Supplementary Material: Unsupervised Learning of Probably Symmetric Deformable 3D Objects from Images in the Wild

Shangzhe Wu Christian Rupprecht Andrea Vedaldi

Visual Geometry Group, University of Oxford

{szwu, chrisr, vedaldi}@robots.ox.ac.uk

1. Technical Details

1.1. Differentiable rendering layer

As noted in the Section 3.3 in the main paper, the reprojection function Π *warps* the canonical image $\mathbf{J}$ to generate the actual image $\mathbf{I}$. In CNNs, image warping is usually regarded as a simple operation that can be implemented efficiently using a bilinear resampling layer [3]. However, this is true only if we can easily send pixels (u', v') in the warped image $\mathbf{I}$ back to pixels (u, v) in the source image $\mathbf{J}$, a process also known as *backward warping*. Unfortunately, in our case the function $\eta_{d,w}$ obtained by Eq. (6) in the paper sends pixels in the opposite way.

Implementing a *forward warping* layer is surprisingly delicate. One way of approaching the problem is to regard this task as a special case of rendering a textured mesh. The *Neural Mesh Renderer* (NMR) of [4] is a differentiable renderer of this type. In our case, the mesh has one vertex per pixel and each group of 2×2 adjacent pixels is tessellated by two triangles. Empirically, we found the quality of the texture gradients of NMR to be poor in this case, likely caused by high frequency content in the texture image $\mathbf{J}$.

We solve the problem as follows. First, we use NMR to warp only the depth map d, obtaining a version $\bar{d}$ of the depth map as seen from the input viewpoint. This has two advantages: backpropagation through NMR is faster and secondly, the gradients are more stable, probably also due to the comparatively smooth nature of the depth map d compared to the texture image $\mathbf{J}$. Given the depth map $\bar{d}$, we then use the inverse of Eq. (6) in the paper to find the warp field from the observed viewpoint to the canonical viewpoint, and bilinearly resample the canonical image $\mathbf{J}$ to obtain the reconstruction.

1.2. Training details

We report the training details including all hyperparameter settings in Table 1, and detailed network architectures in Tables 2 to 4. We use standard encoder networks for both viewpoint and lighting predictions, and encoder-decoder networks for depth, albedo and confidence predictions. In order to mitigate checkerboard artifacts [6] in the predicted depth and albedo, we add a convolution layer after each deconvolution layer and replace the last deconvolotion layer with nearest-neighbor upsampling, followed by 3 convolution layers. Abbreviations of the operators are defined as follows:

- Conv$(c_{in}, c_{out}, k, s, p)$: convolution with c_{in} input channels, c_{out} output channels, kernel size k, stride s and padding p.
- Deconv$(c_{in}, c_{out}, k, s, p)$: deconvolution [9] with c_{in} input channels, c_{out} output channels, kernel size k, stride s and padding p.
- Upsample(s): nearest-neighbor upsampling with a scale factor of s.
- GN(n): group normalization [8] with n groups.
- LReLU(α): leaky ReLU [5] with a negative slope of α.

2. Qualitative Results

We provide more qualitative results in the following and 3D animations in the supplementary video[1]. Fig. 1 reports the qualitative results of the ablated models in Table 3 of the main paper. Fig. 3 shows reconstruction results on human faces from CelebA and 3DFAW. We also show reconstruction results on face paintings and drawings collected from [2] and the Internet in Figs. 4 and 5. Figs. 6 to 8 show results on real cat faces from [10, 7], abstract cats collected from the Internet and synthetic cars rendered using ShapeNet.

Re-lighting. Since our model predicts the intrinsic components of an image, separating the albedo and illumination, we can easily re-light the objects with different lighting conditions. In Fig. 2, we demonstrate results of the intrinsic decomposition and the re-lit faces in the canonical view.

