Towards High-Fidelity 3D Face Reconstruction from In-the-Wild Images Using Graph Convolutional Networks

Jiangke Lin, Yi Yuan*
NetEase Fuxi AI Lab
Hangzhou, China
{linjiangke, yuanyi}@corp.netease.com

Tianjia Shao, Kun Zhou
State Key Lab of CAD&CG, Zhejiang University
Hangzhou, China
tianjiashao@gmail.com, kunzhou@acm.org

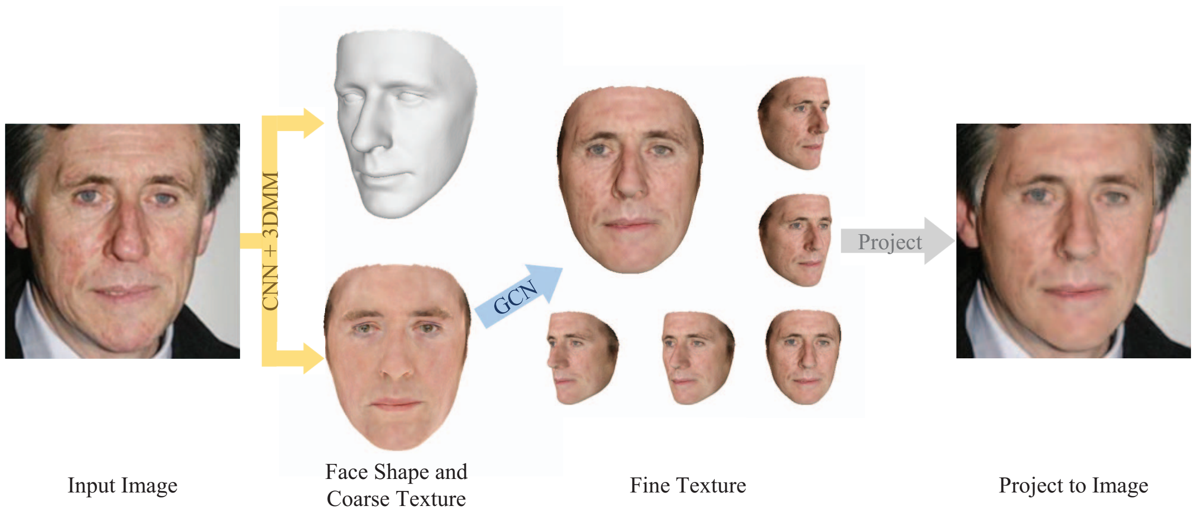

Figure 1: From left to right: the input image, the reconstructed face shape and coarse texture, the fine detailed texture, and the 3D face projected to the input image.

Abstract

3D Morphable Model (3DMM) based methods have achieved great success in recovering 3D face shapes from single-view images. However, the facial textures recovered by such methods lack the fidelity as exhibited in the input images. Recent work demonstrates high-quality facial texture recovering with generative networks trained from a large-scale database of high-resolution UV maps of face textures, which is hard to prepare and not publicly available. In this paper, we introduce a method to reconstruct 3D facial shapes with high-fidelity textures from single-view images in-the-wild, without the need to capture a large-scale face texture database. The main idea is to refine the initial texture generated by a 3DMM based method with facial details from the input image. To this end, we propose to use graph convolutional networks to reconstruct the detailed colors for the mesh vertices instead of reconstructing the UV map. Experiments show that our method can generate high-quality results and outperforms state-of-the-art methods in both qualitative and quantitative comparisons.

*Corresponding author

1. Introduction

Reconstructing the 3D facial shape and texture from a single image is a vital problem in computer vision and graphics. The seminal work of Blanz and Vetter [3] illustrates the power of the 3D Morphable Model (3DMM), which is a statistical model of facial shape and texture built from hundreds of scanned faces. The 3DMM and its variants [4, 6, 14, 18, 23] make it possible to recover the facial shape and albedo from a single image [11, 13, 31, 36]. However, the fidelity of the texture recovered from the 3DMM coefficients is still not high enough. It is mainly because that the texture computed from 3DMM cannot capture the face details of the input image, especially for the

2575-7075/20/$31.00 ©2020 IEEE
DOI 10.1109/CVPR42600.2020.00593

images in-the-wild.

In order to tackle the problem, recently there have been some work [10, 12] trying to reconstruct high-quality textures from 2D images. For example, Deng *et al*. [10] learn a generative model to complete the self-occluded regions in the facial UV map. Gecer *et al*. [12] first utilize GANs to train a generator of facial textures in UV space and then use non-linear optimization to find the optimal latent parameters. While they can achieve high fidelity textures, their methods require a large-scale database of high-resolution UV maps, which is not publicly available. Besides, capturing such a database is rather laborious, which is infeasible for ordinary users.

In this paper, we seek to reconstruct the 3D facial shape with high fidelity texture from a single image, without the need to capture a large-scale face texture database. To achieve this goal, we make a key observation that, though lacking details, the 3DMM texture model can provide a globally-reasonable color for the whole face mesh. We can further refine this initial texture by introducing the facial details from the image to the face mesh. To this end, we propose to reconstruct the detailed colors for the mesh vertices instead of reconstructing the UV map. In particular, we utilize graph convolutional networks (GCN) [9, 20, 27] to decode the image features and propagate the detailed RGB values to the vertices of the face mesh.

Our reconstruction framework works in a coarse-to-fine manner, based on a combination of a 3DMM model and graph convolutional networks. A convolutional neural network (CNN) is trained to regress 3DMM coefficients (shape/expression/texture) and rendering parameters (pose/lighting) from a 2D image. With the 3DMM model, the face shape and initial coarse texture can be computed with an affine model. At a key stage, we utilize a pre-trained CNN to extract face features from the image and feed them to a graph convolutional network to produce detailed colors for the mesh vertices. Our framework adopts a differentiable rendering layer [13] to enable self-supervised training, and further improves the results with a GAN loss [16].

