GAN2X: Non-Lambertian Inverse Rendering of Image GANs

Supplementary Material

Xingang Pan[1] Ayush Tewari[1,2] Lingjie Liu[1] Christian Theobalt[1]

[1]Max Planck Institute for Informatics [2]MIT

{xpan,lliu,theobalt}@mpi-inf.mpg.de ayusht@mit.edu

In this supplementary material, we provide the implementation details, discuss the limitations, and show more qualitative results. We also recommend readers to refer to the video demos at the project page.

1. Implementation Details

Model Architectures. Similar to the design of NeuS [7], our shape MLP F_s has 8 hidden layers while the material MLP F_m has 4 hidden layers. The number of channels for each hidden layer is 256. Apart from the coordinate $\boldsymbol{x}$, F_m also takes the surface normal $\boldsymbol{n}$ and the last feature from F_s as input. Positional encoding [3] is applied to $\boldsymbol{x}$ with 4 frequencies for F_s and 6 frequencies for F_m. In volume rendering, the number of coarse and fine samples are 36 and 36 respectively.

The architecture for viewpoint encoder E_v and lighting encoder E_l is described in Tab. 1. The architecture for image encoder E_I and GAN encoder E_w is described in Tab. 2. These architectures are based on 256^2-resolution input images. In our experiments, we use 256^2 resolution for most datasets except CelebA [2], for which 128^2 resolution is used. For 128^2-resolution input images, the second convolution layer in Tab. 1 and the first ResBlock in Tab. 2 are removed while the output channel of the first convolution layer is increased from 16 to 32. The abbreviations for the network layers are described below:

- 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.
- Avg_pool(s): average pooling with a stride of s.
- ResBlock(c_{in}, c_{out}): residual block as defined in Tab.3.

Encoder	Output size
Conv(3, 16, 4, 2, 1) + ReLU	128
Conv(16, 32, 4, 2, 1) + ReLU	64
Conv(32, 64, 4, 2, 1) + ReLU	32
Conv(64, 128, 4, 2, 1) + ReLU	16
Conv(128, 256, 4, 2, 1) + ReLU	8
Conv(256, 512, 4, 2, 1) + ReLU	4
Conv(512, 512, 4, 1, 0) + ReLU	1
Conv(512, c_{out}, 1, 1, 0) + Tanh	1

Table 1. Network architecture for viewpoint net E_v and lighting net E_l. The output channel size c_{out} is 6 for E_v and 4 or 5 for E_l depending on whether the negative shading term is used.

Encoder	Output size
Conv(3, 16, 4, 2, 1) + ReLU	128
ResBlock(16, 32)	64
ResBlock(32, 64)	32
ResBlock(64, 128)	16
ResBlock(128, 256)	8
ResBlock(256, 512)	4
Conv(512, 1024, 4, 1, 0) + ReLU	1
Conv(1024, 512, 1, 1, 0)	1

Table 2. Network architecture of image encoder E_I and GAN encoder E_w.

Residual path
ReLU + Conv(c_{in}, c_{out}, 3, 2, 1)
ReLU + Conv(c_{out}, c_{out}, 3, 1, 1)
Identity path
Avg_pool(2)
Conv(c_{in}, c_{out}, 1, 1, 0)

Table 3. Network architecture for the ResBlock(c_{in}, c_{out}) in Tab.2. The output of Residual path and Identity path are added as the final output.

Hyperparameters. The hyperparameters used in our experiments are provided in Tab. 4, Tab. 5, and Tab. 6. For clarity, we denote the material network optimization process in **Exploration** as "step 1", the GAN reconstruction process that trains E_w as "step 2", and the **Exploitation** process as "step 3". "Number of stages" denotes how many times the exploration-and-exploitation process are repeated. For the chromaticity-based smoothness loss, we use differ-

Parameter	Value/Range
p	(4, 100)
β	$3e^{-5}$
λ_{vl}	1.0
λ_d	1.0
λ_m	0.2
lr for F_s and F_m	$5e^{-4}$
lr for F_m in step1	$2e^{-3}$
lr for all encoders	$2e^{-4}$

Table 4. Hyper-parameters. lr denotes learning rate.

