground truth samples. We use L1 regularization on the weights of the network using weight decay of 5e-4. The convolutions use Chebyshev filtering with $K = 6$.

5 Experiments

In this section, we evaluate the effectiveness of CoMA on an extreme facial expression dataset. We demonstrate that CoMA allows the synthesis of new expressive faces by sampling from the latent space in Sect. 5.2, including the effect of adding variational loss. Following, we compare CoMA to the widely used PCA representation for reconstructing expressive 3D faces. For this, we evaluate in Sect. 5.3 the ability to reconstruct data similar to the training data (interpolation experiment), and the ability to reconstruct expressions not seen during training (extrapolation experiment). Finally, in Sect. 5.4, we show improved performance by replacing the expression space of state of the art face model, FLAME [28] with our autoencoder.

5.1 Facial Expression Dataset

Our dataset consists of 12 classes of extreme expressions from 12 different subjects. These expressions are complex and asymmetric. The expression sequences

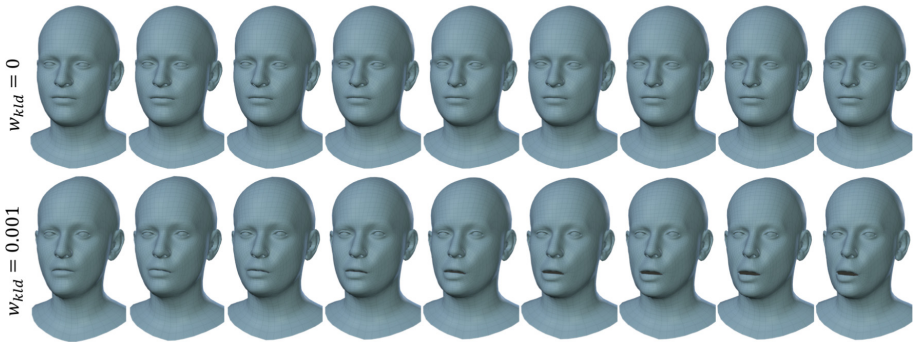

Fig. 4. Sampling using Gaussian noise with variational loss (bottom), and without (top). With $w_{kld} = 0$, the latent representation might not have a Gaussian distribution. Hence, samples on top are not diverse.

in our dataset are – bareteeth, cheeks in, eyebrow, high smile, lips back, lips up, mouth down, mouth extreme, mouth middle, mouth side and mouth up. We show samples from our dataset and the number of frames of each captured sequence in the Supplementary Material.

The data is captured at 60 fps with a multi-camera active stereo system (3dMD LLC, Atlanta) with six stereo camera pairs, five speckle projectors, and six color cameras. Our dataset contains 20,466 3D Meshes, each with about 120,000 vertices. The data is pre-processed using a sequential mesh registration method [28] to reduce the data dimensionality to 5023 vertices.

5.2 Sampling the Latent Space

Let E be the encoder and D be the decoder. We first encode a face mesh from our test set in the latent space to obtain features $z = E(\mathcal{F})$. We then vary each of the components of the latent vector as $\tilde{z}_i = z_i + \epsilon$. We then use the decoder to transform the latent vector into a reconstructed mesh $\tilde{\mathcal{F}} = D(\tilde{z})$. In Fig. 3, we show a diversity of face meshes sampled from the latent space. Here, we extend or contract the latent vector along different dimensions by a factor of 0.3 such that $\tilde{z}_i = (1 + 0.3j)z_i$, where j is the step. In Fig. 3, $j \in [-4, 4]$, and the mean face $\mathcal{F}$ is shown in the middle of the row. More examples are shown in the Supplementary Material.

Variational Convolutional Mesh Autoencoder. Although 3D faces can be sampled from our convolutional mesh autoencoder, the distribution of the latent space is not known. Therefore, sampling requires a mesh to be encoded in that space. In order to constrain the distribution of the latent space, we add a variational loss on our model. Let E be the encoder, D be the decoder, and z be the latent representation of face $\mathcal{F}$. We minimize the loss,

$$l = ||\mathcal{F} - D(z)||_1 + w_{kld} KL(\mathcal{N}(0, 1)||Q(z|\mathcal{F})), \tag{3}$$

