Supplementary Material of Paper 5664

Anonymous CVPR submission

Paper ID 5664

1. Implementation Details

Here we describe more details of the model implementation. As introduced in Section 3, the 3D proxy provides depth, texture, and also light/viewpoint predicted from Φ^l, Φ^ω, respectively. During the learning procedure of ARN and GRN, we directly use the l_o, ω_o from pre-trained Φ^l, Φ^ω proxy for rendering. Then, we jointly train Φ^l, Φ^ω with ARN and GRN in the final mutual learning process.

For the lighting Λ and rasterization Π operation used in Eqns. (1) and (2), we follow a same setting as Unsup3D [6]. Here we provide more details. The lighting Λ is conducted at the canonical view, where we shade the canonical albedo a with the predicted light l by the Lambertian function f_{lam}. Concretely, we first transform the predicted canonical depth d to the normal n, then perform $\mathbf{S} = f_{lam}(n, l)$ to get a shading map $\mathbf{S}$. Finally, the canonical texture t is obtained by $t = \mathbf{S} \odot a$. For the rasterization Π function, we set the Field of View (FOV) of the camera as 10° to calculate the camera matrix. The corresponding projection and warping operations is implemented by neural mesh renderer [4].

2. Details of the Experiment on MICC

In Fig. 1 of the main paper, we perform a cross-view geometry analysis on MICC [1] dataset. MICC is a 3D face dataset containing 53 subjects with its ground truth 3D mesh acquired from a structured light scanning system. Similar to [3], we render a provided face to -45°, 0°, and +45° respectively, each of which contains 3 rendered faces. To evaluate the performance of cross-view geometry modelling, we use the image of one pose for reconstruction and measure the modelled geometry on the other two poses. For instance, we first use the image of 0° as input to recover the geometry, and then calculate the errors with ground truth geometry of -45° and +45°, respectively. The errors are then averaged as the final result. In the experiment, we evaluate our method, Unsup3D [6] and LAP [7], each of which is directly tested on MICC using the pre-trained weights without fine-tuning.

We further show examples in Fig. 1. As illustrated, our method models better facial geometry and organ shapes, while LAP [7] cannot precisely recover the corresponding face structure.

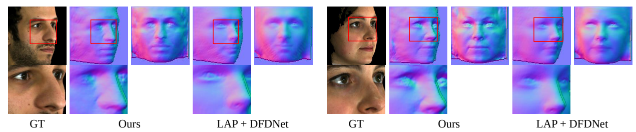

Figure 1. Visual comparison on MICC dataset.

3. More Results

In Fig. 2, we show more results and comparisons with the state-of-the-art methods. Our method predicts finer details and more precise facial shapes against the degraded images.

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*, pages 79–80, 2011. 1

[2] Yao Feng, Haiwen Feng, Michael J Black, and Timo Bolkart. Learning an animatable detailed 3d face model from in-the-wild images. *ACM Transactions on Graphics (TOG)*, 40(4):1–13, 2021. 2

[3] Jianzhu Guo, Xiangyu Zhu, Yang Yang, Fan Yang, Zhen Lei, and Stan Z Li. Towards fast, accurate and stable 3d dense face alignment. In *ECCV*, 2020. 1

[4] Hiroharu Kato, Yoshitaka Ushiku, and Tatsuya Harada. Neural 3d mesh renderer. In *CVPR*, pages 3907–3916, 2018. 1

[5] Xiaoming Li, Chaofeng Chen, Shangchen Zhou, Xianhui Lin, Wangmeng Zuo, and Lei Zhang. Blind face restoration via deep multi-scale component dictionaries. In *European Conference on Computer Vision*, pages 399–415, 2020. 2

[6] Shangzhe Wu, Christian Rupprecht, and Andrea Vedaldi. Unsupervised learning of probably symmetric deformable 3d objects from images in the wild. In *CVPR*, pages 1–10, 2020. 1, 2

[7] Zhenyu Zhang, Yanhao Ge, Renwang Chen, Ying Tai, Yan Yan, Jian Yang, Chengjie Wang, Jilin Li, and Feiyue Huang. Learning to aggregate and personalize 3d face from in-the-wild photo collection. In *CVPR*, pages 14214–14224, 2021. 1, 2

Figure 2. More comparisons with Unsup3D [6], LAP [7] and DECA [2]. Our method uses the low-resolution input, while other approaches leverage DFDNet [5] to pre-process the input.