Supplementary Material for Learning Implicit Surface Light Fields

Michael Oechsle[1,2] Michael Niemeyer[1] Christian Reiser[1]
Lars Mescheder[1,3] Thilo Strauss[2†] Andreas Geiger[1]
[1]Max Planck Institute for Intelligent Systems and University of Tübingen
[2]ETAS GmbH, Bosch Group, Stuttgart [3]Amazon, Tübingen
[1]{firstname.lastname}@tue.mpg.de [†]{firstname.lastname}@etas.com

Abstract

In this supplementary material, we first provide implementation details regarding our network architectures and training procedure in Section 1. Next, we explain our rendering setup in greater detail in Section 2. In Section 3, we discuss how our method can be used in combination with environment maps. Additional experimental results can be found in Section 6.

1. Implementation Details

We first describe our network architectures in Section 1.1 and provide details regarding the training procedure in Section 1.2.

1.1. Network Architectures

Our Model: In Fig. 1, we show the network architectures of our 2-Step model. The first network predicts a 32-dimensional appearance feature vector $\mathbf{f}$ for a given 3D location $\mathbf{p}$. We map the 3D location $\mathbf{p}$ to the positional encoding γ with $k = 10$, following [9].

$$\gamma(\mathbf{x}) = \left[\sin(2^0\pi\mathbf{x}), \cos(2^0\pi\mathbf{x}) ... \sin(2^k\pi\mathbf{x}), \cos(2^k\pi\mathbf{x})\right] \tag{1}$$

In the single view experiment, we condition the network on the geometry encoding $\mathbf{s}$ and input image encoding $\mathbf{z}$ as discussed in the main paper. The second network maps the appearance feature vector $\mathbf{f}$, the view direction $\mathbf{v}$, the light configuration $\mathbf{l}$ (e.g. the light location), and the 3D geometry encoding $\mathbf{s}$ to an RGB value. We encode both spatial inputs ($\mathbf{v}$ and $\mathbf{l}$) to the PE with $f = 4$.

The networks consist of 6 (first network) and 5 (second network) fully-connected residual blocks with a hidden dimension of 128. For the single view experiment, we concatenate all input encodings to form a single vector which we pass through a fully-connected layer and add to the output of every block.

Geometry Encoder: For encoding the geometry of the input object, we randomly sample 2048 points on the surface of the object. We use a PointNet architecture [11] with residual blocks and intermediate pooling [8] to obtain a global feature vector $\mathbf{s}$. The network architecture is shown in Fig. 2.

Image Encoder: We used a Resnet18 [4] network pre-trained on ImageNet [7] for encoding the input image into a global feature encoding $\mathbf{z}$. Our input images have a resolution of 256^2 pixels and the image encoding dimension is 512. A detailed visualization is provided in Fig. 3.

VAE Encoder: We implement our generative model as a variational autoencoder [6]. As encoder, we use a Resnet18-based architecture [4] conditioned on the shape encoding $\mathbf{s}$. The network maps an image $\mathbf{I}$ as well as the corresponding shape encoding $\mathbf{s}$ to a mean and a variance vector of a Gaussian distribution (see Fig. 4). We sample the latent code $\mathbf{z}$ from this Gaussian distribution.

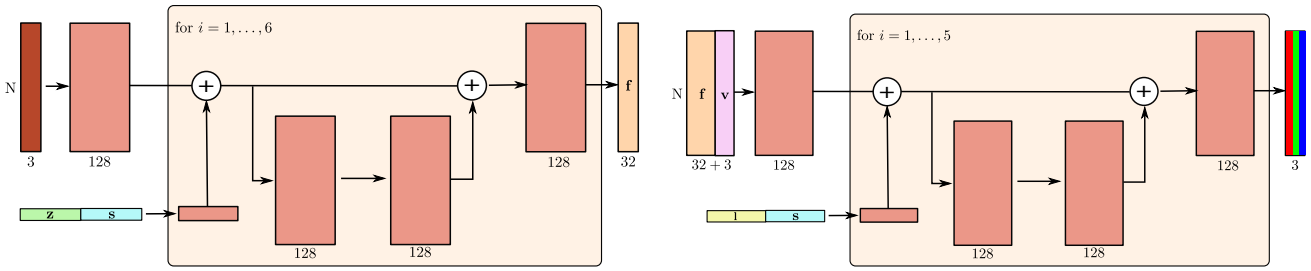

Figure 1. **2-Step Model.** Network architectures of our 2-step model that consists of two separate networks. We pass the 3D locations through the first network to determine the appearance feature vector $\mathbf{f}$. For the single view experiment, we condition this network on the input image ($\mathbf{z}$) and the object shape ($\mathbf{s}$). The second network maps $\mathbf{f}$, the view direction $\mathbf{v}$, the light configuration $\mathbf{l}$ and the shape encoding $\mathbf{s}$ to an RGB value.