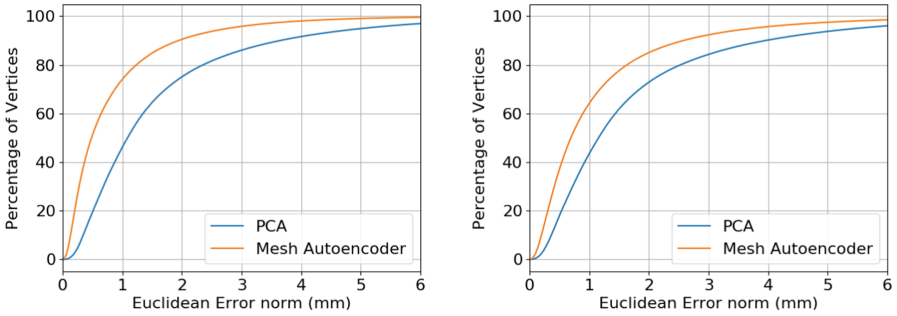

Fig. 5. Cumulative euclidean error between PCA model and mesh autoencoder for interpolation (a) and extrapolation (b) experiments

where $w_{kld} = 0.001$ weights the KL divergence loss. The first term minimizes the L1 reconstruction error, and the second term enforces a unit Gaussian prior $\mathcal{N}(0, 1)$ with zero mean on the distribution of latent vectors $Q(z)$. This enforces the latent space to be a multivariate Gaussian. In Fig. 4, we show visualizations by sampling faces from a Gaussian distribution on this space within $[-3\sigma, 3\sigma]$, where $\sigma = 1$, is the variance of the Gaussian prior. We compare the visualizations by setting $w_{kld} = 0$. We observe that $w_{kld} = 0$ does not enforce any Gaussian prior on $P(z)$, and therefore sampling with Gaussian noise from this distribution results in limited diversity in face meshes. We show more examples in the Supplementary Material.

5.3 Comparison with PCA Spaces

Several face models use PCA space to represent identity and expression variations [1,7,28,41,47]. We perform interpolation and extrapolation experiments to evaluate our performance. We use Scikit-learn [37] to compute PCA coefficients. We consistently use an 8-dimensional latent space to encode the face mesh using both the PCA model and Mesh Autoencoder.

Interpolation Experiment. In order to evaluate the interpolation capability of the autoencoder, we split the dataset in training and test samples with a ratio of 9:1. The test samples are obtained by picking consecutive frames of length 10 uniformly at random across the sequences. We train CoMA for 300 epochs and evaluate it on the test set. We use Euclidean distance for comparison with the PCA method. The mean error with standard deviation, and median errors are shown in Table 3 for comparison.

We observe that our reconstruction error is 50% lower than PCA. At the same time, the number of parameters in CoMA is about 75% fewer than the PCA model as shown in Table 3. Visual inspection of our qualitative results in Fig. 6 shows that our reconstructions are more realistic and are effective in capturing extreme facial expressions. We also show the histogram of cumulative