To summarize, this paper makes the following contributions:

- We propose a coarse-to-fine framework for reconstructing 3D faces with high fidelity textures from single images, without requiring to capture a large scale of high-resolution face texture data.
- To the best of our knowledge, we are the first to use graph convolutional networks to generate high fidelity face texture, which is able to generate detailed colors for the mesh vertices from the image.
- We compare our results with state-of-the-art methods, and ours outperform others in both qualitative and quantitative comparison.

2. Related Work

2.1. Morphable 3D Face Models

Blanz and Vetter [3] introduced the first 3D morphable face model twenty years ago. Since then, there have been multiple variations of 3DMM [4, 6, 14, 18, 23]. Those models produce low-dimensional representations for the facial identity, expression and texture from multiple face scans using PCA. One of the most widely used, publicly available variants of 3DMM is the Basel Face Model (BFM) [26]. It registers a template mesh to the scanned faces with the Optimal Step Nonrigid ICP algorithm, then employs PCA for dimensionality reduction to construct the model. We will use this model as our 3DMM model in our experiments.

In a 3DMM, given the identity coefficients c_i, expression coefficients c_e and texture coefficients c_t, the face shape S and the texture T can be represented as:

$$\begin{aligned} S &= S_{mean} + c_i I_{base} + c_e E_{base} \\ T &= T_{mean} + c_t T_{base} \end{aligned} \tag{1}$$

where S_{mean} and T_{mean} are the mean face shape and texture, and I_{base}, E_{base} and T_{base} are the PCA bases of identity, expression and texture respectively. We use S_{mean}, T_{mean}, I_{base} and T_{base} from BFM [26], and E_{base} built from FaceWarehouse [6].

2.2. Fitting a Morphable Face Model

A classical approach for 3D face reconstruction from a single image is to iteratively fit a template model to the input 2D image. However, this approach is sensitive to the lighting, expression, and pose of the 2D face image. Some improvements [2, 22, 30] have been made to improve the fitting stability and accuracy, but they don't perform well on images in the wild.

Deep learning based methods directly regress the 3DMM coefficients from images. To obtain paired 2D-3D data for supervised learning, Richardson *et al*. [28, 29] generate synthetic data by random sampling from the morphable face model, but this approach may not perform well when dealing with complex lighting, occlusion, and other in-the-wild conditions. Tran *et al*. [35] do not generate synthetic data directly. Instead, they create the ground truth using an iterative optimization to fit a large number of face images. Nevertheless, the problem of being delicate in uncontrolled environments still remains.

Recently, differentiable renderers [13, 34] has been introduced. It renders a 3D face mesh to a 2D image based on the face shape, texture, lighting and other related parameters, and compare the rendered image with the input image to compute the loss in terms of image differences. With such a fixed, differentiable rendering layer, unsupervised or weakly-supervised training is enabled without requiring the training pairs. We also follow this strategy. Specially, we