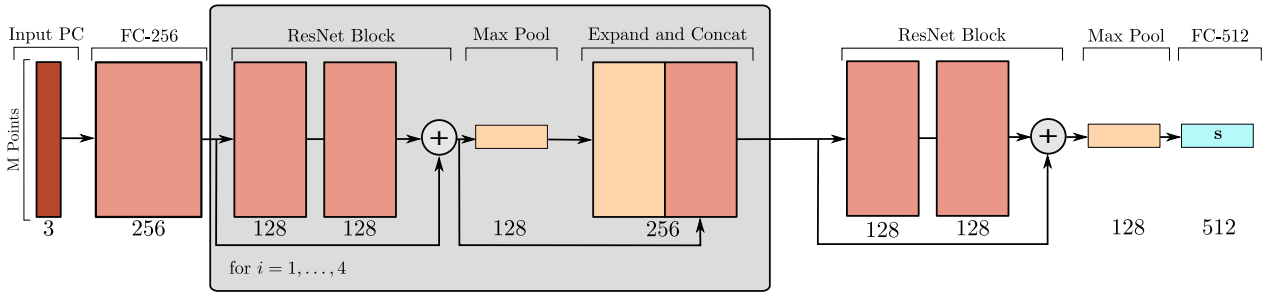

Figure 2. **Geometry Encoder.** We show the architecture of the geometry encoder. The geometry encoder maps a randomly sampled pointcloud of the input object to a global feature vector $\mathbf{s}$.

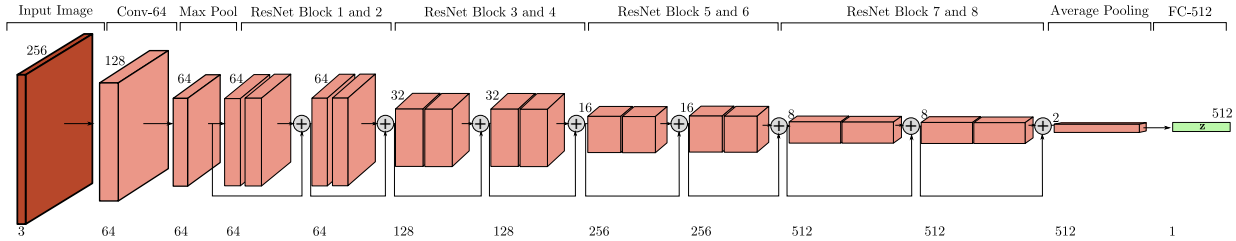

Figure 3. **Image Encoder.** We use a pre-trained Resnet18 for encoding the input image into a global feature vector $\mathbf{z}$.

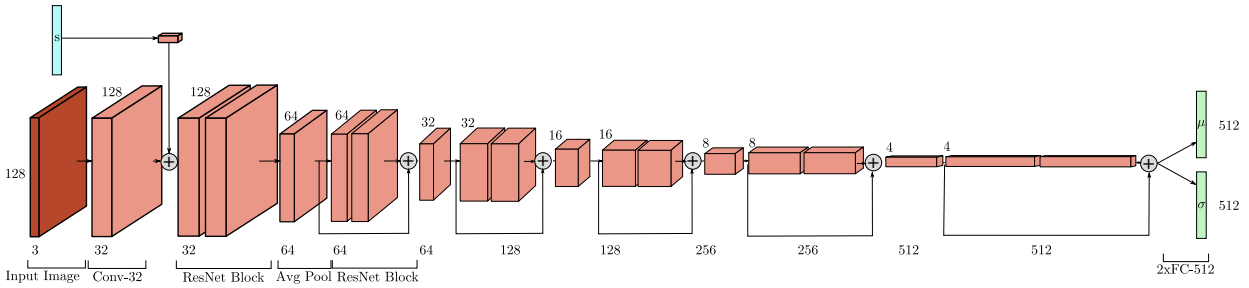

Figure 4. **VAE Encoder.** Our VAE encoder maps an image and a geometry encoding to the mean and variance of the Gaussian distributed latent code $\mathbf{z}$.

Image-to-Image Translation Baseline: In the single view experiment, we use an image-to-image translation approach with a U-Net architecture [12] as baseline. The network maps a depth map to an RGB image and is conditioned on the geometry encoding $\mathbf{s}$, light configuration $\mathbf{l}$, camera position $\mathbf{c}$ and the image encoding $\mathbf{z}$ (see Fig. 5). The network is trained in a supervised fashion with paired examples of depth maps and RGB images. During training, we use the same loss function as for our model which is discussed in the main paper.