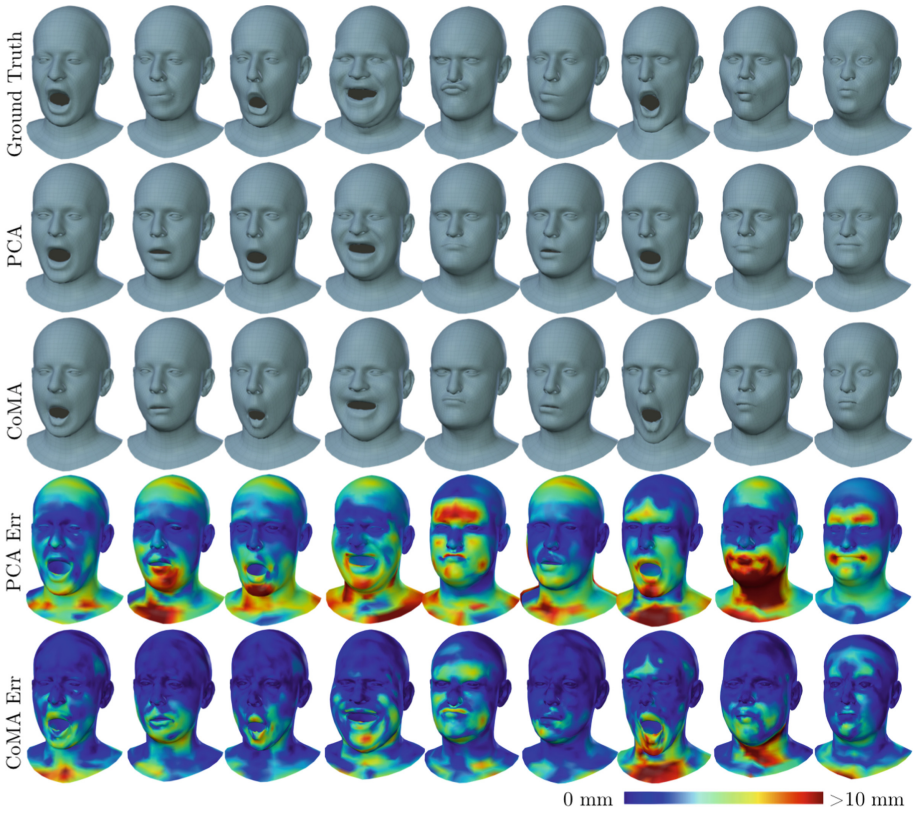

Fig. 6. Comparison with PCA: qualitative results for interpolation experiment

Table 3. Comparison with PCA: interpolation experiments. Errors are in millimeters

	Mean error	Median error	# Parameters
PCA	1.639 ± 1.638	1.101	120,552
Mesh autoencoder	**0.845 ± 0.994**	**0.496**	**33,856**

errors in Fig. 5a. We observe that our Mesh Autoencoder (CoMA) has about 72.6% of the vertices within a Euclidean error of 1 mm, as compared to 47.3% for the PCA model.

Extrapolation Experiment. To measure generalization of our model, we compare the performance of CoMA with the PCA model and FLAME [28]. For comparison, we train the expression model of FLAME on our dataset. The FLAME reconstructions are obtained with latent vector size of 16 with 8 components each for encoding identity and expression. The latent vectors encoded using the PCA model and Mesh autoencoder have a size of 8.

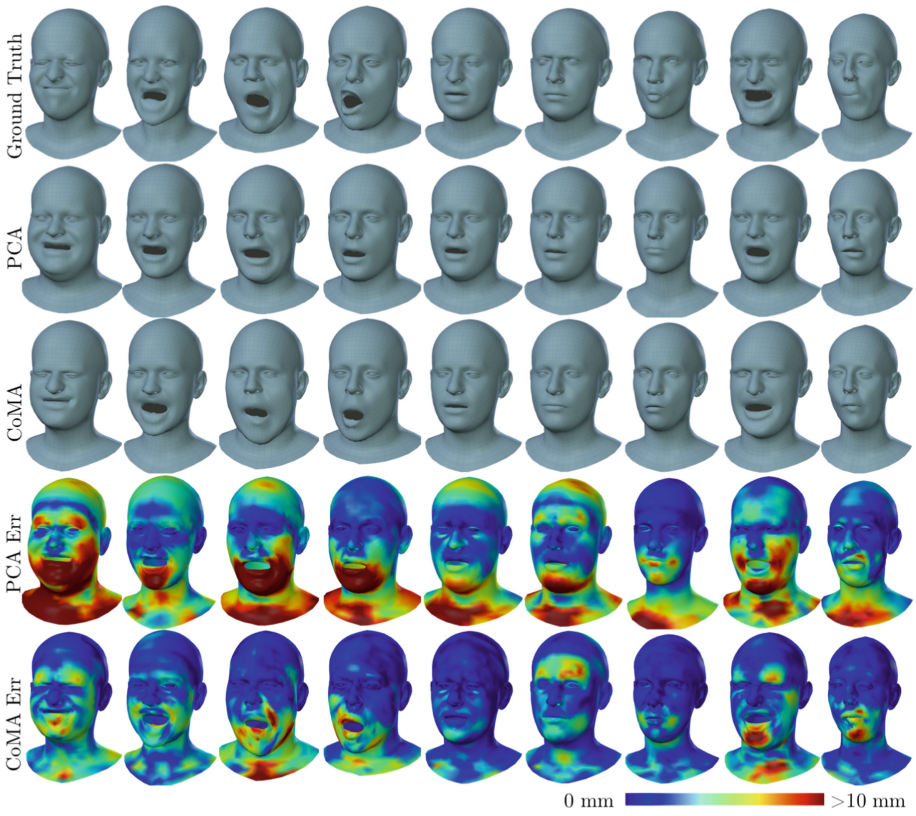

Fig. 7. Comparison with PCA: qualitative results for extrapolation experiment

Table 4. Comparison with PCA: extrapolation experiment. Errors are in millimeters.