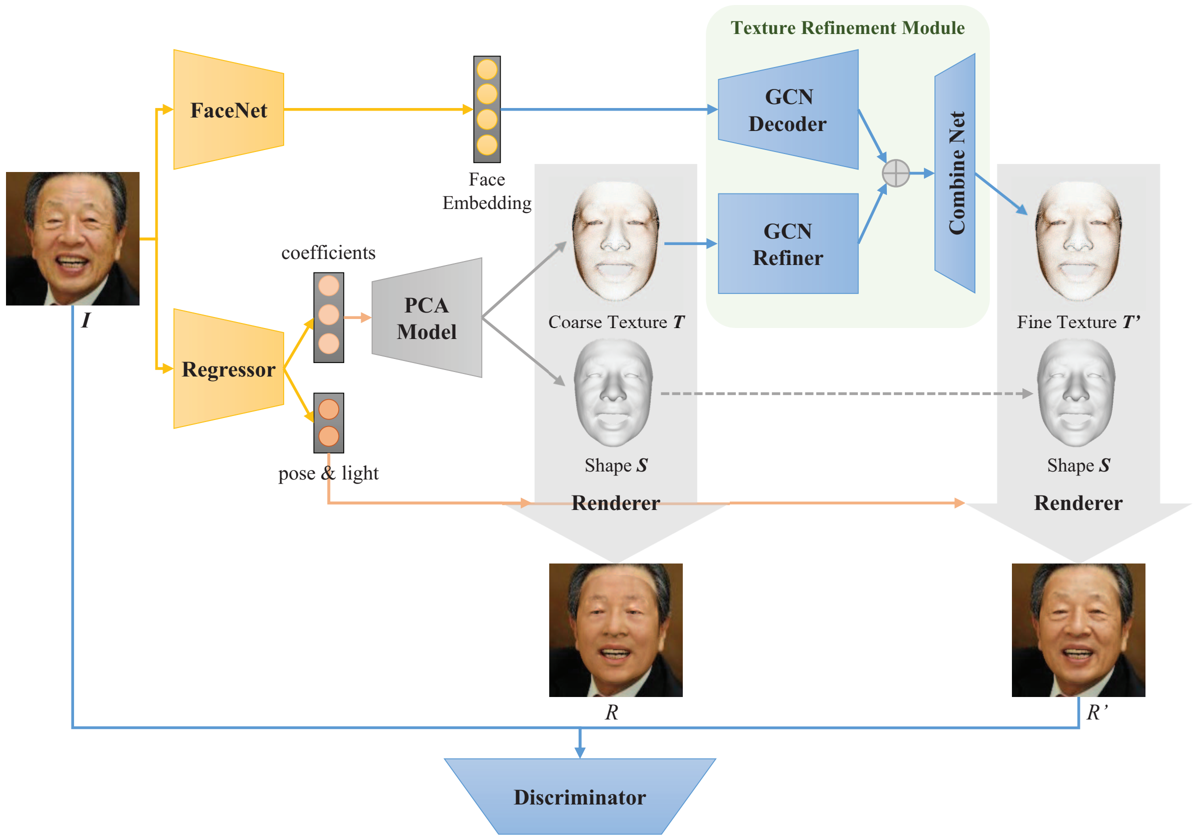

Figure 2: The overall coarse-to-fine framework of our approach. The yellow blocks are pre-trained before others, the grays are non-trainable and the blues are trainable. The Regressor regresses the 3DMM coefficients, face pose and lighting parameters from the input image I, where the 3DMM coefficients will be used to compute the face shape S (coordinates, *e.g.* x, y, z) and coarse texture T (albedo, *e.g.* r, g, b) through the PCA model. The FaceNet is used to extract a face embedding from I. Then the coarse texture and the face embedding are fed into the GCN Refiner and GCN Decoder respectively. The outputs of the two GCNs are concatenated along the channel axis (donated as $\oplus$) and fed to the Combine Net, which produces the fine texture T'. The Discriminator attempts to improve the output of texture refinement module via adversarial training.

adopt the differentiable renderer from Genova *et al.* [13], which is known as "tf-mesh-renderer", as our differentiable rendering layer.

2.3. Texture of Morphable Face Models

Recent deep learning based methods aim to reconstruct the facial textures from single images by regressing the 3DMM texture coefficients. For example, Deng *et al.* [11] proposed a method to simultaneously predict the 3DMM shape and texture coefficients, which employs an illumination and rendering model during training, and conducts image-level and perception-level losses, leading to a better result than others. However, the texture they generated is still inherently limited by the 3DMM texture model. In our method, we deploy their scheme to train a 3DMM coefficient regression model, to get a globally reasonable face texture. Then we utilize graph convolutional networks to refine the texture with the image details.

2.4. Graph Convolutional Networks

To enable convolutional operations in non-Euclidean structured data, Bruna *et al.* [5] utilize the graph Laplacian and the Fourier basis to make the first extension of CNNs on graphs. However, it is computationally expensive and ignores the local features. Defferrard *et al.* [9] propose the ChebyNet, which approximates the spectral filters by truncated Cehbyshev polynomials, avoiding the computation of the Fourier basis. CoMA [27] introduces mesh downsampling and mesh upsampling layers, which constructs an autoencoder to learn a latent embedding of 3D face meshes. Motivated by CoMA [27], our method learns a latent vector representing the detailed face color from a 2D image, then decodes it to produce detailed colors for the face mesh vertices with graph convolutional networks.

3. Approach

We propose a coarse-to-fine approach for 3D face reconstruction. As shown in Fig. 2, our framework is composed of three modules. The feature extraction module includes a Regressor for regressing the 3DMM coefficients, face pose, and lighting parameters, and a FaceNet [32] for extracting image features for the subsequent detail refinement and identity-preserving. The texture refinement module consists of three graph convolutional networks: a GCN Decoder to decode the features extracted from FaceNet and producing detailed colors for mesh vertices, a GCN Refiner to refine the vertex colors generated from the Regressor, and a combiner to combine the two colors to produce final vertex colors. The Discriminator attempts to improve the output of the texture refinement module via adversarial training.

3.1. 3DMM Coefficients Regression

The first step of our algorithm is to regress the 3DMM coefficients and rendering parameters from the input image with CNNs. We adopt the state-of-the-art 3DMM coefficient regressor in [11] for this task. Given a 2D image I, it regresses a 257 dimensional vector $(c_i, c_e, c_t, p, l) \in \mathbb{R}^{257}$, where $c_i \in \mathbb{R}^{80}$, $c_e \in \mathbb{R}^{64}$ and $c_t \in \mathbb{R}^{80}$ represent the 3DMM shape, expression and texture coefficients respectively. $p \in \mathbb{R}^{6}$ is the face pose and $l \in \mathbb{R}^{27}$ is lighting parameters. With the predicted coefficients, the face vertices' 3D positions S and albedo values T can be computed with Eq. 1.

Moreover, we utilize a pre-trained FaceNet [32] to extract a feature vector from the face image. The extracted feature serves two purposes. First, it can be treated as an image feature embedding for our Decoder to generate detailed albedo colors for the mesh vertices, which is described in Sec. 3.3. Second, it can be used to measure the identity distance in the identity-preserving loss in Sec. 3.4.

3.2. Differentiable Rendering

In order to train our networks, we conduct a self-supervised approach with a differentiable rendering layer. That is, we render the face mesh to a 2D image with the predicted parameters, and compute the losses based on the differences between the rendered image and the input image. We adopt the differentiable rendering layer from [13], which introduced a general-purpose, differentiable rasterizer based on a deferred shading model. The rasterizer computes screen-space buffers with triangle IDs and barycentric coordinates for the pixels. The colors and normals from the mesh are interpolated at the pixels. During training, the vertex normals is computed as the average of surrounding triangle normals.

Specifically, with the shape S, texture T and pose generated from the Regressor, we can compute the face albedo projected on the input image. As the projected face albedo is not the final result, we further illuminate the face mesh with the estimated lighting and render it to get the final image R, which is compared with the input image to compute the loss. An example of illumination and rendering is shown in Fig. 3.

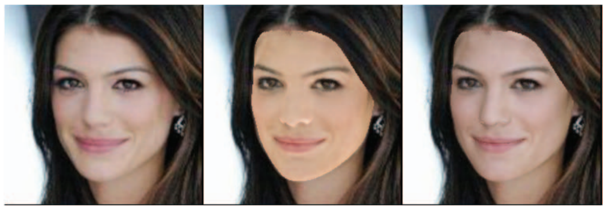

Figure 3: From left to right: the input image, the rendered image with only albedo, and the rendered image with illuminated texture.