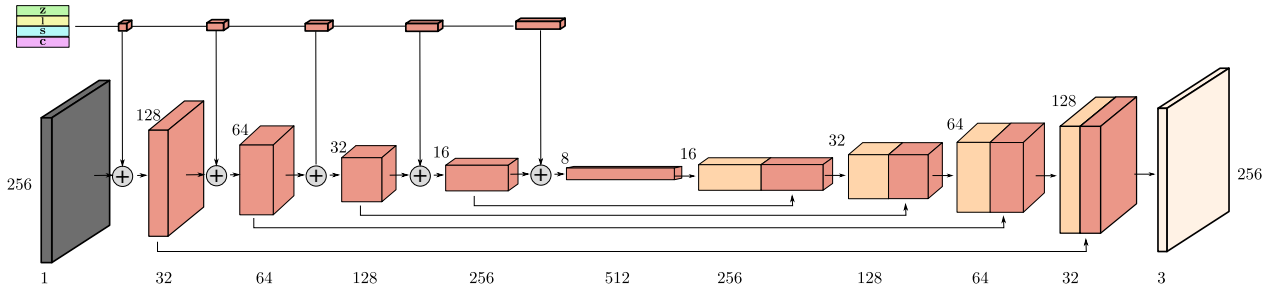

Figure 5. **Image-to-Image Translation.** We show the architecture of the Image-to-Image Translation baseline. We pass a depth image through a U-net architecture to directly infer an RGB image.

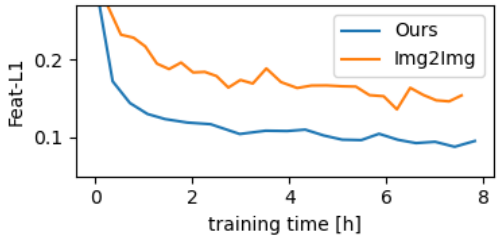

Figure 6. **Training times.** We show the performance of our method on the single objects task over training time.

1.2. Training Details

For simplicity, we explain the training procedure of a single training iteration considering one element of the batch. Depending on the experiment, we first encode the provided conditions into encodings $\mathbf{s}$, $\mathbf{z}$, or $\mathbf{l}$. Second, we sample a set of 500 (single view experiment) and 2048 (single object experiment on chair) pixels of the GT depth map that are inside the object region and map these pixels to their 3D locations using camera intrinsics. For the single object experiment with the car and small pointlights, a lot of ground truth pixel inside the object region are black, so that the training can collapse to only produce black outputs. We emprically found that we can circumvent the collapse by sampling brighter pixels more often. We then set the view directions $\mathbf{v}$ to the normalized vector from the 3D locations to the camera center. Next, we use the 3D locations, view directions and the encodings to obtain the corresponding RGB values. We then determine the loss between the predicted and the ground truth RGB values. In all experiments, we use the ADAM optimizer [5] with a learning rate of 10^{-4}. For the single view experiment and the generative model, we train our models with a batch size of 16.

For the single object task (Sec. 4.1) we train our method for 2 days on a 1080TI. Fig. 6 shows the performance against training time of our method and the Img2Img baseline. Our model for single view appearance prediction trains up to 7 days. Inferring an image of resolution 512^2 requires approx. 200ms.

2. Rendering Setup

In this section, we describe the rendering setups in greater detail. We used the Blender Cycles render engine [2] for rendering 3D objects from the Photoshape dataset [10] as discussed in the main paper. The 3D objects are in canonical pose and scaled to the unit cube. We randomly sample camera locations on the northern hemisphere with radius 1.5 (blender units). Similarly, we sample light locations on the northern hemisphere with radius 10. The number of light locations per view differs across the experiments and is hence given in the respective section. For the single view experiment, we additionally use a uniform background light of strength 0.2 to guarantee that the input images are not completely black. For the reflection analysis experiment, we sample the color of the light source uniformly in RGB space. For all other experiments, we use a white light source.

