

Figure 1: More results from CFD. The first column are the input images.

Appendices

A. More Results From CFD

We present more results from CFD test set in Figure 1.

B. More Results From FaceWarehouse

We present more results from Facewarehouse test set in Figure 3.

C. More Results from Images in the Wild

Though our training set only contains images captured in a lab setting with frontal faces and uniform lighting, we show that our trained model can work on more challenging in-the-wild images in Figure 4.

D. More Results on Continuous Editing

As we use expression coefficients as conditions, we can trivially manipulate faces continuously. We show more results on continuous editing in the submitted video.

E. More Details in $\mathcal{L}_{\text{real}}$, $\mathcal{L}_{\text{pair}}$ and $\mathcal{L}_{\text{iden}}$

As mentioned in the paper, our min-max game objective is composed of three terms: the realism term $\mathcal{L}_{\text{real}}$, the pair-wise term $\mathcal{L}_{\text{pair}}$ and the identity term $\mathcal{L}_{\text{iden}}$. These three loss terms are calculated from three discriminators D_{real} , D_{pair} and D_{iden} respectively using LSGAN[3]. Let $\mathbf{T}_{i,p}^{\text{real}}$, $\mathbf{T}_{i,p}^{\text{fake}}$ be the real and fake textures of identity \mathbf{a}_i under the expression \mathbf{e}_p . Let $\bar{L}_2(\mathbf{x}) = \|\mathbf{x} - 1\|^2$, $L_2(\mathbf{x}) = \|\mathbf{x}\|^2$, our loss terms are calculated as follows:

- $\mathcal{L}_{\text{real}}$ is used to differentiate real images and fake generated images:

$$\mathcal{L}_{\text{real}} = \bar{L}_2(D_{\text{real}}(\mathbf{T}_{i,p}^{\text{real}})) + L_2(D_{\text{real}}(\mathbf{T}_{i,p}^{\text{fake}})). \quad (1)$$

- $\mathcal{L}_{\text{pair}}$ is used to differentiate matched pairs of real texture and expressions $(\mathbf{T}_{i,p}^{\text{real}}, \mathbf{e}_p)$ from matched pairs of fake texture and expressions $(\mathbf{T}_{i,p}^{\text{fake}}, \mathbf{e}_p)$ and mismatched pairs of real texture and expressions $(\mathbf{T}_{i,p}^{\text{real}}, \mathbf{e}_r)$, where \mathbf{e}_r is a random expression:

$$\begin{aligned} \mathcal{L}_{\text{pair}} = & 2\bar{L}_2(D_{\text{pair}}(\mathbf{T}_{i,p}^{\text{real}}, \mathbf{e}_p)) \\ & + L_2(D_{\text{pair}}(\mathbf{T}_{i,p}^{\text{fake}}, \mathbf{e}_p)) + L_2(D_{\text{pair}}(\mathbf{T}_{i,p}^{\text{real}}, \mathbf{e}_r)). \end{aligned} \quad (2)$$

where we multiply the first term by 2 to prevent the discriminator from simply producing a small value.

- $\mathcal{L}_{\text{iden}}$ is used to differentiate real textures with the same identity $(\mathbf{T}_{i,p}^{\text{real}}, \mathbf{T}_{i,q}^{\text{real}})$, from real and fake textures with the same identity $(\mathbf{T}_{i,p}^{\text{real}}, \mathbf{T}_{i,q}^{\text{fake}})$, and real textures with different identities $(\mathbf{T}_{i,p}^{\text{real}}, \mathbf{T}_{j,q}^{\text{real}})$, where p, q index random expressions and i, j index different identities.

$$\begin{aligned} \mathcal{L}_{\text{iden}} = & 2\bar{L}_2(D_{\text{iden}}(\mathbf{T}_{i,p}^{\text{real}}, \mathbf{T}_{i,q}^{\text{real}})) \\ & + L_2(D_{\text{iden}}(\mathbf{T}_{i,p}^{\text{real}}, \mathbf{T}_{i,q}^{\text{fake}})) + L_2(D_{\text{iden}}(\mathbf{T}_{i,p}^{\text{real}}, \mathbf{T}_{j,q}^{\text{real}})). \end{aligned} \quad (3)$$

