Supplementary Material for Toward Realistic Single-View 3D Object Reconstruction with Unsupervised Learning from Multiple Images

Anonymous ICCV submission

Paper ID 9848

Abstract

In this supplementary document, we provide further details about the network architecture and the execution time. We also include additional qualitative results.

1. Architecture details

1.1. Networks for depth and albedo estimation

The networks for depth and albedo estimation $\mathcal{F}_d$ and $\mathcal{F}_a$ have the same architecture, which is presented in Table 1. The only difference between these networks is the number of outputs, which is denoted as *cout* in Table 1. $cout = 1$ for the depth estimator $\mathcal{F}_d$ and $cout = 3$ for the albedo estimator $\mathcal{F}_a$.

	Output size
Encoder	
Conv(3 , 64, 4, 2,1) + GN(16) + LReLU(0.2)	32
Conv(64, 128, 4, 2, 1) + GN(32) + LReLU(0.2)	16
Conv(128, 256, 4, 2, 1) + GN(64) + LReLU(0.2)	8
Conv(256, 512, 4, 2, 1) + LReLU(0.2)	4
Conv(512, 256, 4, 1, 0) + ReLU	1
Decoder	
Deconv(256, 512, 4, 1, 0) + ReLU	4
Conv(512, 512, 3, 1, 1) + ReLU	4
Deconv(512, 256, 4, 2, 1) + GN(64) + ReLU	8
Conv(256, 256, 3, 1, 1) + GN(64) + ReLU	8
Deconv(256, 128, 4, 2, 1) + GN(32) + ReLU	16
Conv(128, 128, 3, 1, 1) + GN(32) + ReLU	16
Deconv(128, 64, 4, 2, 1) + GN(16) + ReLU	32
Conv(64, 64, 3, 1, 1) + GN(16) + ReLU	32
Upsample(2)	64
Conv(64, 64, 3, 1, 1) + GN(16) + ReLU	64
Conv(64, 64, 5, 1, 2) + GN(16) + ReLU	64
Conv(64, **cout**, 5, 1, 2) + Tanh → **output**	64

Table 1: The network architecture for the depth and albedo estimators.

1.2. Network for confident map estimation

	Output size
Encoder	
Conv(**cin** , 64, 4, 2,1) + GN(16) + LReLU(0.2)	32
Conv(64, 128, 4, 2, 1) + GN(32) + LReLU(0.2)	16
Conv(128, 256, 4, 2, 1) + GN(64) + LReLU(0.2)	8
Conv(256, 512, 4, 2, 1) + LReLU(0.2)	4
Conv(512, 128, 4, 1, 0) + ReLU	1
Decoder	
Deconv(128, 512, 4, 1, 0) + ReLU	4
Deconv(512, 256, 4, 2, 1) + GN(64) + ReLU	8
Deconv(256, 128, 4, 2, 1) + GN(32) + ReLU	16
↳ Conv(128, 2, 3, 1, 1) + SoftPlus → **output**	16
Deconv(128, 64, 4, 2, 1) + GN(16) + ReLU	32
Deconv(64, 64, 4, 2, 1) + GN(16) + ReLU	64
↳ Conv(64, 2, 5, 1, 2) + SoftPlus → **output**	64

Table 2: The network architecture for the confidence map estimators.

We present the architecture for confidence networks in Table 2. Note that each network has two outputs for the l_1 and perceptual loss components. This time, the difference between these networks is the number of input channels cin. With the single-view confidence network $\mathcal{F}_c$, there is only one single RGB image fed through, so $cin = 3$. With the cross-view network $\mathcal{F}_{cc}$, however, we stack two different views of the object by channel and feed into the network, making $cin = 6$.

1.3. Networks for viewpoint and lighting estimation.

Finally, the architecture of the viewpoint and lighting estimation networks is shown in Table 3. Unlike the previous networks, which are auto-encoders, these networks are regression networks with different numbers of output channels $cout$. We use $cout = 6$ for the viewpoint estimator $\mathcal{F}_v$ and $cout = 4$ for the lighting estimator $\mathcal{F}_l$.

	Output size
Encoder	
Conv(3 , 32, 4, 2,1) + ReLU	32
Conv(32 , 64, 4, 2,1) + ReLU	16
Conv(64, 128, 4, 2, 1) + ReLU	8
Conv(128, 256, 4, 2, 1) + ReLU	4
Conv(256, 256, 4, 1, 0) + ReLU	1
Conv(256, **cout**, 1, 1, 0) + Tanh $\rightarrow$ **output**	1

Table 3: Network architecture for the viewpoint and lighting regressors.

2. Inference time

We implement LeMul using Pytorch 1.2.0 and test on single RTX 2080 Ti. Assuming a properly cropped input image, the inference times for each network module and the rendering function $\mathcal{R}$ using Phong shading model [3] are shown in Table 4. The total time to recover the 3D model (depth + albedo) from an image is only 3.95ms, making our 3D reconstruction speed more than 250fps.

	Module	Speed (ms)
(1)	Depth estimation	2.49
(2)	Albedo estimation	1.46
(3)	View estimation	0.41
(4)	Light estimation	1.47
(5)	Rendering ($\mathcal{R}$)	9.09
(1) + (2)	3D modeling	3.95

Table 4: Running time.

3. Extra qualitative figures

We provide additional qualitative examples on the tested datasets in Fig. 1. For each dataset, two extra examples are shown.