3.3. Texture Refinement Module

Our texture refinement module is composed of three graph convolutional networks, namely a Decoder, a Refiner and a combine net. Unlike other work [10, 12] using the UV map as the face texture representation, we directly manipulate the albedo RGB values of the vertices on the face mesh. We deploy a face texture mesh consisting of a set of vertices and triangles, denoted as $\mathcal{M} = (\mathcal{V}, \mathcal{A})$, where $\mathcal{V} \in \mathbb{R}^{n\times 3}$ stores vertex colors, and $\mathcal{A} \in \{0,1\}^{n\times n}$ is the adjacency matrix, representing the triangles. In the adjacency matrix $\mathcal{A}$, if vertex i and j are connected, then $\mathcal{A}_{ij} = 1$, and $\mathcal{A}_{ij} = 0$ otherwise. For normalization, a Laplacian matrix is computed as $\mathcal{L} = \mathcal{I} - D^{-\frac{1}{2}}\mathcal{A}\mathcal{D}^{-\frac{1}{2}}$, where $\mathcal{I}$ is the identity matrix, and $\mathcal{D}$ is the diagonal matrix representing the degree of each vertex in $\mathcal{V}$ as $\mathcal{D}^{ii} = \sum_j \mathcal{A}^{ij}$. The spectral graph convolution operator * between x and y is defined as a Hadamard product in the Fourier space: $x * y = U((U^T x) \odot (U^T y))$, where U is the eigenvectors of Laplacian matrix. To address the problem of computationally expensive caused by non-sparsity of U, [9] proposed a fast spectral convolution method, which constructs mesh filtering with a kernel g_θ using a recursive Chebyshev polynomial [9, 17]:

$$g_\theta(\mathcal{L}) = \sum_{k=0}^{K-1} \theta_k T_k(\widetilde{\mathcal{L}}), \tag{2}$$

where $\widetilde{\mathcal{L}} = 2\mathcal{L}/\lambda_{max} - I$ is the scaled Laplacian matrix, λ_{max} is the maximum eigenvalue of the Laplacian matrix, $\theta \in \mathbb{R}^K$ is the Chebyshev coefficients vector, and $T_k \in \mathbb{R}^{n\times n}$ is the Chebyshev polynomial of order k. T_k is computed recursively as $T_k(x) = 2xT_{k-1}(x) - T_{k-2}(x)$ with the initial $T_0 = 1$ and $T_1 = x$. For each layer, the spectral convolution can be defined as:

$$y_j = \sum_{i=1}^{F_in} g_{\theta_{i,j}}(\mathcal{L})x_i \in \mathbb{R}^n, \tag{3}$$

where $x \in \mathbb{R}^{n \times F_{in}}$ is the input with F_{in} features, and y_j denote the j^{th} feature of $y \in \mathbb{R}^{n \times F_{out}}$, which has F_{out} features. For each spectral convolution layer, there are $F_{in} \times F_{out}$ trainable parameters.

Our Decoder takes as input the feature vector from FaceNet and produces the albedo RGB values for each vertex. The Decoder architecture is built following the idea of residual networks, which consists of four spectral residual blocks. The spectral upsampling layers are placed between each two spectral residual blocks. Each spectral residual block contains two Chebyshev convolutional layers and one shortcut layer. Every Chebyshev convolutional layer uses $K = 6$ Chebyshev polynomials and is followed by a biased ReLU layer [15].

The Refiner refines the vertex colors from the 3DMM model with spectral convolutional layers. It also contains spectral residual blocks similar to the Decoder. However, only one downsampling layer and one upsampling layer are deployed in the bottom and at the top of the network separately.

To produce the final vertex colors with details, the combination net concatenates the outputs of the Decoder and Refiner along the channel axis, and feed them into a graph convolutional layer, followed by a *tanh* activation layer.

We also adopt the same differentiable renderer to self-supervise the training of our texture refinement module as in Sec. 3.2. Specifically, we render the 3D face mesh to R' with illumination and compare it with the input image. To make the final texture own higher fidelity, we further utilize an adversarial training strategy. Since we do not use the real 3D face data, the Discriminator is deployed on the input 2D images and the rendered images from the reconstructed 3D face meshes. Our Discriminator contains 6 convolutional layers with kernel size 3, each of which is followed by a max-pooling layer. During training, we follow the procedure described in Wasserstein GANs with gradient penalty [16].

3.4. Losses

3.4.1 Pixel-wise Loss

A straightforward objective is to minimize the differences between the input images and the rendered images. However, as the input face image may have occlusions (e.g., self-occlusion, glasses, and masks), we only compute the Euclidean distance between certain face regions M_{face}. The face regions are acquired by a pre-trained face segmentation networks following [33] on Halen dataset [21], and we use

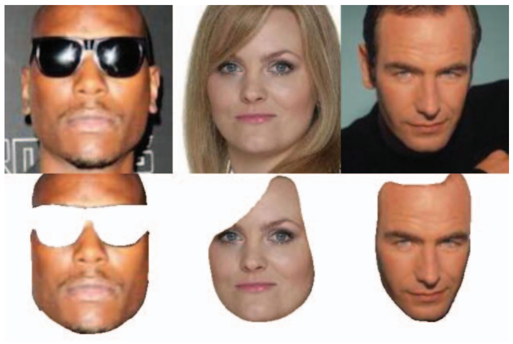

Figure 4: The first row is the input images, and the second row is the face regions for computing losses.

the regions of the face, eyebrows, eyes, nose, and mouth to compute the loss. Some examples are illustrated in Fig. 4.

The pixel-wise loss is defined as:

$$L_{pix}(x, x') = \frac{\sum M_{proj} M_{face} ||x - x'||_2}{\sum M_{proj} M_{face}}, \tag{4}$$

where x is the input image, x' is the rendering image with illumination, and M_{proj} denotes the region that the face mesh can be projected onto.

