MoFaNeRF: Morphable Facial Neural Radiance Field - Supplementary Material

Yiyu Zhuang*, Hao Zhu*, Xusen Sun, and Xun Cao†

Nanjing University, Nanjing, China
{yiyu.zhuang, xusensun}@smail.nju.edu.cn {zhuhaoese, caoxun}@nju.edu.cn

1 Supplementary Materials

1.1 Overview

The supplementary material contains a video available at `https://neverstopzyy.github.io/mofanerf` and additional descriptions. The video shows a brief overview of our method and the animation of rigging and editing. The additional content contains more results of the image-based fitting (Section 1.3), failure cases (Section 1.4), comparison with previous reconstruction-based view synthesis (Section 1.5), randomly generated faces (Section 1.6), the network architecture (Section 1.8), details about the training (Section 1.9), and the source of used face images (Section 1.10).

1.2 Animation of Rigging and Editing

After the face is fitted or generated, it can be rigged by driving the expression code ϵ, and be edited by changing shape code β and appearance code α. The animation of rigging and editing results are shown in the supplementary video (Part 3 and 4). We can see that the face can be driven by the expression code extracted from a RGB video, and the face can morph smoothly in the dimensions of shape, appearance and expression.

1.3 More Results of image-based fitting

We show more faces fitted to a single image in Figure 1 and Figure 3, which is the extension of Figure 7 in the main paper.

1.4 Failure Cases in Fitting

We show some failure cases of the image-based fitting in Figure 2. The first column on the left shows that the fitting results are bad in the extreme lighting. As our model is trained in the images with relatively diffused lighting, large areas of shadow will interfere with our fitting. Lighting models may be introduced in

* These authors contributed equally to this work.
† Xun Cao is the corresponding author.

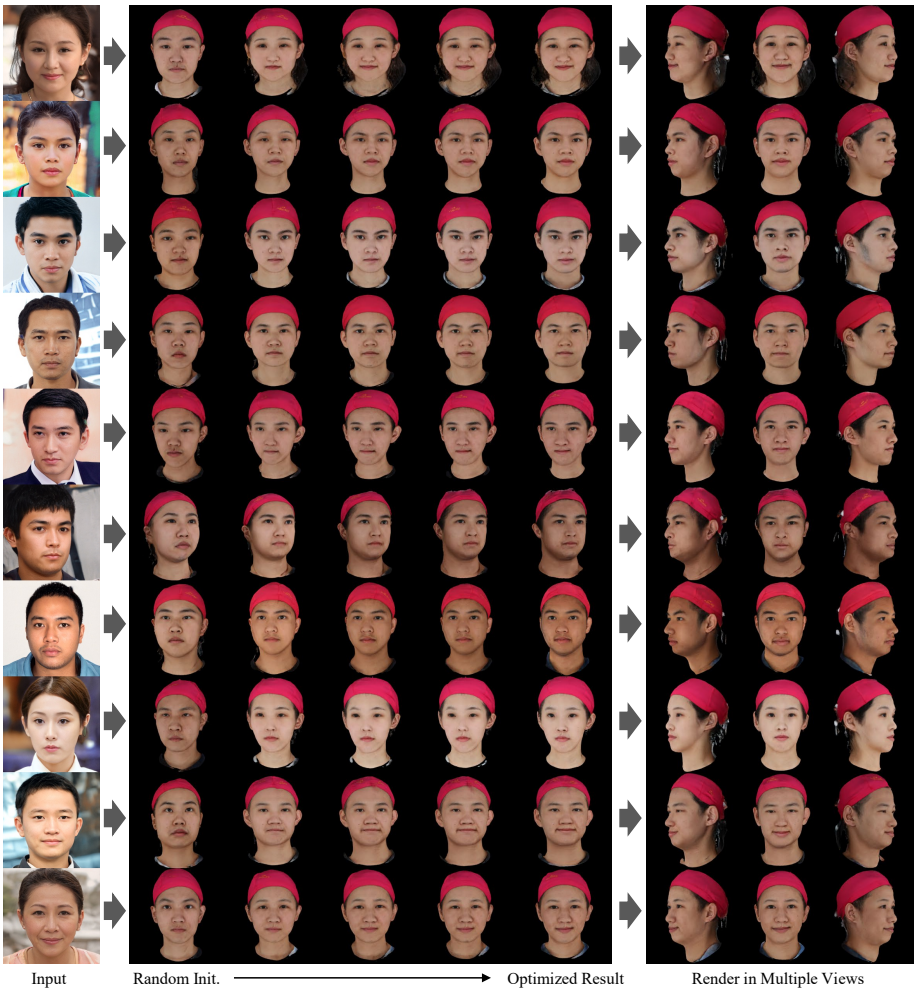

Fig. 1. More fitting results of MoFaNeRF to a single-view image based on FaceScape model.