Joint Pre-training	Value
Number of samples	(1k/200/100)
Number of re-rendered samples m	(32/80/160)
Number of stages	5
Step 1 iterations (1st stage)	50k
Step 1 iterations (other stages)	30k
Step 2 iterations	8k
Step 3 iterations	50k
$(\sigma_{min}, \sigma_{max})$	(0.3, 1.0)
$(\lambda_{reg1}, \lambda_{reg2})$	(0.2, 0.1)
Instance-specific fine-tuning	**Value**
Number of re-rendered samples m	(800/400/400)
Number of stages	3
Step 1 iterations	6k
Step 2 iterations	2k
Step 3 iterations	15k
$(\sigma_{min}, \sigma_{max})$	(0.8, 1.0)
$(\lambda_{reg1}, \lambda_{reg2})$	(0.2, 0.1)
Shading-based refinement	**Value**
Iterations	1k
$(\sigma_{min}, \sigma_{max})$	(1.0, 1.0)
$(\lambda_{reg1}, \lambda_{reg2})$	(0.5, 0.2)

Table 5. Hyper-parameters for CelebA, CelebA-HQ, and AFHQ Cat datasets. (x/y/z) denote values for CelebA, CelebA-HQ, and AFHQ Cat respectively.

ent λ_{reg} values for albedo and specularity maps, which are denoted as λ_{reg1} and λ_{reg2} respectively.

Training Process. For CelebA, CelebA-HQ, and AFHQ Cat datasets, we first pre-train our model on multiple samples jointly as mentioned in Sec. 3.2 of the main paper. We then perform instance-specific training for any individual sample. For LSUN car, we do not perform joint training, but first train on 128^2 resolution and then fine-tune on 256^2 resolution for each instance. For all datasets, shading-based refinement is finally applied to further refine the results. Note that for the application of lifting 2D GAN to 3D GAN, only joint pre-training is involved as there is no need for instance-specific training. And for application on real images, only joint pre-training and shading-based refinement are involved. This is because instance-specific fine-tuning for real images would require GAN inversion, which harms editability and thus does not perform very stable in practice.

Losses. Similar to [7, 9], we use an Eikonal term to regularize the SDF of F_s by $\mathcal{L}_{eik} = \frac{1}{n}\sum_i (|\nabla F_s(\boldsymbol{x}_i)| - 1)^2$, where $\boldsymbol{x}_i$ are the sampled points and the loss weight for this reg-

Pre-train on 128 resolution	Value
Number of re-rendered samples m	800
Number of stages	4
Step 1 iterations (1st stage)	15k
Step 1 iterations (other stages)	10k
Step 2 iterations	4k
Step 3 iterations	25k
$(\sigma_{min}, \sigma_{max})$	(0.3, 1.0)
$(\lambda_{reg1}, \lambda_{reg2})$	(0.2, 0.1)
Fine-tune on 256 resolution	**Value**
Number of re-rendered samples m	400
Number of stages	3
Step 1 iterations	6k
Step 2 iterations	2k
Step 3 iterations	15k
$(\sigma_{min}, \sigma_{max})$	(0.8, 1.0)
$(\lambda_{reg1}, \lambda_{reg2})$	(0.2, 0.1)
Shading-based refinement	**Value**
Iterations	1k
$(\sigma_{min}, \sigma_{max})$	(1.0, 1.0)
$(\lambda_{reg1}, \lambda_{reg2})$	(0.5, 0.2)

Table 6. Hyper-parameters for LSUN car dataset.

ularization term is 0.1. For CelebA-HQ and LSUN car, we also include a mask loss in the same way as [7], where the masks are obtained from off-the-shelf segmentation models.

The training process of Eq. 7 in the main paper is typically done by randomly sampling 512 pixels on the image and applying the losses with respect to these pixels. However, the perceptual loss [10] cannot by applied to these scattered pixels and rendering the whole image is infeasible due to heavy memory consumption. To this end, for each training iteration, we randomly choose from two pixel sampling strategies, where the first is random sampling as mentioned before and the second is to sample a 32^2 image patch. The first pixel sampling strategy preserves the randomness of sampled pixel positions while the second strategy allows perceptual loss to be applied to the image patch. The perceptual loss has a loss weight of 0.1.

Other Training Details. Our implementation is based on PyTorch [6]. We use Adam optimizer [1] in all experiments. The joint pre-training is run on 2 RTX 8000 GPUs, while all other instance-specific trainings are run on 1 RTX 8000 GPU.

For CelebA, CelebA-HQ, and AFHQ Cat datasets, we adopt a symmetry assumption on object shape and material at step3 of the first stage in joint pre-training. This is done by randomly flipping the shape and material during training, which is similar to [8]. This symmetry assumption helps to infer a canonical face pose.

Note that the exploration process of our method involves randomly sampling multiple viewpoints and lighting conditions. Here we follow [4], where the viewpoints are sampled from a prior multi-variate normal distribution and the lighting conditions are sampled from a prior uniform distribution.