3. Spatially Varying Illumination

To determine the surface light field wrt. environment maps, we approximate the illumination with a set of small light sources. We determine equidistributed points on a sphere using the method from [3]. We query the color at each of these

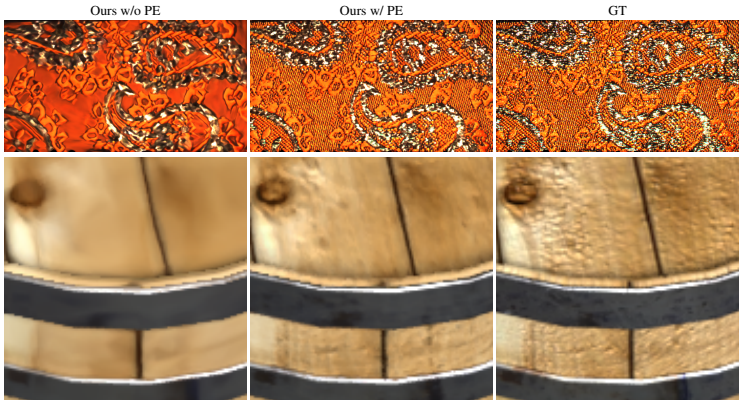

Figure 7. **Positional Encoding in the Single Object Experiments** The positional encoding significantly improves performance of our method on representing high-frequency details.

	FID		SSIM		Feature-ℓ_1	
	Ours	Ours w/o PE	Ours	Ours w/o PE	Ours	Ours w/o PE
photoshape	**20.602**	21.208	0.961	**0.963**	**0.131**	**0.131**
cars	39.170	**37.506**	0.935	**0.938**	0.187	**0.183**
lamps	**42.464**	44.604	**0.969**	0.968	**0.150**	0.153
sofas	32.172	**31.242**	**0.961**	0.957	**0.136**	**0.136**
mean	**33.602**	33.640	**0.957**	0.956	**0.151**	**0.151**

Table 1. **Positional Encoding in the Single View Experiment** The method with positional encoding compares similar to the ablation for the single view experiment in our setup.

points from the corresponding pixels in the environment map and consider each point as a single point light source. For each light source, we predict a separate RGB image and combine the resulting images as described in the main paper.

As our rendering procedure of the GT images contains a transformation from a radiometrically linear color space to the sRGB space[1], the predictions of our model are also in the non-linear sRGB space. For composing predicted images, it is thus necessary to transform them into a linear color space [1] before adding the RGB values. In the linear space, we add the images channel- and pixel-wise and then multiply the sum with a constant value reducing overexposure. The constant value is chosen to have a mean intensity of 0.35 in the image. Intuitively, this corresponds to varying the shutter time of a real camera. Finally, we transform the result back to the sRGB space.

4. Role of Positional Encoding

Recent works on implicit representations suggest using a positional fourier encoding in order to cirumvent bias of fully connected neural networks [9]. This enables network representing high-frequency details without extensive training times which is especially useful for representing textures. We can confirm the effectivenss of the PE in our single object experiment, see Fig. 7.

However it is unclear whether it also improves performance for tasks that require the model to generalize across additional input modalities as the image and shape conditions. Hence we compared both in our single view experiment. we found that for our experiment the positional encoding doesn't improve the performance of our method, see table Table 1.

5. Comparsion to the 1 step version

We compare our model to a network architecture consisting of a single step for predicting RGB color value from the same inputs as our original 2 step model, illustrated in Fig. 8. The comparison in Table 2 shows that the proposed method (2 step) achieves slightly better performance than the 1 step version on the Photoshapes dataset. More importantly, inferring views

[1] https://docs.blender.org/manual/en/latest/render/color_management.html

with novel light settings is more efficient with the 2 step model, so that it significantly reduces render times with environment maps, see in Table 2.

	FID	SSIM	Feat-ℓ_1	Render Time $n = 1000$
2 Step	**20.602**	**0.961**	**0.131**	**51.68s**
1 Step	20.900	**0.961**	0.132	85.13s

Table 2. **Comparison of the 1-step and 2-step model.** We compare both methods for the single view appearance prediction task on the Photoshapes dataset. For the comparison of the render time, $n = 1000$ point light sources are used to model the environment map.