3.4.2 Identity-Preserving Loss

Using the pixel-level loss could generate a generally good result. However, the reconstructed 3D face might not "look like" the input 2D face image, especially under extreme circumstances. To tackle this issue, we define a loss function at the face feature level, which is called the identity-preserving loss. The loss requires the features between the input image and rendered image, extracted by the FaceNet, should be close to each other. We define the identity-preserving loss as a cosine distance:

$$L_{id}(x, x') = 1 - \frac{< F(x), F(x') >}{||F(x)|| \cdot ||F(x')||}, \tag{5}$$

where x is the input image, x' is the rendered image, and $F(\cdot)$ is the feature extraction function by FaceNet. $< F(x), F(x') >$ is the inner product.

3.4.3 Vertex-wise Loss

When training the refinement module, the graph convolutional networks may not learn the texture RGB values properly from the image due to the occluded regions on the face. Therefore we construct a vertex-wise loss based on the texture predicted by the Regressor as assistance at the early stage of refinement module training, and then reduce the weights of the vertex-wise loss gradually. In order to obtain the small details from a human face, we also retrieve the vertex colors T_p by projecting the face vertices to the 2D image, then feed them to the Refiner along with 3DMM texture RGB values. Taking those two into consideration, our vertex-wise loss can be defined as:

$$L_{vert}(x, x') = \frac{1}{N}\sum_{i=1}^{N} ||x_i - x'_i||_2, \tag{6}$$

where N is the number of vertices. x_i is the albedo value T generated by the Regressor or the retrieved vertex color T_p, and x'_i is the refined albedo T' from the refinement module or the illuminated T', denoted as $\widetilde{T'}$.

3.4.4 Adversarial Loss

For the adversarial training, we adopt the paradigm of Improved Wasserstein GAN [16], whose adversarial loss is defined as:

$$\begin{aligned} L_{adv} = &\mathop{\mathbb{E}}_{x' \sim \mathbb{P}_{R'}}[D(x')] - \mathop{\mathbb{E}}_{x \sim \mathbb{P}_I}[D(x)] + \\ &\lambda \mathop{\mathbb{E}}_{\hat{x} \sim \mathbb{P}_{\hat{x}}}[(||\nabla_{\hat{x}} D(\hat{x})||_2 - 1)^2], \end{aligned} \tag{7}$$

where $\tilde{x}$ is the rendered image from the refined texture and x is the input image. $\hat{x}$ is the random sample which is uniformly sampled along the straight lines between the points sampled from the input image distribution $\mathbb{P}_I$ and the rendered image distribution $\mathbb{P}_{R'}$.

4. Implementation Details

Before training, every face image has been aligned following the method of [7]. Then we use the networks from [33] pre-trained on the Halen dataset [21] for face segmentation, generating a face region mask for every face image. The face image dataset we use is CelebA [24], and the 3D morphable face model is Basel Face Model [26]. The Regressor network is adopted from [11], which predicts the 3DMM coefficients, pose and lighting parameters, and generates the face shape and coarse texture via the 3DMM model. We set the input image size to 224×224 and the number of vertices to 35,709, the same as [11].

After getting the face shape and coarse texture, the next step is to train the texture refinement module and discriminator. The training loss is defined as:

$$\begin{aligned} L = &\sigma_1(L_{pix}(I, R') + \sigma_2 L_{id}(I, R') + \sigma_3 L_{adv}) \\ &+ \sigma_4(L_{vert}(T, T') + L_{vert}(T_p, \widetilde{T'})), \end{aligned} \tag{8}$$

where $\sigma_2 = 0.2$ and $\sigma_3 = 0.001$ are fixed during training. As for σ_1 and σ_4, we train the texture refinement module starting with a "warm up" manner. Initially, we set $\sigma_1 = 0$ and $\sigma_4 = 1$. After one epoch of warm up, σ_1 is gradually increased to 1 and σ_4 is decreased to 0 accordingly.

During inference, our network can process about 50 images every second in parallel with an NVIDIA 1080Ti GPU.

5. Experimental Results

In this section, we demonstrate our results of the proposed framework and compare the reconstructed 3D faces with prior works. We also uploaded a video rotating the 3D face to YouTube[†] to show our results at large poses.

5.1. Qualitative Comparison

Fig. 5 shows our results compared to the most recent methods [8, 11, 12, 13]. The first row shows the input images and the second row shows our results. The remaining rows demonstrate the results of Deep Facial Details Nets [8], Deep 3D Face Reconstruction [11], GANFIT [12], and the method of Genova *et al.* [13] which introduced a differentiable renderer.

It is worth to mention that in [8, 12], they used their own captured data, which is not publicly available. Specifically, Chen *et al.* [8] captured 366 high-quality 3D scans from 122 different subjects, and Gecer *et al.* [12] trained a progressive GAN to model the distribution of UV representations of 10,000 high-resolution textures, but we only make use of the in-the-wild 2D face image dataset.

The method of [8] used UNets to produce displacement maps for facial details synthesis, and the networks are trained semi-supervised, with labeled 3D face exploited. However, their work mostly focused on local face shape details, the global shape of the reconstructed 3D face are not as good as the locals.

With a large scale UV map dataset, [12] can generate high fidelity textures, but the results are lack of diversity in the respect of lights and holistic colors, as the predicted lights and colors are not so accurate.

Genova *et al.* [13] introduced an end-to-end, unsupervised training procedure, to learn to regress 3DMM coefficients. Deng *et al.* [11] predicted the lights and face pose simultaneously with 3DMM coefficients, which generated a better result under complex lighting situations. Still, our results have more detailed information since theirs are produced by a 3DMM model, which is not able to handle in-the-wild situations very well.