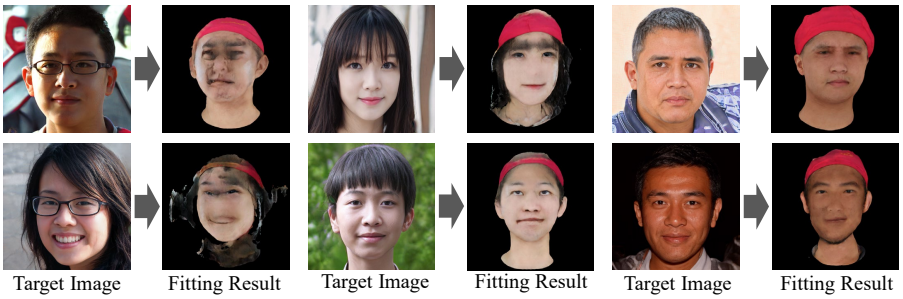

Fig. 2. Failure cases of fitting MoFaNeRF to a single image.

Fig. 3. More fitting results of MoFaNeRF to a single-view image based on HeadSpace model.

future work to improve the generalization ability for complex lighting conditions. The second column from the left shows that the fitting may fail for the faces with large occluded regions. Fitting MoFaNeRF to an occluded face is still a challenging task to be solved. The third column from the left shows that the fitting results degraded for the faces that are quite different from the FaceScape dataset in shape (top) or skin color (bottom). The generalization of image-based fitting still needs to be improved.

1.5 Fitting v.s. Single-View Reconstruction

As shown in Figure 4, we compare our method with four state-of-the-art Single-View Reconstruction(SVR) methods[11,9,3,2] in rendering performance. These methods take the single-view image as input and predict the mesh with texture. The images are rendered from the predicted meshes in the frontal view and $\pm 60^\circ$ side views. Please note that FaceScape-fit[11], 3DDFAv2[3] and DECA[2]

Fig. 4. We compare our fitting and rendering result with SOTA single-view reconstruction methods (SVR). In the red circles, we can see that the inaccurately predicted shape of the nose leads to artifacts in the side view. 3DDFAv2, FaceScape-fit, and DECA commonly contain artifacts on the cheeks due to the misalignment of the predicted shape and the source image. Besides, all four methods cannot align ears well, so no texture is assigned to ears.

Fig. 5. More random generated results by our models. We visualize them in three views with yaw angles in $[-60°, 0°, 60°]$ and pitch angles in $0°$.

reconstructed the full head, however, their textures come from the source image and only facial textures are assigned. Therefore, we only render the regions with texture for these three methods.

We can see that the inaccurately predicted shape of FaceScape-fit[11], 3DDFA-v2[3] and DECA[2] leads to the artifacts in the side views, as shown in the red dotted circles. Besides, these methods commonly contain wrong scratches on the cheeks due to the misalignment of the predicted shape and the source image. Though the MGCNet doesn't contain the scratches problems, its texture tends to be a mean texture with less detail. We can also observe that in some cases the shape of the nose is unfaithful. Besides, all four methods cannot align ears well, so no texture is assigned to ears. By contrast, our rendering results contain fewer artifacts and are more plausible in the side views. The ears are also rendered in our method, which makes the side-view rendering complete.

1.6 Results of Random Generation

Some randomly generated faces are shown in Figure 5. We can see that our model covers a wide range of facial shapes, appearances and expressions.

1.7 Disentanglement between appearance and geometry

In addition to Fig. 3 of the main paper, we supplement the experiment in Fig. 6 to extract the shape from the radiance fields, and report the chamfer distance to quantitatively measure the shape consistency. The figure and quantitative results show that the extracted shape is changed only when the shape code is changed, which indicates the effective disentanglement of shape from appearance and expression.