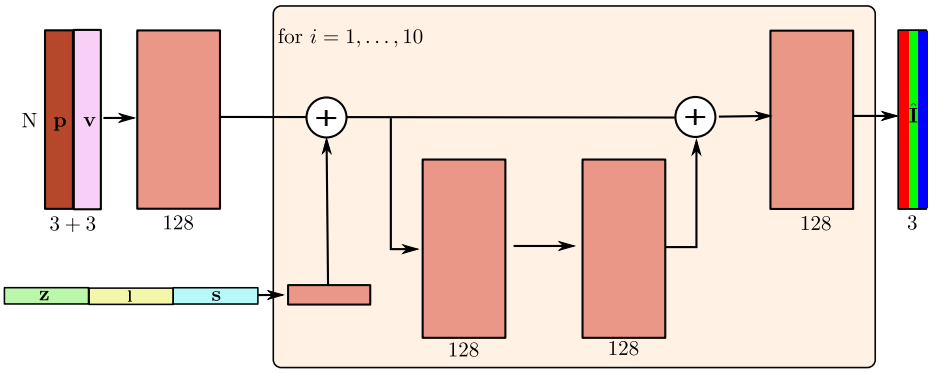

Figure 8. **1-Step Model.** Network architecture of the 1-step model. The network maps a 3D location $\mathbf{p}$ and a view direction $\mathbf{v}$ to an RGB value. We condition the network on the image encoding $\mathbf{z}$, the light configuration $\mathbf{l}$ and the shape encoding $\mathbf{s}$.

6. Additional Results

In this section, we provide more results for the single object task (Fig. 9, Fig. 10, Fig. 15) as well as the single view appearance prediction experiments with ground truth geometry (Fig. 12 Fig. 13, Fig. 14) and with reconstructed geometry (Fig. 16). We also show additional results for real input images in Fig. 17. Additional latent space interpolations of our generative model can be found in in Fig. 18.

A qualitative comparison of our models trained on only two views with one light configuration and ten views with four light configurations for each view is depicted in Fig. 19. The models trained on two training views show less accurate shadows and reflections compared to the models trained on all views. However, even when using only two views per object during training our model is able to capture the coarse overall appearance of the object.

Figure 9. **Single Object per Model.** In each column the light location is kept fixed, while the viewpoint is varied. Conversely in each row the viewpoint is kept fixed, while the light location is varied. The lateral parts of the calculator showcase accurate specularities. Both objects demonstrate perservance of high-frequency texture details. In particular even the calculator's button labels are readable (when zooming in).

Figure 10. **Single Object per Model.** In each column the light location is kept fixed, while the viewpoint is varied. Conversely in each row the viewpoint is kept fixed, while the light location is varied. The 3D scanned model of a human demonstrates that the method scales to objects with detailed geometry. Please note how the model learns that the metal parts of the barrel have different reflective properties than the wooden parts, which can be seen by the presence of pronounced specularties on the metal parts right next to wooden parts with significantly less specularities.