	Mesh autoencoder		PCA		FLAME [28]	
Sequence	Mean error	Median	Mean error	Median	Mean error	Median
Bareteeth	**1.376 ± 1.536**	**0.856**	1.957 ± 1.888	1.335	2.002 ± 1.456	1.606
Cheeks in	**1.288 ± 1.501**	**0.794**	1.854 ± 1.906	1.179	2.011 ± 1.468	1.609
Eyebrow	**1.053 ± 1.088**	**0.706**	1.609 ± 1.535	1.090	1.862 ± 1.342	1.516
High smile	**1.205 ± 1.252**	**0.772**	1.841 ± 1.831	1.246	1.960 ± 1.370	1.625
Lips back	**1.193 ± 1.476**	**0.708**	1.842 ± 1.947	1.198	2.047 ± 1.485	1.639
Lips up	**1.081 ± 1.192**	**0.656**	1.788 ± 1.764	1.216	1.983 ± 1.427	1.616
Mouth down	**1.050 ± 1.183**	**0.654**	1.618 ± 1.594	1.105	2.029 ± 1.454	1.651
Mouth extreme	**1.336 ± 1.820**	**0.738**	2.011 ± 2.405	1.224	2.028 ± 1.464	1.613
Mouth middle	**1.017 ± 1.192**	**0.610**	1.697 ± 1.715	1.133	2.043 ± 1.496	1.620
Mouth open	**0.961 ± 1.127**	**0.583**	1.612 ± 1.728	1.060	1.894 ± 1.422	1.544
Mouth side	**1.264 ± 1.611**	**0.730**	1.894 ± 2.274	1.132	2.090 ± 1.510	1.659
Mouth up	**1.097 ± 1.212**	**0.683**	1.710 ± 1.680	1.159	2.067 ± 1.485	1.680

To evaluate generalization capability of our model, we reconstruct the expressions that are completely unseen by our model. We perform 12 different experiments for evaluation. For each experiment, we split our dataset by completely excluding one expression set from all the subjects of the dataset. We test our Mesh Autoencoder on the excluded expression. We compare the performance of our model with PCA and FLAME using the Euclidean distance (mean, standard deviation, median). We perform 12 fold cross validation, one for each expression as shown in Table 4. In Table 4, we also show that our model performs better than PCA and FLAME [28] on all expression sequences. We show the qualitative results in Fig. 7. We show the cumulative Euclidean error histogram in Fig. 5b. For a 1 mm accuracy, Mesh Autoencoder captures 63.8% of the vertices while the PCA model captures 45%.

5.4 DeepFLAME

FLAME [28] is a state of the art model for face representation that combines linear blendskinning for head and jaw motion with linear PCA spaces to represent identity and expression shape variations. To improve the reconstruction error of FLAME, we replace the PCA expression space of FLAME with our autoencoder, and refer to the new model as DeepFLAME. We compare the performance of DeepFLAME with FLAME by varying the size of the latent vector for encoding. Head rotations are factored out for comparison since they are well modeled by linear blendskinning in FLAME, and we consider only the expression space. The reconstruction accuracy is measured using Euclidean distance metric. We show the comparisons in Table 5. The median reconstruction of DeepFLAME is lower for all chosen latent space dimensions, while the mean reconstruction error is lower for up to 12 latent variables. This shows that DeepFLAME provides a more compact face representation; i.e., captures more shape variation with fewer latent variables.

Table 5. Comparison of FLAME and DeepFLAME. DeepFLAME is obtained by replacing expression model of FLAME with CoMA. All errors are in millimeters.