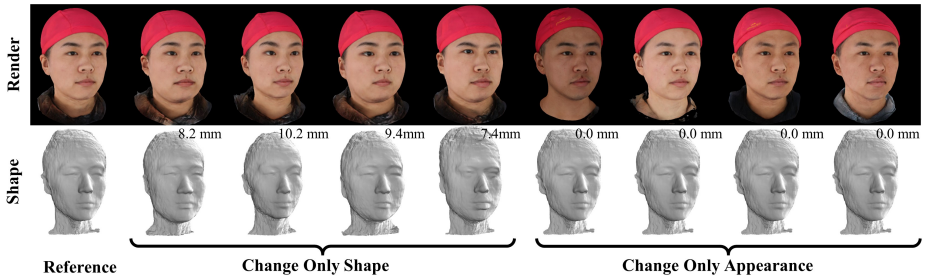

Fig. 6. Extracted shape for verifying disentanglement.

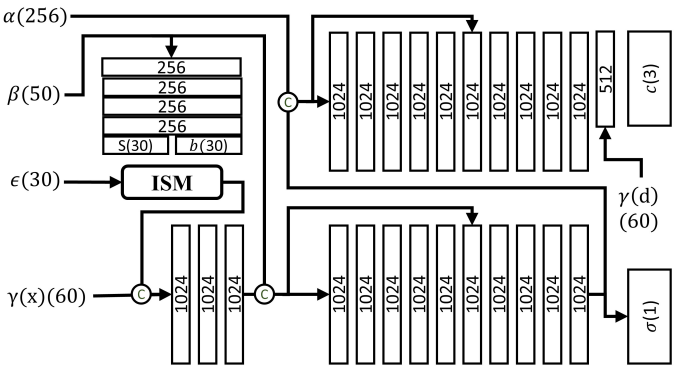

Fig. 7. The detailed parameters of the network in MoFaNeRF. The number in brackets indicate the length of the tensor.

1.8 Parameters of Network

The parameters of our network are shown in Figure 7. The boxes represent the full connection layer, where the numbers represent the number of neurons. The circles with C inside represent the concatenating operation between tensors. The numbers follow the parameter means the dimension of this parameter. α, β, ϵ are the parameters of appearance, shape and expression. $\gamma(\mathbf{x})$ and $\gamma(\mathbf{d})$ mean the position encoding of the position code $\mathbf{x}$ and viewing direction $\mathbf{d}$. σ and $\mathbf{c}$ are the density and color that form the radiance field. The parameters of the TEM module are shown in Table 1.

1.9 Details of Training and Data

Implement details. Our model is implemented on PyTorch [1]. In our experiment, 1024 rays are sampled in an iteration, each with 64 sampled points in the coarse volume and additional 64 in the fine volume. The strategy of hierarchical volume sampling is used to refine coarse volume and fine volume simultaneously, as defined in NeRF[8]. The resolution of the images rendered by MoFaNeRF is 256, and the RefineNet takes the image rescaled to 512×512 as input, and synthesizes the final image in 512×512.

Table 1. The detailed parameters of the TEM in MoFaNeRF. All convolution layers and linear layers are followed by Leaky ReLU[7] with negative slope of 0.2 and 0.1 respectively, except for layers "mu" and "std". 'Repara.': means the reparameterization method[6,10] to produce latent code from the distribution $\mathcal{N}(\mu, \sigma)$. k: kernel size ($k \times k$). s: stride in both horizontal and vertical directions. p: padding size ($p \times p$). c: number of output channels. d: output spatial dimension ($d \times d$). 'Conv': convolution layer. 'Linear': fully connected layer. 'Flatten': flatten layer.

Name	Type	input	(k,s,p)	c	d
conv1	Conv	textureMap	(4,2,1)	32	256
conv2	Conv	conv1	(4,2,1)	32	128
conv3	Conv	conv2	(4,2,1)	32	64
conv4	Conv	conv3	(4,2,1)	32	32
conv5	Conv	conv4	(4,2,1)	64	16
conv6	Conv	conv5	(4,2,1)	128	8
conv7	Conv	conv6	(4,2,1)	256	4
flat0	Flatten	conv7	–*	4096	1
line1	Linear	flat0	–	512	1
mu	Linear	line1	–	256	1
logstd	Linear	line1	–	256	1
para.	Repara.	(mu,logstd)	–	256	1
line2	Linear	para.	–	256	1
line3	Linear	line2	–	256	1
app.	Linear	line3	–	256	1

* '–' means meaningless parameters.