F. Network Architectures

For our texture branch generator, we use pix2pix [2] and attention map [4]. For our texture branch input, we concatenate input texture(3), normal(3), area deformation(1), curvature(1), normal difference(3), position different(3) and noise (1) together, thus the total number of channels for our input is 15. For our output, we use a separate attention map for each R,G,B channel, therefore the number of channels for our output is 6. To avoid being saturated in the gradients, we do not use any sigmoid or tanh activations. For the hidden layers in the middle, we use the UNet structure with skip link. For the encoder, we use convolutional layers with

filter size 4, stride 2 and padding 1 for downsampling. For the decoder, we use bilinear upsampling followed by a convolutional layer with filter size 3, stride 1 and padding 1 for upsampling. Following the notation from [2], we use Ck denote Convolution-BatchNorm-ReLU layer with k filters.

encoder:

$C32 - C64 - C64$

decoder:

$C64 - C32 - C6$

All ReLUs in the encoder are leaky, with slope 0.2. All ReLUs in the decoder are not leaky.

G. More Details in the Global Branch

The goal of the global branch is to blend the rendered image seamlessly into the background. We first generate a margin by calling the OpenCV dilate function with a kernel size of 12. The our global branch takes the rendered face, input image and regions outside of the margin as input to hallucinate inside. Sometimes there is still an observable boundary in which case we apply image blending. We blend the image based on the vertex distance d from the source expression mesh to the target expression mesh. The blending alpha is determined heuristically as $\exp(d^2/4)$.

H. More Comparison with Texture Mapping

We show more examples of the difference between our approach and direct texture mapping approach. To manipulate expression in the image, one can change only the underlying shape without substantially changing the texture like [1]. However this can result in many artifacts, especially when the source and target expressions significantly differ. For example, in Fig. 2, if one directly uses the texture extracted from the source image and renders it with a smiling face shape, the missing crease and teeth and image distortion make the result less realistic. Our texture branch learns to reconstruct these missing parts and change the local appearance near the eyes, which makes the resulting image look natural.

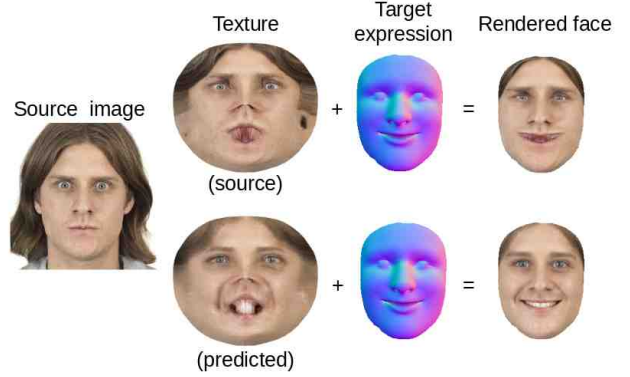


Figure 2: Demonstration of texture branch. This example shows transferring to a smiling expression. **Top:** the direct texture mapping approach [1]. **Bottom:** rendering using our predicted texture with the same smiling shape.

References

- [1] C. Cao, Y. Weng, S. Zhou, Y. Tong, and K. Zhou. Faceware-house: A 3d facial expression database for visual computing. *IEEE Transactions on Visualization and Computer Graphics*, 20(3):413–425, 2014. 2
- [2] P. Isola, J. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5967–5976, 2017. 1, 2
- [3] X. Mao, Q. Li, H. Xie, R. Y. K. Lau, and Z. Wang. Multi-class generative adversarial networks with the L2 loss function. *CoRR*, abs/1611.04076, 2016. 1
- [4] A. Pumarola, A. Agudo, A. M. Martinez, A. Sanfeliu, and F. Moreno-Noguer. Ganimation: Anatomically-aware facial animation from a single image. In *Proceedings of the European Conference on Computer Vision*, pages 835–851, 2018. 1