	DeepFLAME		FLAME [28]	
#dim of z	Mean error	Median	Mean error	Median
2	$\mathbf{0.610 \pm 0.851}$	**0.317**	0.668 ± 0.876	0.371
4	$\mathbf{0.509 \pm 0.746}$	**0.235**	0.589 ± 0.803	0.305
6	$\mathbf{0.464 \pm 0.711}$	**0.196**	0.525 ± 0.743	0.252
8	$\mathbf{0.432 \pm 0.681}$	**0.169**	0.477 ± 0.691	0.217
10	$\mathbf{0.421 \pm 0.664}$	**0.162**	0.439 ± 0.655	0.193
12	$\mathbf{0.388 \pm 0.630}$	**0.139**	0.403 ± 0.604	0.172
14	0.371 ± 0.605	**0.128**	$\mathbf{0.371 \pm 0.567}$	0.152
16	0.372 ± 0.611	**0.125**	$\mathbf{0.351 \pm 0.543}$	0.139

5.5 Discussion

The focus of CoMA is to model facial shape for reconstruction applications. The Laplace-Beltrami operator (LBo) describes the intrinsic surface geometry and is invariant under isometric surface deformations. This isometry invariance of the LBo is beneficial for shape matching and registration. Since changes in facial expression are near isometric deformations [9, Sect. 13.3], applying LBo to expressive faces would result in a loss of most expression-related shape variations, making it infeasible to model such variations. The graph Laplacian used by CoMA in contrast to the LBo is not isometry invariant.

While we evaluate CoMA on face shapes, it is applicable to any class of objects. Similar to existing statistical models however, it requires all meshes in dense vertex correspondence; i.e. all meshes need to share the same topology. A future research direction is to directly learn a 3D face representation from raw 3D face scans or 2D images without requiring vertex correspondence.

As is also true for other deep learning based models, the performance of CoMA could further improve with more training data. The amount of existing 3D face data however is very limited. The data scarcity especially limits our expression model to outperform existing models for higher latent space dimensions (>12 see Table 5). We predict superior quality on larger datasets and plan to evaluate CoMA on significantly more data in the future.

As CoMA is an end-to-end trained model, it could also be combined with some existing image convolutional network to regress the 3D face shape from 2D images. We will explore this in future work.

6 Conclusion

We have introduced CoMA, a new representation for 3D faces of varying shape and expression. We designed CoMA as a hierarchical, multi-scale representation to capture global and local shape and expression variations of multiple scales. To do so, we introduce novel sampling operations and combine these with fast graph convolutions in an autoencoder network. The locally invariant filters, shared across the mesh surface, significantly reduce the number of filter parameters in the network, and the non-linear activation functions capture extreme facial expressions. We evaluated CoMA on a dataset of extreme 3D facial expressions that we will make publicly available for research purposes along with the trained model. We showed that CoMA significantly outperforms state-of-the-art models in 3D face reconstruction applications while using 75% fewer model parameters. CoMA outperforms the linear PCA model by 50% on interpolation experiments and generalizes better on completely unseen facial expressions. We further demonstrated that CoMA in a variational setting allows us to synthesize new expressive faces by sampling the latent space.

Acknowledgement. We thank T. Alexiadis and J. Márquez for data aquisition; H. Feng for rendering the figures; S. Wuhrer for advice on mesh convolutions; and G. Pavlakos, D. Paschalidou and S. Pujades for helping us with paper revisions.