[1] https://www.youtube.com/watch?v=5rPJyrU-WE4

Parameter	Value/Range
Optimizer	Adam
Learning rate	1×10^{-4}
Number of epochs	30
Batch size	64
Loss weight λ_{f}	0.5
Loss weight λ_{p}	1
Input image size	64×64
Output image size	64×64
Depth map	$(0.9, 1.1)$
Albedo	$(0, 1)$
Light coefficient k_s	$(0, 1)$
Light coefficient k_d	$(0, 1)$
Light direction l_x, l_y	$(-1, 1)$
Viewpoint rotation $w_{1:3}$	$(-60°, 60°)$
Viewpoint translation $w_{4:6}$	$(-0.1, 0.1)$
Field of view (FOV)	10

Table 1: Training details and hyper-parameter settings.

Encoder	Output size
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, c_{out}, 1, 1, 0) + Tanh $\rightarrow$ *output*	1

Table 2: Network architecture for viewpoint and lighting. The output channel size c_{out} is 6 for viewpoint, corresponding to rotation angles $w_{1:3}$ and translations $w_{4:6}$ in x, y and z axes, and 4 for lighting, corresponding to k_s, k_d, l_x and l_y.

Testing on videos. To further assess our model, we apply the model trained on CelebA faces to VoxCeleb [1] videos *frame by frame* and include the results in the supplementary video. Our trained model works surprisingly well, producing consistent, smooth reconstructions across different frames and recovering the details of the facial motions accurately.

References

[1] Joon Son Chung, Arsha Nagrani, and Andrew Zisserman. VoxCeleb2: Deep speaker recognition. In *INTERSPEECH*, 2018. 2

[2] Elliot J. Crowley, Omkar M. Parkhi, and Andrew Zisserman. Face painting: querying art with photos. In *Proc. BMVC*, 2015. 1

[3] Max Jaderberg, Karen Simonyan, Andrew Zisserman, and Koray Kavukcuoglu. Spatial transformer networks. In *NeurIPS*, 2015. 1

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

[5] Andrew L. Maas, Awni Y. Hannun, and Andrew Y. Ng. Rectifier nonlinearities improve neural network acoustic models. In *Proc. ICML*, 2013. 1

Encoder	Output size
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	**Output size**
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, c_{out}, 5, 1, 2) + Tanh $\rightarrow$ *output*	64

Table 3: Network architecture for depth and albedo. The output channel size c_{out} is 1 for depth and 3 for albedo.

Encoder	Output size
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, 128, 4, 1, 0) + ReLU	1
Decoder	**Output size**
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 $\rightarrow$ *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 $\rightarrow$ *output*	64

Table 4: Network architecture for confidence maps. The network outputs two pairs of confidence maps at different spatial resolutions for photometric and perceptual losses.

[6] Augustus Odena, Vincent Dumoulin, and Chris Olah. Deconvolution and checkerboard artifacts. *Distill*, 2016. 1

[7] Omkar M. Parkhi, Andrea Vedaldi, Andrew Zisserman, and C. V. Jawahar. Cats and dogs. In *Proc. CVPR*, 2012. 1

[8] Yuxin Wu and Kaiming He. Group normalization. In *Proc. ECCV*, 2018. 1

[9] Matthew D. Zeiler, Graham W. Taylor, and Rob Fergus. Adaptive deconvolutional networks for mid and high level feature learning. In *Proc. ICCV*, 2011. 1

[10] Weiwei Zhang, Jian Sun, and Xiaoou Tang. Cat head detection - how to effectively exploit shape and texture features. In *Proc. ECCV*, 2008. 1