References

[1] R. Gross, I. Matthews, J. Cohn, T. Kanade, and S. Baker. Multi-pie. In *Automatic Face Gesture Recognition, 2008. FG '08. 8th IEEE International Conference on*, pages 1–8, Sept 2008. 3

[2] 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. 3

[3] Bui Tuong Phong. Illumination for computer generated pictures. *Communications of the ACM*, 18(6):311–317, 1975. 2

[4] Lior Wolf, Tal Hassner, and Itay Maoz. Face recognition in unconstrained videos with matched background similarity. In *CVPR 2011*, pages 529–534. IEEE, 2011. 3

[5] Shangzhe Wu, Christian Rupprecht, and Andrea Vedaldi. Unsupervised learning of probably symmetric deformable 3d objects from images in the wild. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1–10, 2020. 3

[6] Dong Yi, Zhen Lei, Shengcai Liao, and Stan Z Li. Learning face representation from scratch. *arXiv preprint arXiv:1411.7923*, 2014. Available: http://www.cbsr.ia.ac.cn/english/CASIA-WebFace-Database.html. 3

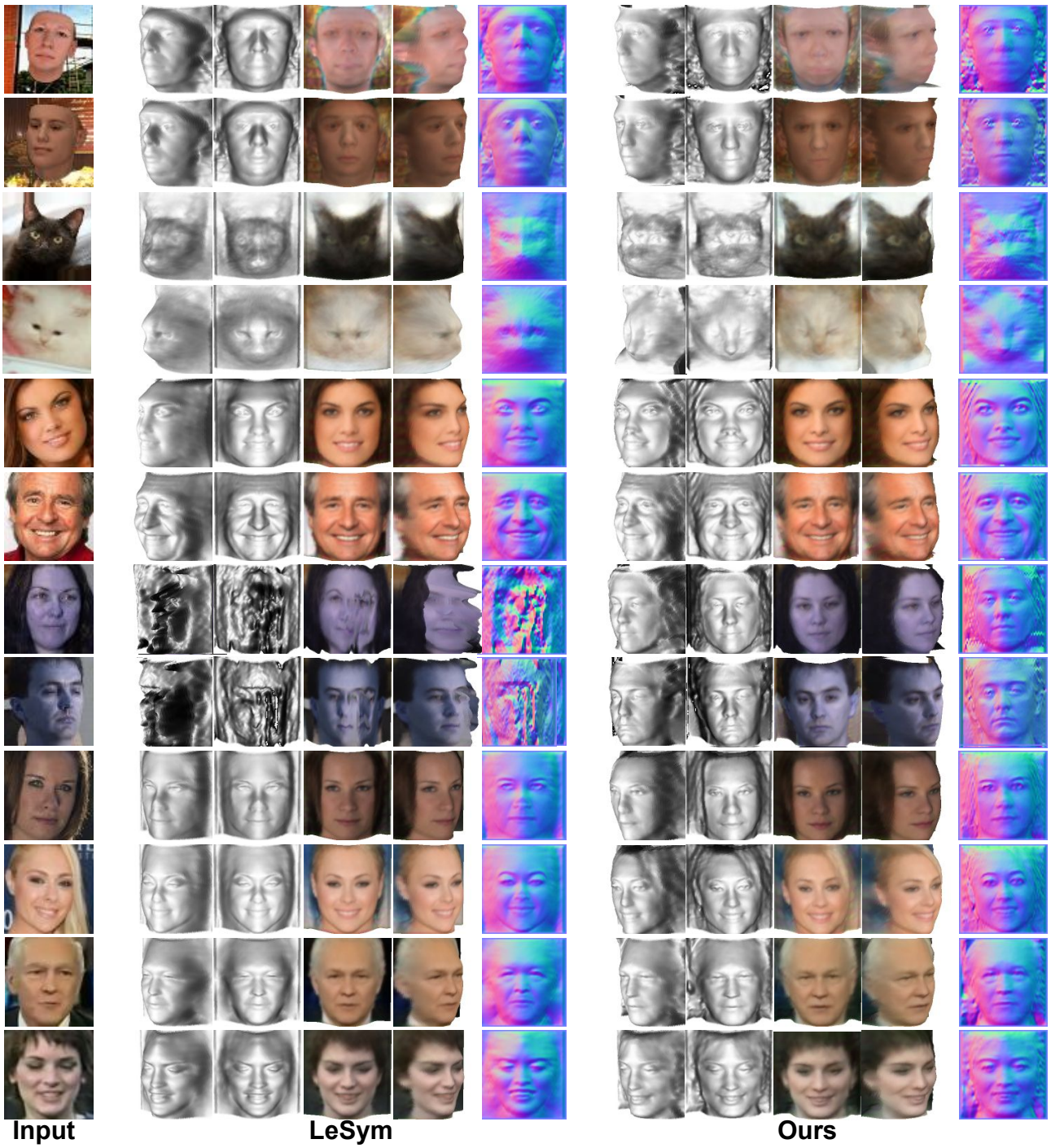

Figure 1: Comparing the reconstructed 3D models from the baseline method LeSym [5] and ours on six datasets.The datasets from top to bottom (2 samples each dataset): BFM [5], Cat Faces [5], CelebA [2], Multi-PIE [1] CASIA-WebFace [6] and Youtube Faces [4]. For each 3D model, we provide two textureless views, two textured views, and the canonical normal map