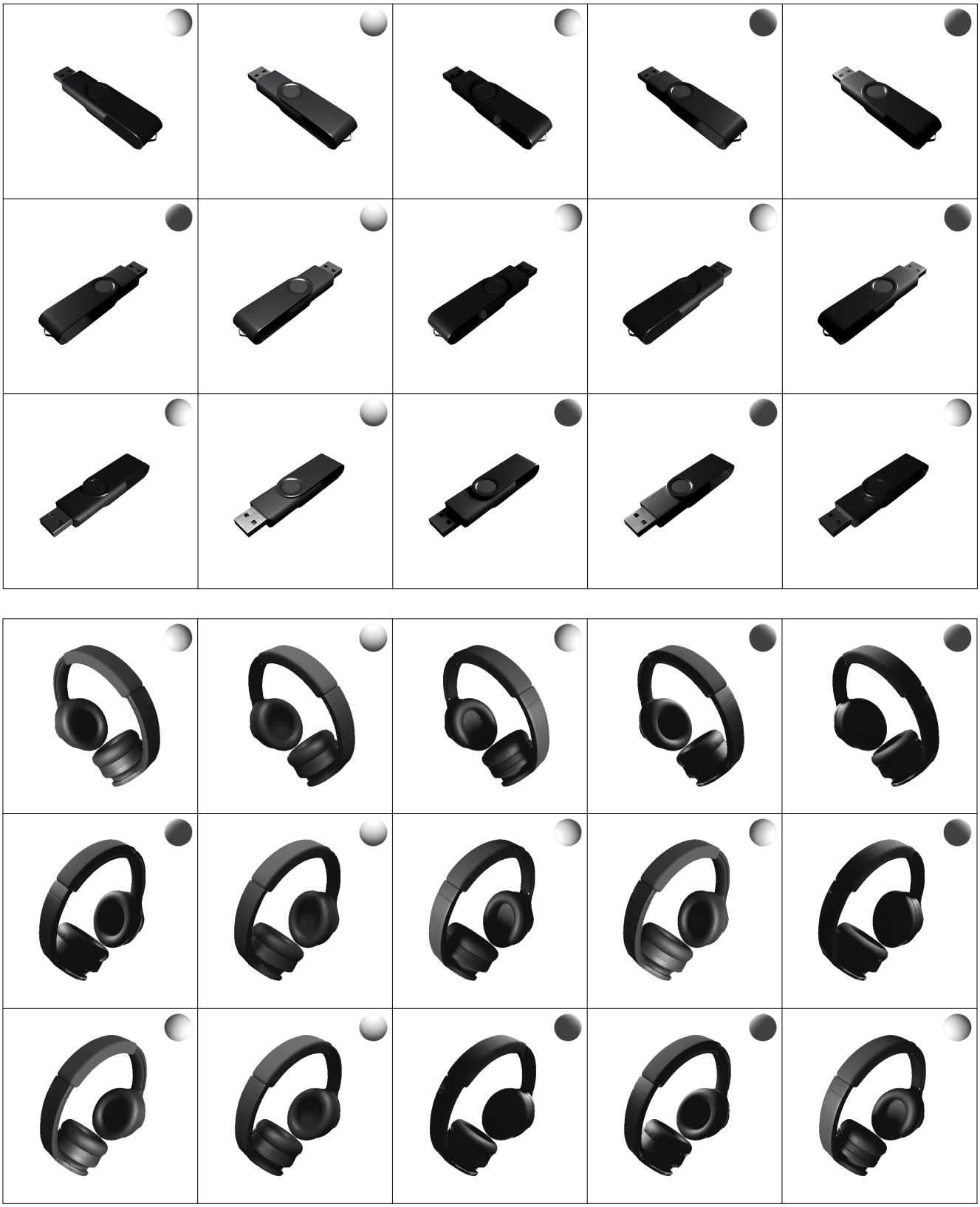

Figure 11. **Single Object per Model.** In each column the light location is kept fixed, while the viewpoint is varied. Conversely in each row the viewpoint is kept fixed, while the light location is varied. These renderings demonstrate that our method picks up on subtle differences between materials. For instance the metal parts are rendered differently than the plastic parts of the USB stick.

Figure 12. **Single View Appearance Prediction.** Results of our 2-step model for predicting the appearance from a single image of an unseen object of photoshapes. We show predictions for four different light configurations.

Input Image Predictions

Figure 13. **Single View Appearance Prediction.** Results of our model for predicting the appearance from a single image of an unseen object of the car class. We show predictions for four different light configurations.

Input Image Predictions

Figure 14. **Single View Appearance Prediction.** Results of our model for predicting the appearance from a single image of an unseen object of the lamp and sofa class. We show predictions for four different light configurations.