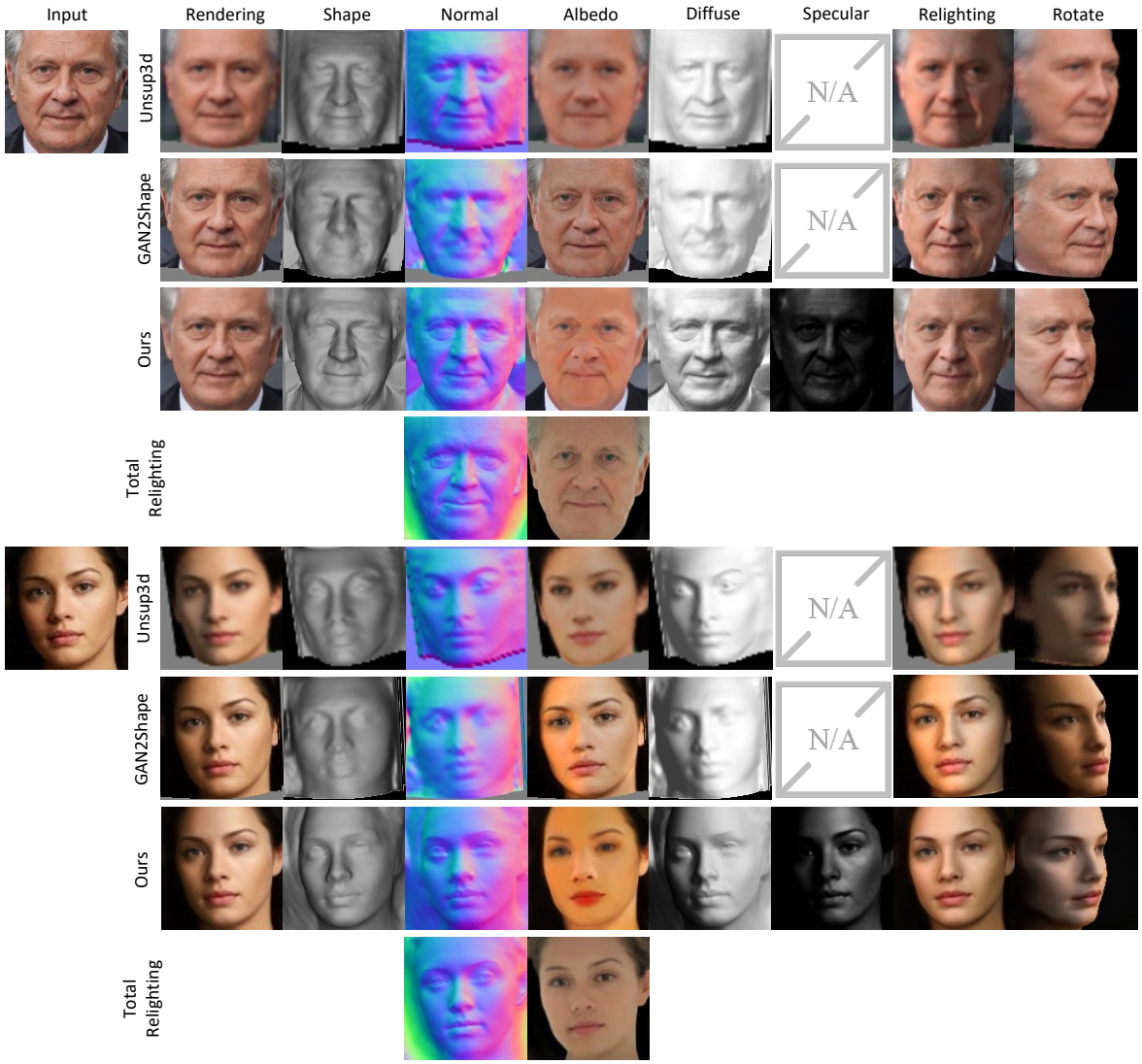

Figure 1. More qualitative comparison. This is an extension of Fig. 4 in the main paper.

2. Limitations

While our approach shows promising inverse rendering results, it also has some limitations. First, we assume simplified lighting to reduce ambiguity and to make the problem tractable, which may not be sufficient to well approximate more complex lighting. Besides, as the exploration step in our approach relies on a convex shape prior, it mainly works for roughly convex objects and is hard to be applied to more complex objects (*e.g.*, bicycles). This might be alleviated via the recent advancement in 3D-aware GANs that allow explicit camera pose control.

3. Qualitative Results

In this section, we provide more qualitative results of our method. Fig. 1 provides more qualitative comparison with Unsup3d [8] and GAN2Shape [4]. We also show the albedo and normal of the supervised method Total Relighting [5] as a reference. More qualitative results are shown in Fig. 2.