We use the Adam optimizer[5] with the initial learning rate as 5×10^{-4}, decayed exponentially as 2×10^{-5}, $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-7}$. Our model is trained for roughly $400k$ iterations, which takes about 2 days on dual NVIDIA GTX3090 GPUs.

Details about data preparation. Taking face orientation as the reference direction, we evenly select 6 pitch angles from $-30°$ to $45°$ and 20 yaw angles from $-90°$ to $+90°$ degrees for rendering in 120 viewpoints. The samples of 120 views are shown in Figure 8 and Figure 9. We use all the training data to train the network of MoFaNeRF, and randomly select $24,000$ rendering results to train the RefineNet.

Initially, we plan to use the raw scanned multi-view images released by FaceScape [11,12], however, we find the camera locations are not uniform for all these 7180 tuples of images. We contacted the authors of FaceScape and learned that the reason was that the capturing took place in two locations, where the camera setups and parameters were changed several times. Therefore, we use the multi-view images to color the raw scanned models, then render them according to the viewpoints as set above to obtain high-fidelity multi-view images with uniform camera parameters.

Fig. 8. Multi-view images are generated by FaceScape in 120 views with 6 pitch angles in $[-30^\circ \sim +45^\circ]$ and 20 yaw angles in $[-90^\circ \sim +90^\circ]$.

Fig. 9. Multi-view images are generated by HeadSpace in 120 views with 6 pitch angles in $[-30^\circ \sim +45^\circ]$ and 20 yaw angles in $[-90^\circ \sim +90^\circ]$.

1.10 Face Image Source

To avoid portrait infringement, we used some synthesized 'in-the-wild' face images for testing our model. The source images in Figure 1, 4, 9, 10 are synthesized by StyleGANv2[4], which is released under Nvidia source code license. So the use of these virtual portraits will not raise infringement issues. We have signed the license agreement with the authors of FaceScape [11,12] to obtain the permission to use the dataset for non-commercial research purpose, and the permission to publish the subjects of 12, 17, 40, 49, 57, 92, 97, 168, 211, 212, 215, 234, 260, 271, 326 in Figure 2, 3, 5, 6, 7, 8 of this paper.

References

1. Pytorch. https://pytorch.org/
2. Feng, Y., Feng, H., Black, M.J., Bolkart, T.: Learning an animatable detailed 3d face model from in-the-wild images. ToG **40**(4), 1–13 (2021)
3. Guo, J., Zhu, X., Yang, Y., Yang, F., Lei, Z., Li, S.Z.: Towards fast, accurate and stable 3d dense face alignment. In: ECCV. pp. 152–168 (2020)
4. Karras, T., Laine, S., Aittala, M., Hellsten, J., Lehtinen, J., Aila, T.: Analyzing and improving the image quality of stylegan. In: CVPR. pp. 8110–8119 (2020)
5. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. ICLR (2015)
6. Kingma, D.P., Welling, M.: Auto-encoding variational bayes. In: ICLR (2014)
7. Maas, A.L., Hannun, A.Y., Ng, A.Y., et al.: Rectifier nonlinearities improve neural network acoustic models. In: ICML. vol. 30, p. 3. Citeseer (2013)
8. Mildenhall, B., Srinivasan, P.P., Tancik, M., Barron, J.T., Ramamoorthi, R., Ng, R.: Nerf: Representing scenes as neural radiance fields for view synthesis. In: ECCV. pp. 405–421 (2020)
9. Shang, J., Shen, T., Li, S., Zhou, L., Zhen, M., Fang, T., Quan, L.: Self-supervised monocular 3d face reconstruction by occlusion-aware multi-view geometry consistency. In: ECCV. pp. 53–70 (2020)
10. Wang, Z., Bagautdinov, T., Lombardi, S., Simon, T., Saragih, J., Hodgins, J., Zollhofer, M.: Learning compositional radiance fields of dynamic human heads. In: CVPR. pp. 5704–5713 (2021)
11. Yang, H., Zhu, H., Wang, Y., Huang, M., Shen, Q., Yang, R., Cao, X.: Facescape: a large-scale high quality 3d face dataset and detailed riggable 3d face prediction. In: CVPR (2020)
12. Zhu, H., Yang, H., Guo, L., Zhang, Y., Wang, Y., Huang, M., Shen, Q., Yang, R., Cao, X.: Facescape: 3d facial dataset and benchmark for single-view 3d face reconstruction. arXiv preprint arXiv:2111.01082 (2021)