5.2. Quantitative Comparison

To quantitatively evaluate our reconstructed 3D face, we employ the metrics on 2D images, since we do not have 3D ground-truth data. The metrics are computed between the input images and the projected images from face meshes, specifically, when the face mesh is projected to a 2D image, only the face region is projected, Fig. 6 shows an example.

Firstly, we use the L_1 distance, peak signal-to-noise ratio (PSNR) and structural similarity index (SSIM), which measure the results in pixel level. Then we evaluate the results in perception level by calculating the cosine similarity of

[†]https://youtu.be/ZGdkfhsd_Hw

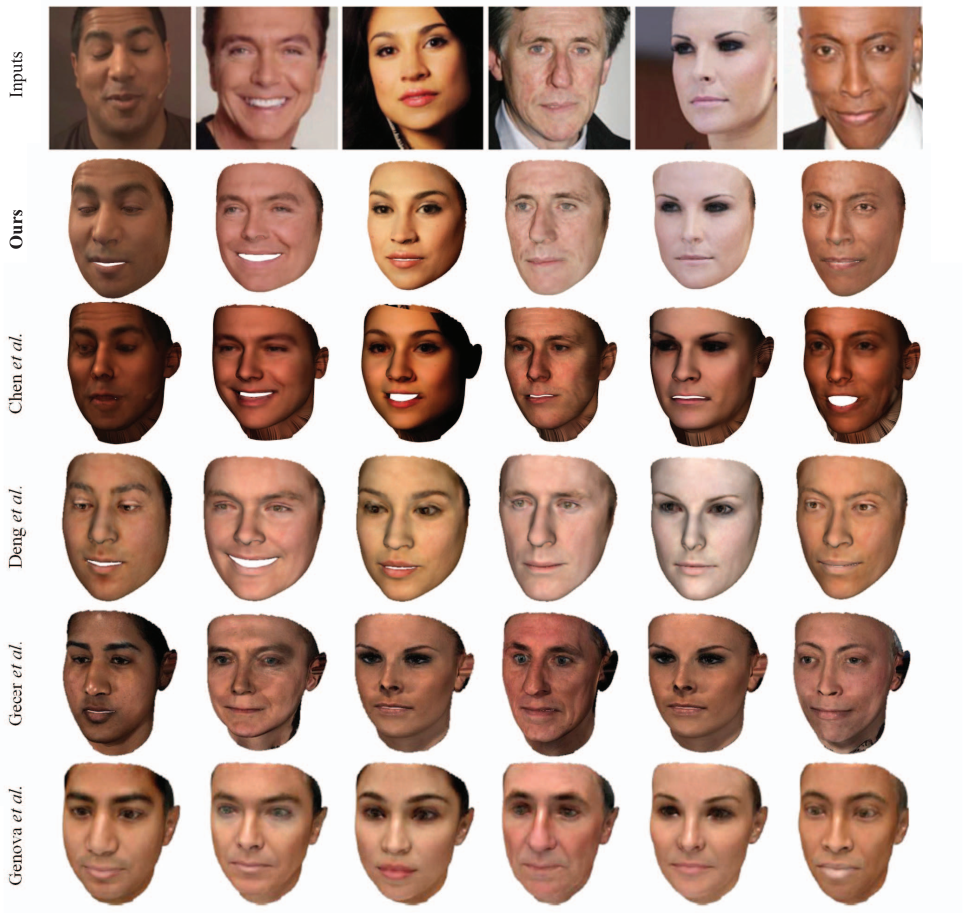

Figure 5: Comparison of our results with other methods. The first row is input images, our results are shown in the second row, and the remaining rows are reconstructed 3D faces obtained by [8, 11, 12, 13], respectively.

feature vectors, which are extracted by two state-of-the-art pretrained face recognition networks, LightCNN [37] and evoLVe [38]. We do not use FaceNet because it appears in our training pipeline, making it unsuitable as an evaluation metric. Since most 3D face reconstruction methods focus on the shape of the face, not the texture, making it difficult to find enough metrics to compare with others. Additionally, we also compare the earth mover's distance between distributions of VGG-Face [25] feature vector similarity from same and different identities on MICC datasets [1]. The quantitative comparison is shown in Tab. 1.

Our results are better than others in multiple evaluation metrics. A lower L_1 distance and higher PSNR and SSIM indicate that our reconstructed 3D face textures is more close to the input images in pixel level. And both state-of-the-art face recognition networks believe that our results and the input images are more likely to be the same person.

In terms of MICC datasets, the facial meshes used in [13] contain ears and neck region, while ours not. And we follow their procedure, only use the rendered images (third image in Fig. 6 demonstrates an example) for computing similarity score. The above factors may lead to lower scores of our method.

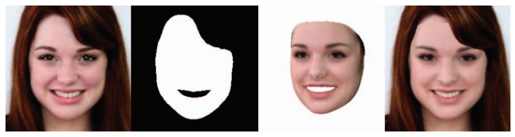

Figure 6: From left to right: input image, face region mask, reconstructed 3D face mesh, and the projected image for quantitative comparison.

5.3. Ablation Study

Fig. 7 demonstrates the ablation study of our approach, where our full model presents a more detailed and realistic texture than its variants. Tab. 2 shows the quantitative metrics. The coarse texture is generated by the Regressor and

		[11]	[13]	Ours	[10]*
CelebA	L_1 distance ↓	0.052	/	**0.034**	/
	PSNR ↑	26.58	/	**29.69**	22.9∼26.5
	SSIM ↑	0.826	/	0.894	0.887∼**0.898**
	LightCNN ↑	0.724	/	**0.900**	/
	evoLVe ↑	0.641	/	**0.848**	/
MICC	VGG-Face same ↓	0.09	0.09	**0.08**	/
	VGG-Face diff. ↑	0.11	**0.32**	0.11	/

Table 1: Quantitative comparison. *: as a reference, we also list the PSNR and SSIM of [10], but it should be noted that their metrics are computed on UV maps since they have built a dataset containing 21,384 real UV maps for training, while we do not use such data.