Note that our method repeats the exploration-and-exploitation process for several stages. We show the effects of this progressive training in Fig. 3. It can be seen that the results get more accurate with more training stages. The shading-based refinement further refines the results to be more precise. In Fig. 4, we provide some examples of re-rendered images and projected images during training. It can be observed that the projected images have similar viewpoints and lighting conditions as the re-rendered images, but are more natural and thus provide useful information to refine the object intrinsics.

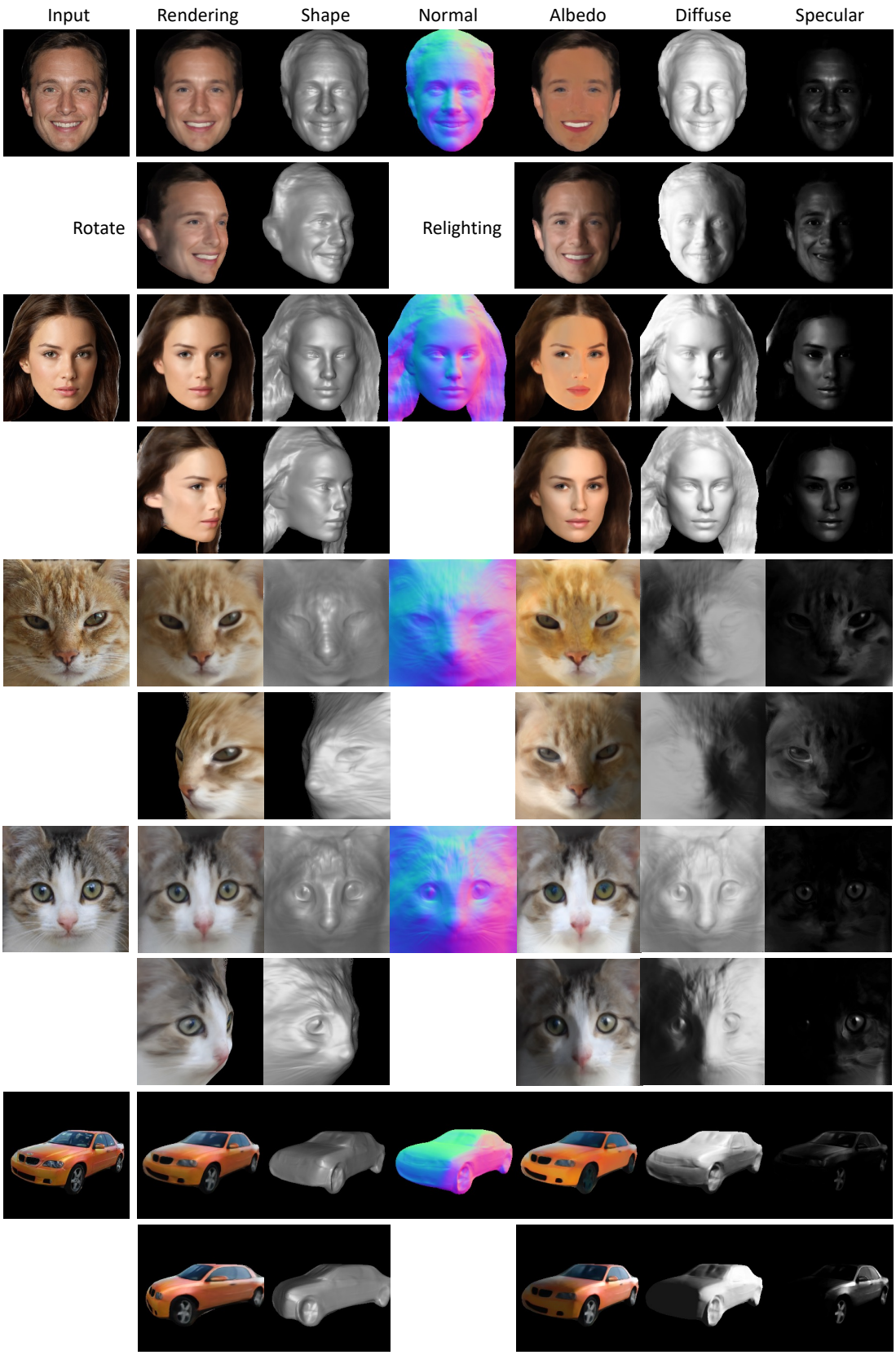

Figure 2. More qualitative results. This is an extension of Fig. 5 in the main paper.