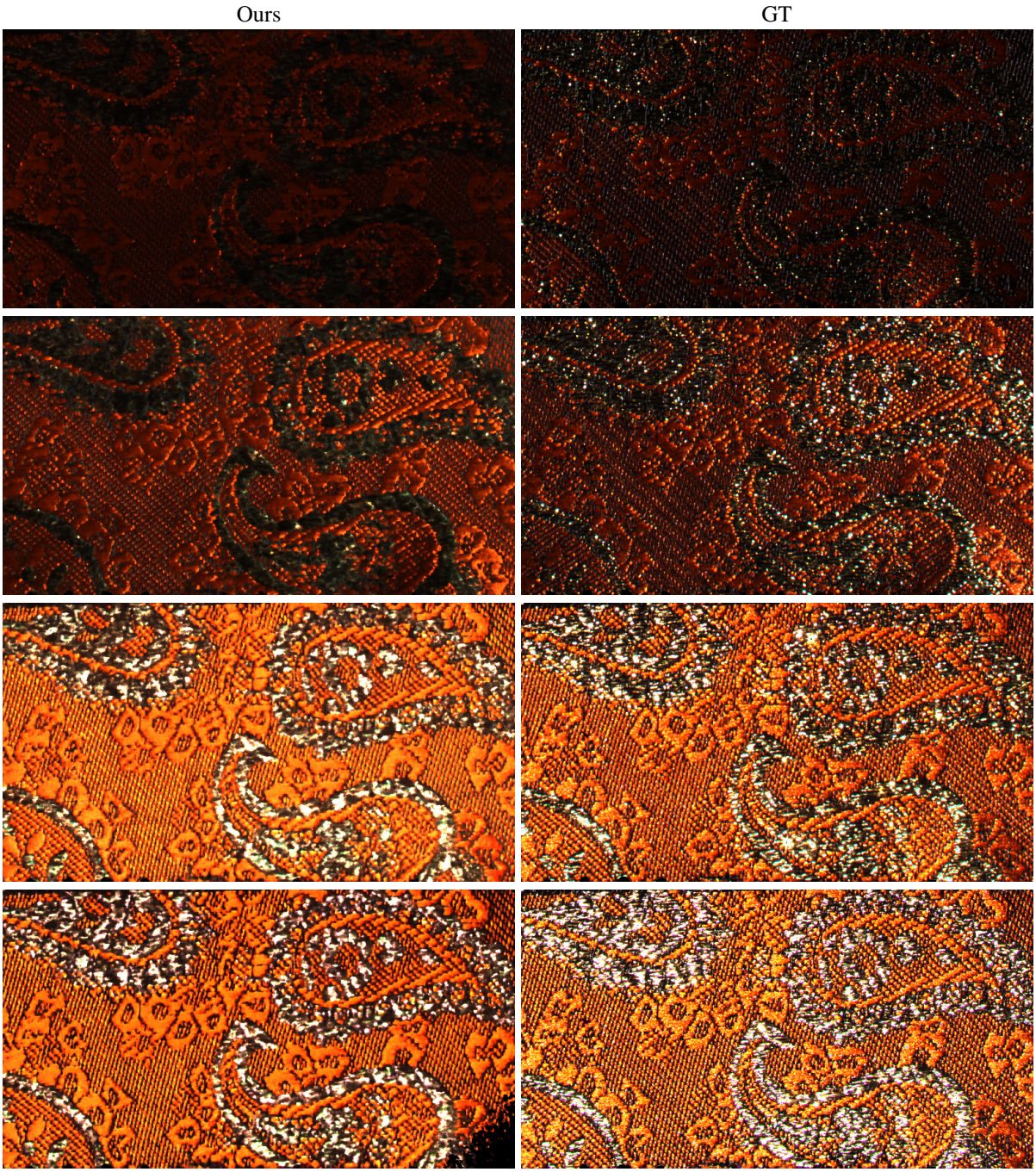

Figure 15. **Test samples for the planar svBRDF model.** Outputs of our method for novel viewing directions and light locations for the planar svBRDF sample.

Figure 16. **Image-based Reconstruction.** We show reconstructions of both geometry and appearance from a single input image for four different light configurations.