3DMM model, which is able to produce a basic shape and texture in general, however, the details such as lentigo or eyes are not faithfully predicted.

Starting by L_{pix} and L_{id}, the networks are no longer restricted to a 3DMM model, and predicts higher fidelity face skin and eyes. With the help of adversarial training, the results become less blurry and more realistic. Finally, we construct our full model by adding the L_{vert}, the predictions contain more details, and looked very similar to the input image.

We also replace GCNs with fully-connected layers or convolutional layers on unwrapped UV space, and the performance are no better than GCNs, as shown in Fig. 7 and Tab. 2. We reach to the similar conclusion as [39], that using FCs or CNNs on UV spaces leads to a large number of parameters in the network and does not utilize the spatial information of the 3D facial structure.

Losses L_{pix}	L_{id}	L_{adv}	L_{vert}	PSNR	SSIM	LightCNN	evoLVe
				26.58	0.826	0.724	0.641
✓	✓			28.57	0.863	0.828	0.738
✓	✓	✓		29.30	0.872	0.840	0.755
✓	✓	✓	✓	**29.69**	**0.894**	**0.900**	**0.848**
Fully-connected				25.88	0.820	0.629	0.544
Convolutional				27.54	0.848	0.798	0.696

Table 2: Metrics on ablation study, higher is better.

To improve the results, we train and test our proposed networks on a higher resolution image dataset, CelebA-HQ [19], as well. Higher resolution helps to reduce the checkerboard-like artifacts and produce better results. A comparison is shown in Fig. 8.

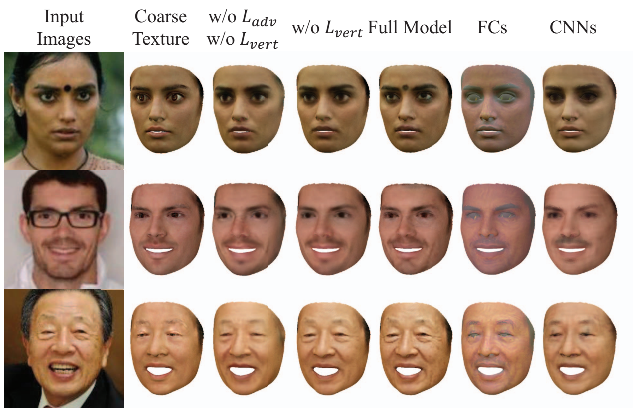

Figure 7: Ablation study on different losses of our proposed approach, the full model leads to a better result than others.

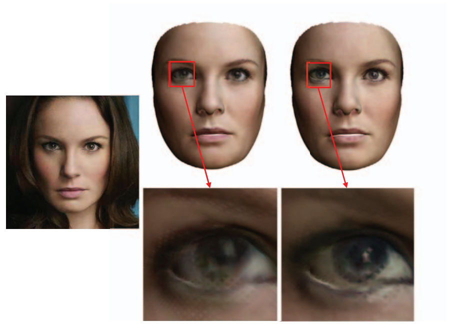

Figure 8: Comparison with higher resolution image. From left to right: the input image, result generated from resolution of 224×224 and 512×512.

6. Conclusions

In this paper, we present a novel 3D face reconstruction method, which produces face shapes with high fidelity textures from single-view images. To the best of our knowledge, we are the first to use graph convolutional networks to generate high fidelity face textures from the single images. Compared with other methods, our method does not require capturing high-resolution face texture datasets but can generate realistic face textures with just a 3DMM model. In the future, we will try to minimize the checkerboard-like artifacts and generate more detailed face shapes and expressions using graph convolutional networks as well.

Acknowledgments

We thank anonymous reviewers for their valuable comments. The work was supported by National Key R&D Program of China (2018YFB1004300), NSF China (No. 61772462, No. 61572429, No. U1736217) and the 100 Talents Program of Zhejiang University.

References

[1] Andrew D. Bagdanov, Alberto Del Bimbo, and Iacopo Masi. The florence 2d/3d hybrid face dataset. In *Proceedings of the 2011 Joint ACM Workshop on Human Gesture and Behavior Understanding*, J-HGBU '11, page 79–80, New York, NY, USA, 2011. ACM.

[2] Volker Blanz and Thomas Vetter. Face recognition based on fitting a 3d morphable model. *IEEE Transactions on pattern analysis and machine intelligence*, 25(9):1063–1074, 2003.

[3] Volker Blanz, Thomas Vetter, et al. A morphable model for the synthesis of 3d faces. In *Siggraph*, volume 99, pages 187–194, 1999.

[4] James Booth, Anastasios Roussos, Allan Ponniah, David Dunaway, and Stefanos Zafeiriou. Large scale 3d morphable models. *International Journal of Computer Vision*, 126(2-4):233–254, 2018.

[5] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203*, 2013.

[6] Chen Cao, Yanlin Weng, Shun Zhou, Yiying Tong, and Kun Zhou. Facewarehouse: A 3d facial expression database for visual computing. *IEEE Transactions on Visualization and Computer Graphics*, 20(3):413–425, 2013.

[7] Dong Chen, Gang Hua, Fang Wen, and Jian Sun. Supervised transformer network for efficient face detection. In *European Conference on Computer Vision*, pages 122–138. Springer, 2016.

[8] Zhang Chen, Guli Zhang, Ziheng Zhang, Kenny Mitchell, Jingyi Yu, et al. Photo-realistic facial details synthesis from single image. *arXiv preprint arXiv:1903.10873*, 2019.

[9] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in neural information processing systems*, pages 3844–3852, 2016.