References

1. Amberg, B., Knothe, R., Vetter, T.: Expression invariant 3D face recognition with a morphable model. In: International Conference on Automatic Face Gesture Recognition, pp. 1–6 (2008)
2. Blanz, V., Vetter, T.: A morphable model for the synthesis of 3D faces. In: SIGGRAPH, pp. 187–194 (1999)
3. Booth, J., Roussos, A., Ponniah, A., Dunaway, D., Zafeiriou, S.: Large scale 3D morphable models. Int. J. Comput. Vis. **126**, 1–22 (2017)
4. Boscaini, D., Masci, J., Melzi, S., Bronstein, M.M., Castellani, U., Vandergheynst, P.: Learning class-specific descriptors for deformable shapes using localized spectral convolutional networks. In: Eurographics Symposium on Geometry Processing, pp. 13–23 (2015)
5. Boscaini, D., Masci, J., Rodolà, E., Bronstein, M.: Learning shape correspondence with anisotropic convolutional neural networks. In: Advances in Neural Information Processing Systems, pp. 3189–3197 (2016)
6. Bouaziz, S., Wang, Y., Pauly, M.: Online modeling for realtime facial animation. ACM Trans. Graph. **32**(4), 40 (2013)
7. Breidt, M., Bülthoff, H.H., Curio, C.: Robust semantic analysis by synthesis of 3D facial motion. In: International Conference on Automatic Face and Gesture Recognition and Workshops, pp. 713–719 (2011)
8. Brock, A., Lim, T., Ritchie, J.M., Weston, N.: Generative and discriminative voxel modeling with convolutional neural networks. arXiv preprint arXiv:1608.04236 (2016)
9. Bronstein, A.M., Bronstein, M.M., Kimmel, R.: Numerical Geometry of Non-Rigid Shapes. Springer, Heidelberg (2008). https://doi.org/10.1007/978-0-387-73301-2
10. Bronstein, M.M., Bruna, J., LeCun, Y., Szlam, A., Vandergheynst, P.: Geometric deep learning: going beyond euclidean data. Signal Process. Mag. **34**(4), 18–42 (2017)
11. Bruna, J., Zaremba, W., Szlam, A., LeCun, Y.: Spectral networks and locally connected networks on graphs. CoRR abs/1312.6203 (2013)
12. Brunton, A., Bolkart, T., Wuhrer, S.: Multilinear wavelets: a statistical shape space for human faces. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds.) ECCV 2014. LNCS, vol. 8689, pp. 297–312. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-10590-1_20
13. Brunton, A., Salazar, A., Bolkart, T., Wuhrer, S.: Review of statistical shape spaces for 3D data with comparative analysis for human faces. Comput. Vis. Image Underst. **128**, 1–17 (2014)
14. Cao, C., Weng, Y., Zhou, S., Tong, Y., Zhou, K.: Facewarehouse: a 3D facial expression database for visual computing. Trans. Vis. Comput. Graph. **20**(3), 413–425 (2014)
15. Chung, F.R.K.: Spectral Graph Theory, vol. 92. American Mathematical Soc., Providence (1997)
16. Cosker, D., Krumhuber, E., Hilton, A.: A FACS valid 3D dynamic action unit database with applications to 3D dynamic morphable facial modeling. In: International Conference on Computer Vision, pp. 2296–2303 (2011)
17. Defferrard, M., Bresson, X., Vandergheynst, P.: Convolutional neural networks on graphs with fast localized spectral filtering. In: Advances in Neural Information Processing Systems, pp. 3844–3852 (2016)

18. Abrevaya, V.F., Wuhrer, S., Boyer, E.: Multilinear autoencoder for 3D face model learning. In: Winter Conference on Applications of Computer Vision, pp. 1–9 (2018)
19. Ferrari, C., Lisanti, G., Berretti, S., Bimbo, A.D.: Dictionary learning based 3D morphable model construction for face recognition with varying expression and pose. In: International Conference on 3D Vision, pp. 509–517 (2015)
20. Garland, M., Heckbert, P.S.: Surface simplification using quadric error metrics. In: Proceedings of the 24th Annual Conference on Computer Graphics And Interactive Techniques, pp. 209–216. ACM Press/Addison-Wesley Publishing Co. (1997)
21. Glorot, X., Bordes, A., Bengio, Y.: Deep sparse rectifier neural networks. In: Fourteenth International Conference on Artificial Intelligence and Statistics (2011)
22. Goodfellow, I., et al.: Generative adversarial nets. In: Advances in Neural Information Processing Systems, pp. 2672–2680 (2014)
23. Hammond, D.K., Vandergheynst, P., Gribonval, R.: Wavelets on graphs via spectral graph theory. Appl. Comput. Harmonic Anal. **30**(2), 129–150 (2011)
24. Henaff, M., Bruna, J., LeCun, Y.: Deep convolutional networks on graph-structured data. CoRR abs/1506.05163 (2015)
25. Jackson, A.S., Bulat, A., Argyriou, V., Tzimiropoulos, G.: Large pose 3D face reconstruction from a single image via direct volumetric CNN regression. In: International Conference on Computer Vision (2017)
26. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. In: International Conference on Learning Representations (2016)
27. Li, H., Weise, T., Pauly, M.: Example-based facial rigging. ACM Trans. Graph. **29**(4), 32 (2010)
28. Li, T., Bolkart, T., Black, M.J., Li, H., Romero, J.: Learning a model of facial shape and expression from 4D scans. ACM Trans. Graph. **36**(6), 194 (2017)
29. Litany, O., Bronstein, A., Bronstein, M., Makadia, A.: Deformable shape completion with graph convolutional autoencoders. arXiv preprint arXiv:1712.00268 (2017)
30. Maron, H., et al.: Convolutional neural networks on surfaces via seamless toric covers. ACM Trans. Graph. **36**(4), 71:1–71:10 (2017)
31. Masci, J., Boscaini, D., Bronstein, M., Vandergheynst, P.: Geodesic convolutional neural networks on Riemannian manifolds. In: International Conference on Computer Vision Workshops, pp. 37–45 (2015)
32. Monti, F., Boscaini, D., Masci, J., Rodolà, E., Svoboda, J., Bronstein, M.M.: Geometric deep learning on graphs and manifolds using mixture model CNNs (2017)
33. Neumann, T., Varanasi, K., Wenger, S., Wacker, M., Magnor, M., Theobalt, C.: Sparse localized deformation components. Trans. Graph. (Proc. SIGGRAPH Asia) **32**(6), 179:1–179:10 (2013)
34. van den Oord, A., et al.: WaveNet: a generative model for raw audio. CoRR abs/1609.03499 (2016)
35. Oord, A.V.D., Kalchbrenner, N., Kavukcuoglu, K.: Pixel recurrent neural networks. arXiv preprint arXiv:1601.06759 (2016)
36. Paysan, P., Knothe, R., Amberg, B., Romdhani, S., Vetter, T.: A 3D face model for pose and illumination invariant face recognition. In: International Conference on Advanced Video and Signal Based Surveillance, pp. 296–301 (2009)
37. Pedregosa, F., et al.: Scikit-learn: machine learning in Python. J. Mach. Learn. Res. **12**, 2825–2830 (2011)
38. Savran, A., et al.: Bosphorus database for 3D face analysis. In: Schouten, B., Juul, N.C., Drygajlo, A., Tistarelli, M. (eds.) BioID 2008. LNCS, vol. 5372, pp. 47–56. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-89991-4_6