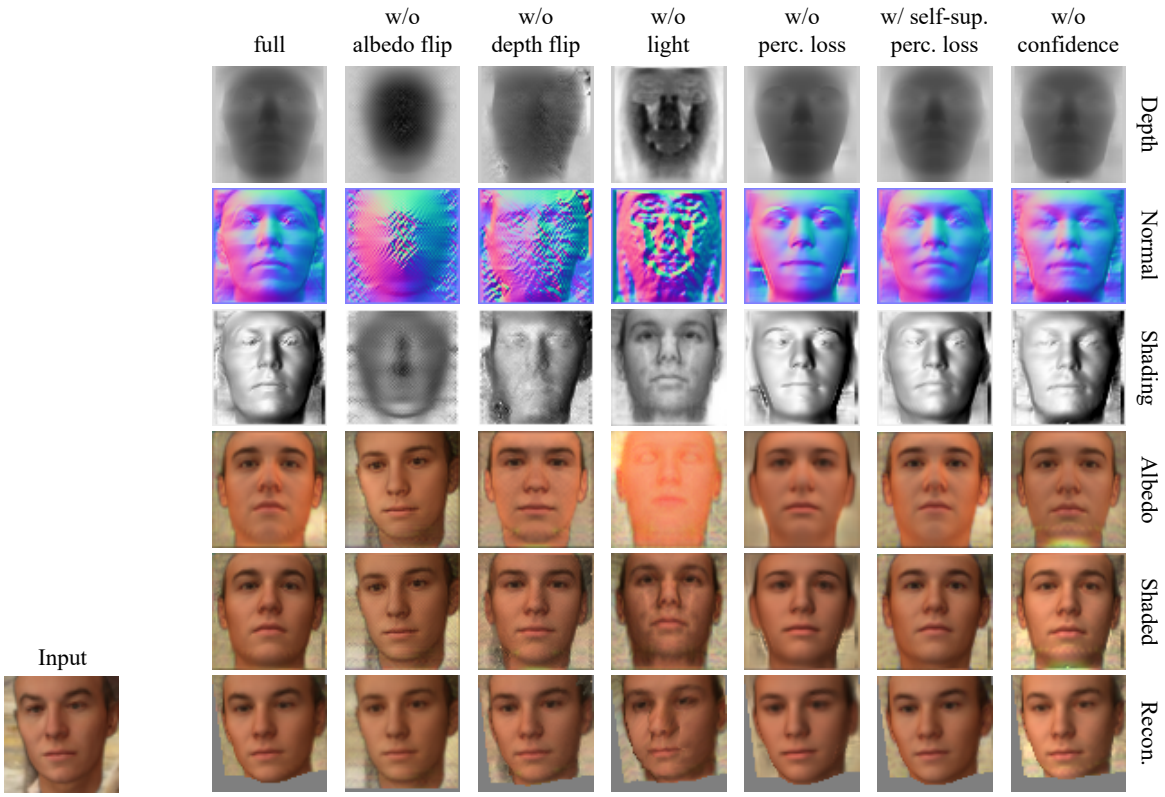


Figure 1: **Qualitative results of the ablated models.**

Figure 2: **Re-lighting effects.**

Figure 3: **Reconstruction of human faces.**