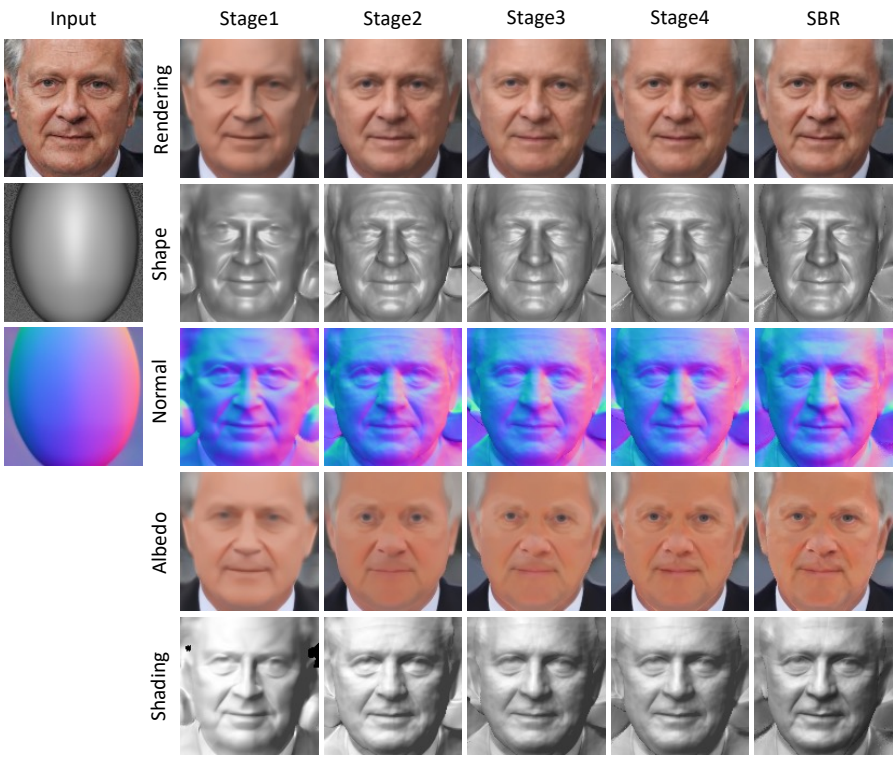

Figure 3. Effects of progressive training and shading-based refinement (SBR).

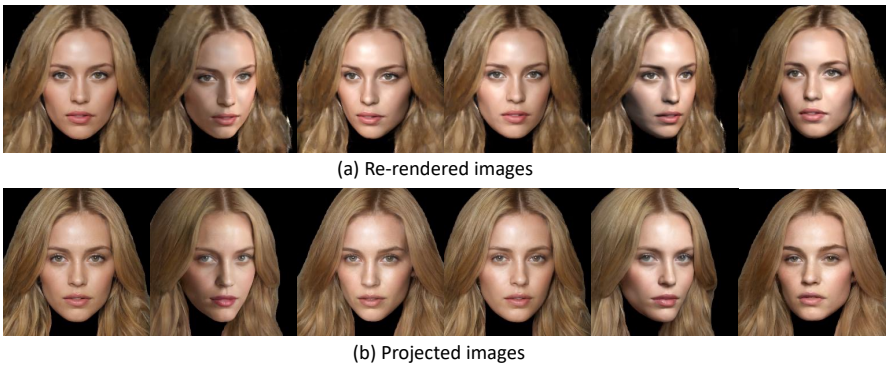

Figure 4. Examples of (a) re-rendered images and (b) their corresponding projected images.

References

[1] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 2

[2] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *ICCV*, 2015. 1

[3] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020. 1

[4] Xingang Pan, Bo Dai, Ziwei Liu, Chen Change Loy, and Ping Luo. Do 2d gans know 3d shape? unsupervised 3d shape reconstruction from 2d image gans. In *ICLR*, 2021. 2, 3

[5] Rohit Pandey, Sergio Orts Escolano, Chloe Legendre, Christian Haene, Sofien Bouaziz, Christoph Rhemann, Paul Debevec, and Sean Fanello. Total relighting: learning to relight portraits for background replacement. *ACM Transactions on Graphics (TOG)*, 40(4):1–21, 2021. 3

[6] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017. 2

[7] Peng Wang, Lingjie Liu, Yuan Liu, Christian Theobalt, Taku

Komura, and Wenping Wang. Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. In *NeurIPS*, 2021. 1, 2

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

[9] Lior Yariv, Yoni Kasten, Dror Moran, Meirav Galun, Matan Atzmon, Basri Ronen, and Yaron Lipman. Multiview neural surface reconstruction by disentangling geometry and appearance. In *NeurIPS*, 2020. 2

[10] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018. 2