39. Sinha, A., Bai, J., Ramani, K.: Deep learning 3D shape surfaces using geometry images. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016. LNCS, vol. 9910, pp. 223–240. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46466-4_14
40. Taigman, Y., Yang, M., Ranzato, M., Wolf, L.: DeepFace: closing the gap to human-level performance in face verification. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1701–1708 (2014)
41. Tewari, A., et al.: MoFA: model-based deep convolutional face autoencoder for unsupervised monocular reconstruction. In: International Conference on Computer Vision (2017)
42. Thies, J., Zollhöfer, M., Nießner, M., Valgaerts, L., Stamminger, M., Theobalt, C.: Real-time expression transfer for facial reenactment. Trans. Graph. **34**(6), 183:1–183:14 (2015)
43. Thies, J., Zollhöfer, M., Stamminger, M., Theobalt, C., Nießner, M.: Face2Face: real-time face capture and reenactment of RGB videos. In: Conference on Computer Vision and Pattern Recognition, pp. 2387–2395 (2016)
44. Tran, A.T., Hassner, T., Masi, I., Paz, E., Nirkin, Y., Medioni, G.: Extreme 3D face reconstruction: looking past occlusions. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2018)
45. Verma, N., Boyer, E., Verbeek, J.: Dynamic filters in graph convolutional networks. CoRR abs/1706.05206 (2017)
46. Vlasic, D., Brand, M., Pfister, H., Popović, J.: Face transfer with multilinear models. Trans. Graph. **24**(3), 426–433 (2005)
47. Yang, F., Wang, J., Shechtman, E., Bourdev, L., Metaxas, D.: Expression flow for 3D-aware face component transfer. Trans. Graph. **30**(4), 60:1–60:10 (2011)
48. Yi, L., Su, H., Guo, X., Guibas, L.J.: SyncSpecCNN: synchronized spectral CNN for 3D shape segmentation (2017)
49. Yin, L., Chen, X., Sun, Y., Worm, T., Reale, M.: A high-resolution 3D dynamic facial expression database. In: International Conference on Automatic Face and Gesture Recognition, pp. 1–6 (2008)
50. Yin, L., Wei, X., Sun, Y., Wang, J., Rosato, M.J.: A 3D facial expression database for facial behavior research. In: International Conference on Automatic Face and Gesture Recognition, pp. 211–216 (2006)