[10] Jiankang Deng, Shiyang Cheng, Niannan Xue, Yuxiang Zhou, and Stefanos Zafeiriou. Uv-gan: Adversarial facial uv map completion for pose-invariant face recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7093–7102, 2018.

[11] Yu Deng, Jiaolong Yang, Sicheng Xu, Dong Chen, Yunde Jia, and Xin Tong. Accurate 3d face reconstruction with weakly-supervised learning: From single image to image set. In *IEEE Computer Vision and Pattern Recognition Workshops*, 2019.

[12] Baris Gecer, Stylianos Ploumpis, Irene Kotsia, and Stefanos Zafeiriou. Ganfit: Generative adversarial network fitting for high fidelity 3d face reconstruction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1155–1164, 2019.

[13] Kyle Genova, Forrester Cole, Aaron Maschinot, Aaron Sarna, Daniel Vlasic, and William T Freeman. Unsupervised training for 3d morphable model regression. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8377–8386, 2018.

[14] Thomas Gerig, Andreas Morel-Forster, Clemens Blumer, Bernhard Egger, Marcel Luthi, Sandro Schönborn, and Thomas Vetter. Morphable face models-an open framework. In *2018 13th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2018)*, pages 75–82. IEEE, 2018.

[15] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 315–323, 2011.

[16] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. In *Advances in neural information processing systems*, pages 5767–5777, 2017.

[17] David K Hammond, Pierre Vandergheynst, and Rémi Gribonval. Wavelets on graphs via spectral graph theory. *Applied and Computational Harmonic Analysis*, 30(2):129–150, 2011.

[18] Patrik Huber, Guosheng Hu, Rafael Tena, Pouria Mortazavian, P Koppen, William J Christmas, Matthias Ratsch, and Josef Kittler. A multiresolution 3d morphable face model and fitting framework. In *Proceedings of the 11th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications*, 2016.

[19] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*, 2017.

[20] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[21] Vuong Le, Jonathan Brandt, Zhe Lin, Lubomir Bourdev, and Thomas S Huang. Interactive facial feature localization. In *European conference on computer vision*, pages 679–692. Springer, 2012.

[22] Martin D Levine and Yingfeng Chris Yu. State-of-the-art of 3d facial reconstruction methods for face recognition based on a single 2d training image per person. *Pattern Recognition Letters*, 30(10):908–913, 2009.

[23] Tianye Li, Timo Bolkart, Michael J Black, Hao Li, and Javier Romero. Learning a model of facial shape and expression from 4d scans. *ACM Transactions on Graphics (TOG)*, 36(6):194, 2017.

[24] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of the IEEE international conference on computer vision*, pages 3730–3738, 2015.

[25] Omkar M Parkhi, Andrea Vedaldi, and Andrew Zisserman. Deep face recognition. 2015.

[26] Pascal Paysan, Reinhard Knothe, Brian Amberg, Sami Romdhani, and Thomas Vetter. A 3d face model for pose and illumination invariant face recognition. In *2009 Sixth IEEE International Conference on Advanced Video and Signal Based Surveillance*, pages 296–301. Ieee, 2009.

[27] Anurag Ranjan, Timo Bolkart, Soubhik Sanyal, and Michael J Black. Generating 3d faces using convolutional mesh autoencoders. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 704–720, 2018.

[28] Elad Richardson, Matan Sela, and Ron Kimmel. 3d face reconstruction by learning from synthetic data. In *2016 Fourth International Conference on 3D Vision (3DV)*, pages 460–469. IEEE, 2016.

[29] Elad Richardson, Matan Sela, Roy Or-El, and Ron Kimmel. Learning detailed face reconstruction from a single image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1259–1268, 2017.

[30] Sami Romdhani and Thomas Vetter. Estimating 3d shape and texture using pixel intensity, edges, specular highlights, texture constraints and a prior. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 2, pages 986–993. IEEE, 2005.

[31] Soubhik Sanyal, Timo Bolkart, Haiwen Feng, and Michael J Black. Learning to regress 3d face shape and expression from an image without 3d supervision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7763–7772, 2019.

[32] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 815–823, 2015.

[33] Tianyang Shi, Yi Yuan, Changjie Fan, Zhengxia Zou, Zhenwei Shi, and Yong Liu. Face-to-parameter translation for game character auto-creation. *arXiv preprint arXiv:1909.01064*, 2019.

[34] Ayush Tewari, Michael Zollhofer, Hyeongwoo Kim, Pablo Garrido, Florian Bernard, Patrick Perez, and Christian Theobalt. Mofa: Model-based deep convolutional face autoencoder for unsupervised monocular reconstruction. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1274–1283, 2017.

[35] Anh Tuan Tran, Tal Hassner, Iacopo Masi, and Gérard Medioni. Regressing robust and discriminative 3d morphable models with a very deep neural network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5163–5172, 2017.

[36] Fanzi Wu, Linchao Bao, Yajing Chen, Yonggen Ling, Yibing Song, Songnan Li, King Ngi Ngan, and Wei Liu. Mvf-net: Multi-view 3d face morphable model regression. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 959–968, 2019.

[37] Xiang Wu, Ran He, Zhenan Sun, and Tieniu Tan. A light cnn for deep face representation with noisy labels. *IEEE Transactions on Information Forensics and Security*, 13(11):2884–2896, 2018.

[38] Jian Zhao, Jianshu Li, Xiaoguang Tu, Fang Zhao, Yuan Xin, Junliang Xing, Hengzhu Liu, Shuicheng Yan, and Jiashi Feng. Multi-prototype networks for unconstrained set-based face recognition. In *IJCAI*, 2019.

[39] Yuxiang Zhou, Jiankang Deng, Irene Kotsia, and Stefanos Zafeiriou. Dense 3d face decoding over 2500fps: Joint texture & shape convolutional mesh decoders. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1097–1106, 2019.