Figure 4: **Reconstruction of face paintings.**

Figure 5: **Reconstruction of abstract faces.**

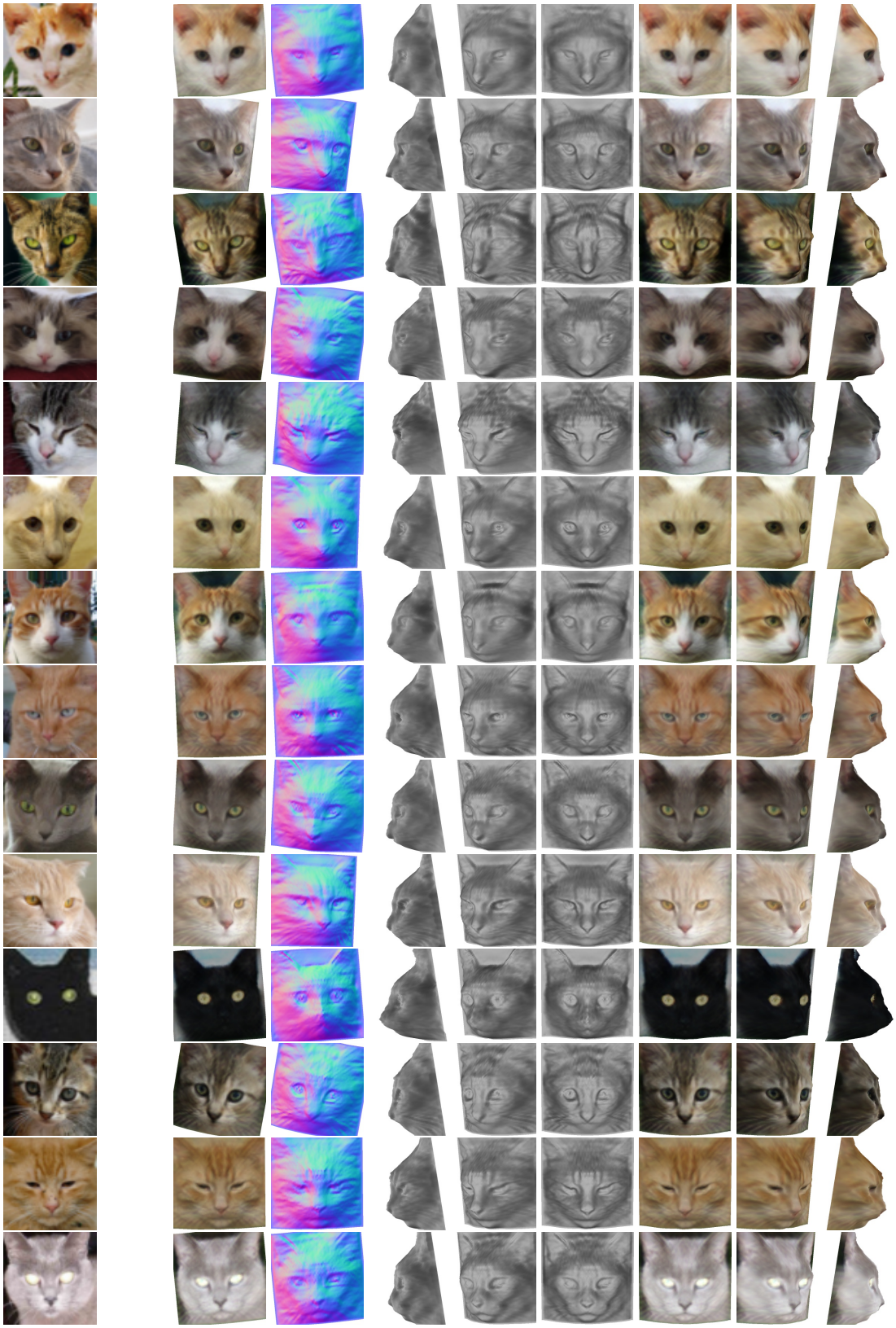
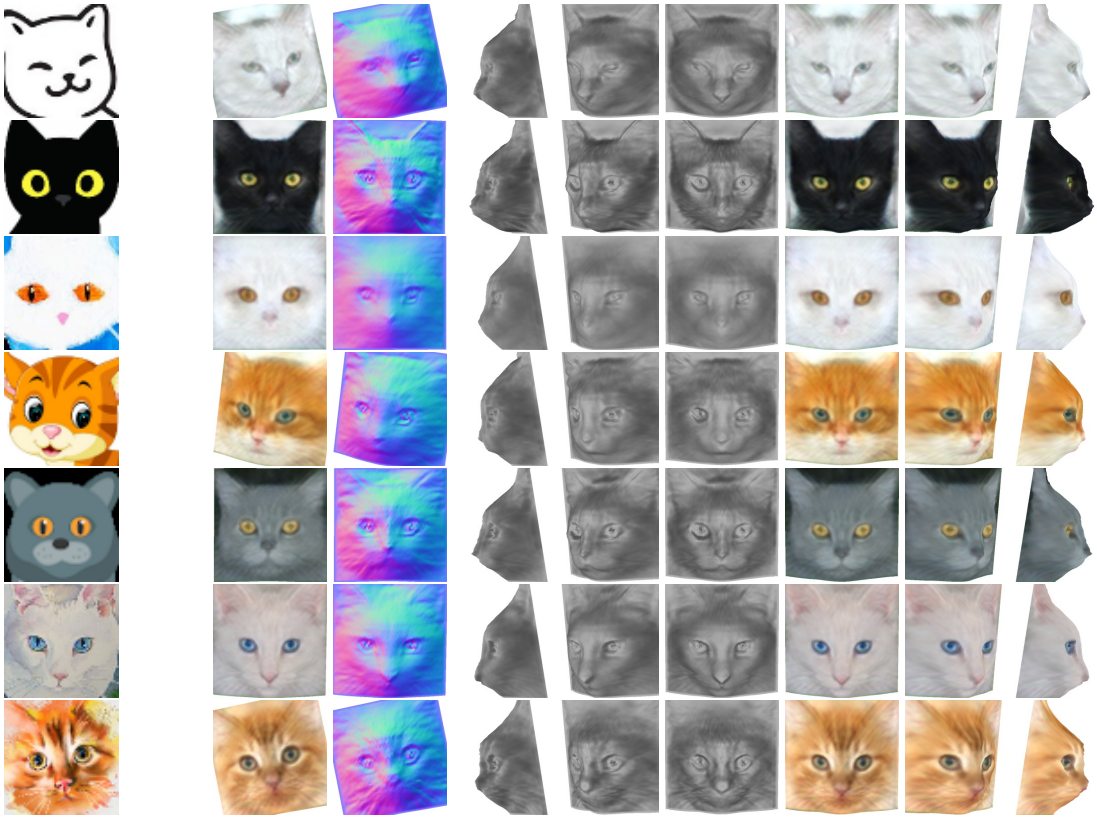
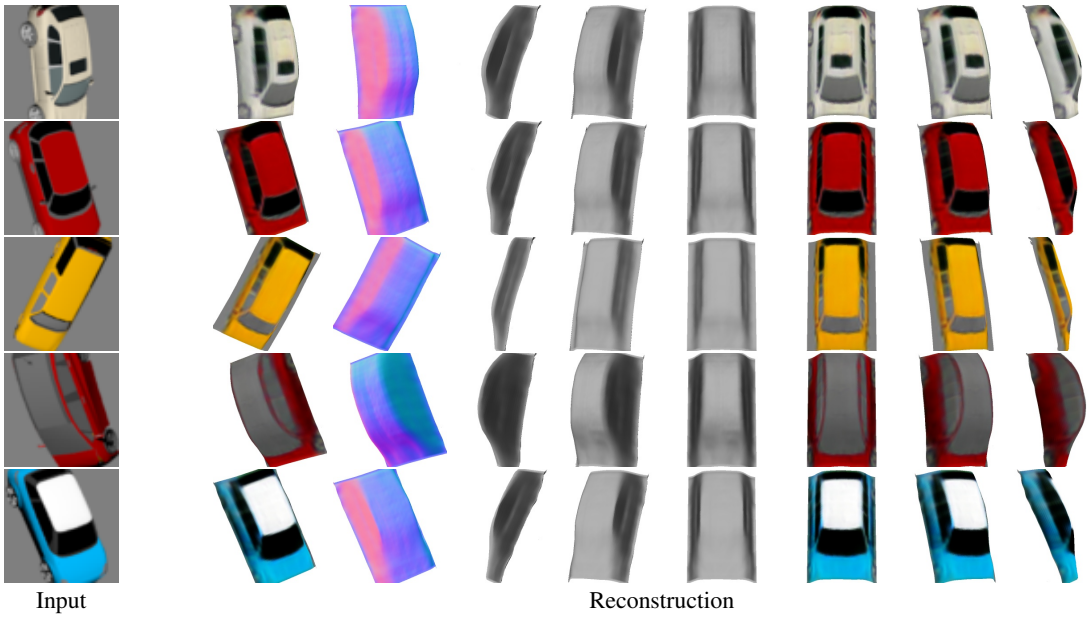


Figure 6: **Reconstruction of cat faces.**



Input Reconstruction

Figure 7: **Reconstruction of abstract cats.**



Input Reconstruction

Figure 8: **Reconstruction of synthetic cars.**