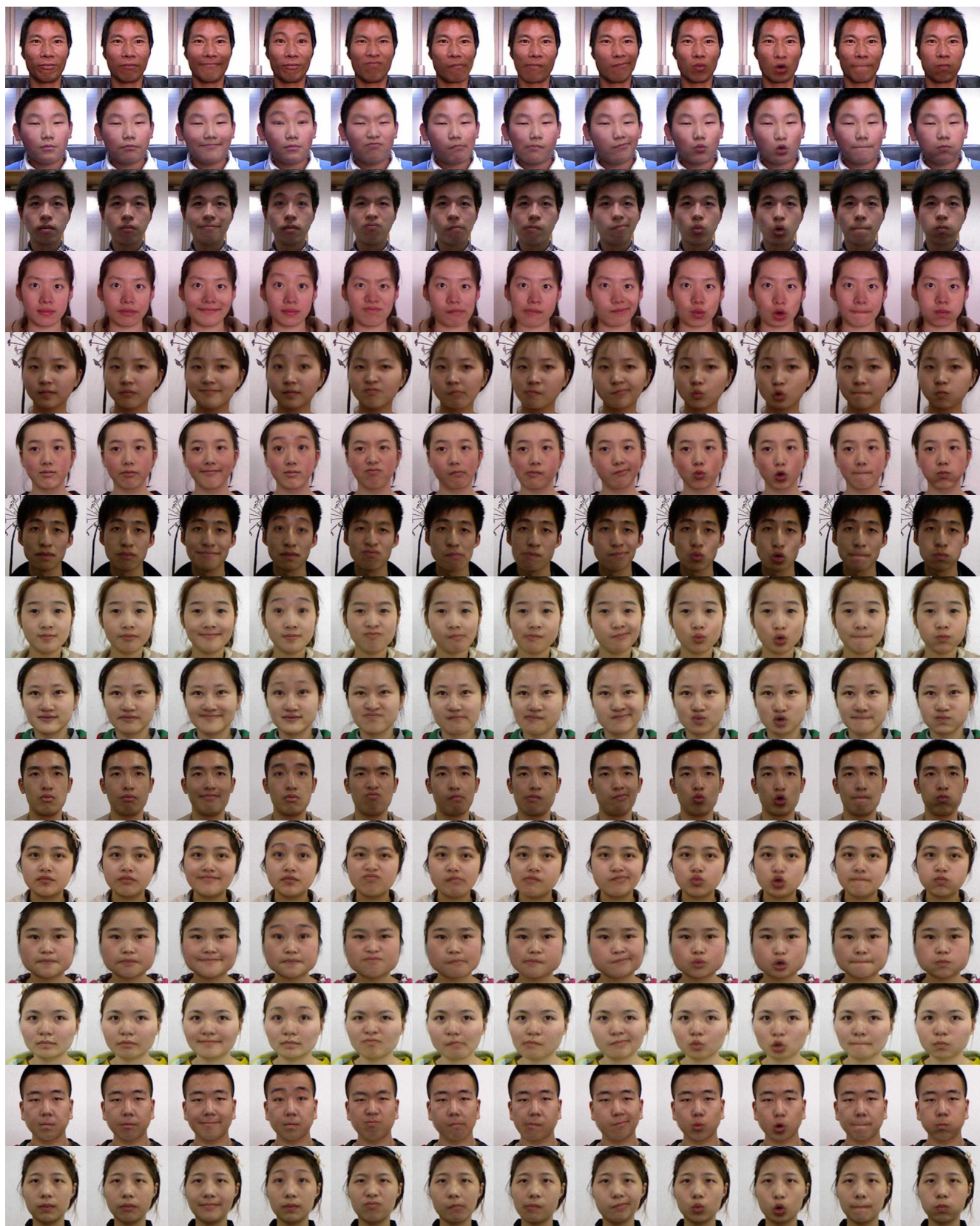


Figure 3: More results from FaceWarehouse. The first column are the input images.



Figure 4: More results from in-the-wild images. Note that our training set only contains images captured with frontal faces and good lighting, our trained model can work on some challenging in-the-wild images as well.