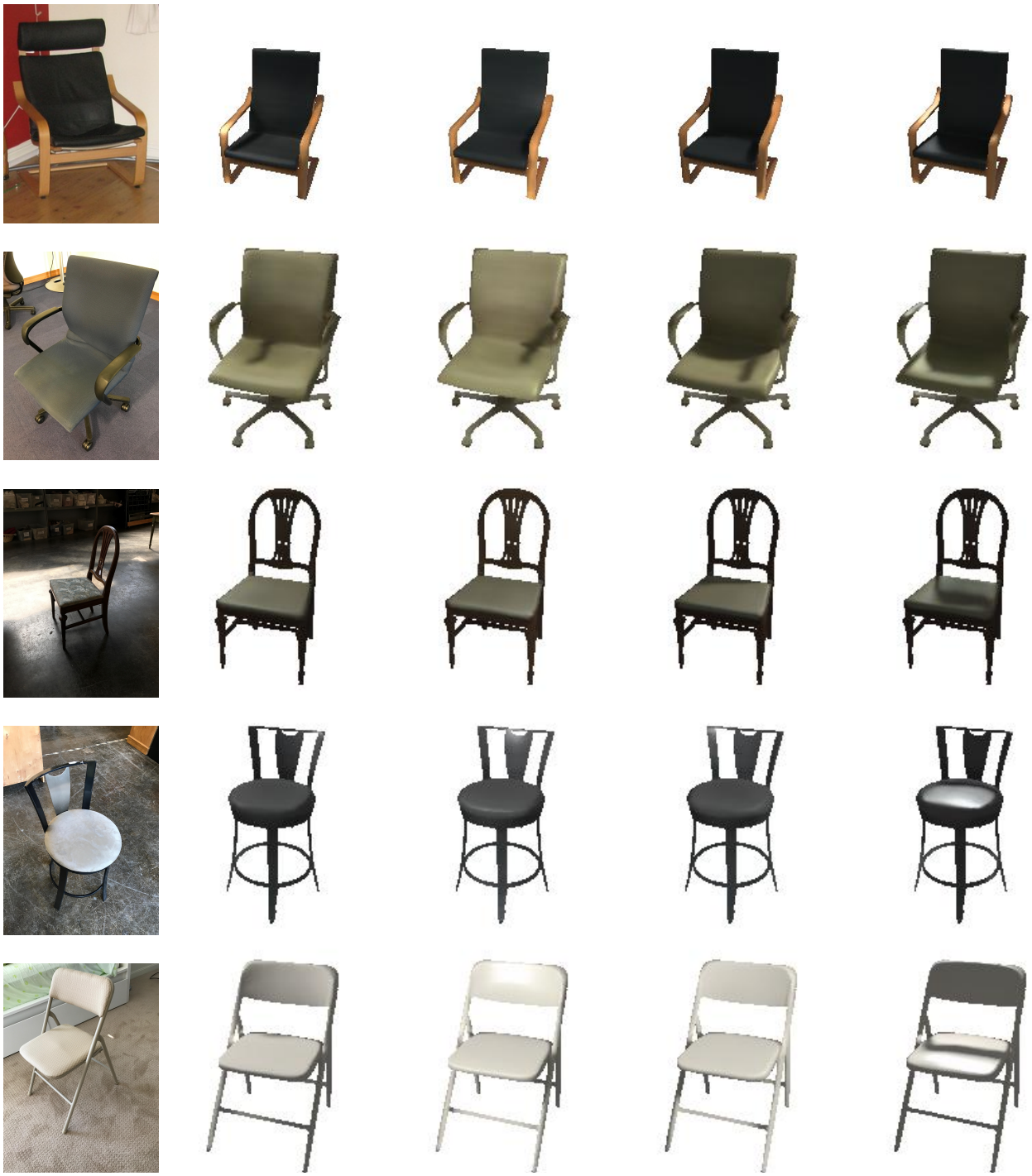

Figure 17. **Real Images.** Results of our method when using real images of unseen objects as input. We use images and corresponding 3D geometries from the Pix3D dataset [13].

Figure 18. **Generative Model.** We show predictions of our generative model for interpolated latent codes. The light location is fixed for all predictions.

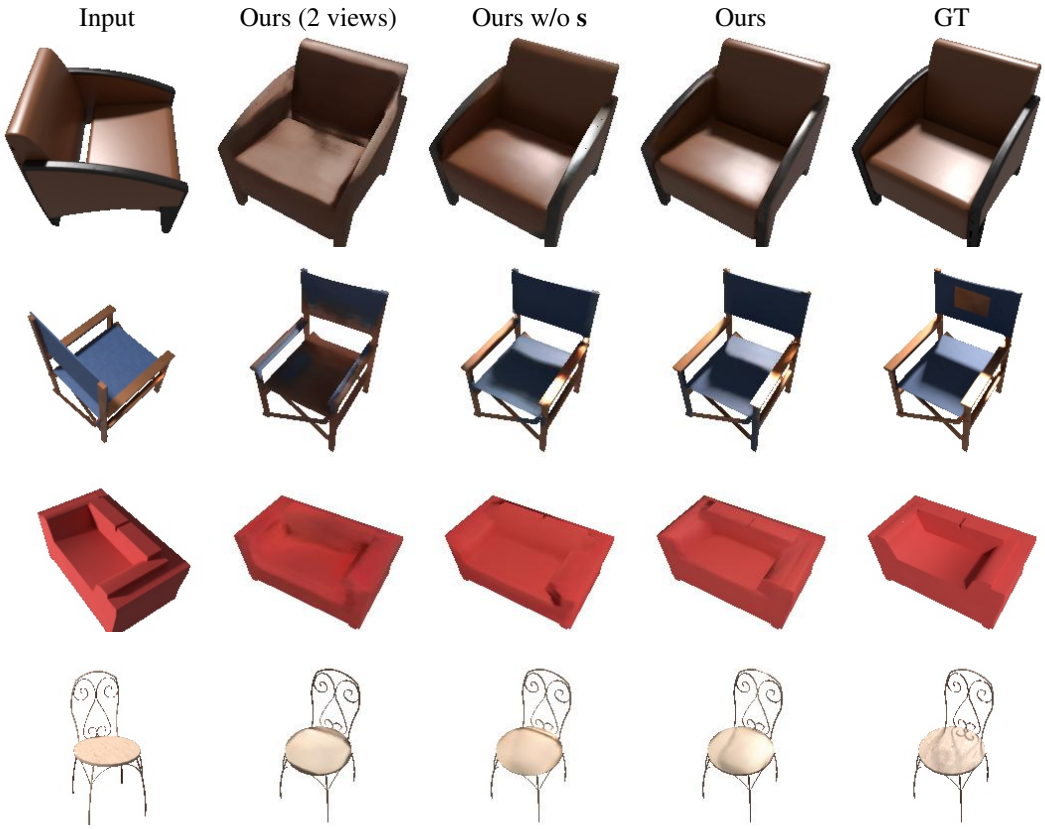

Figure 19. **Single View Appearance Prediction.** Qualitative comparison of our method trained on two views and all training views per objects as well as trained without shape encoding $\mathbf{s}$. While the models trained on two views predict reasonable texture, they show less accurate shadows and specular reflections.

References

[1] M. Anderson, R. Motta, S. Chandrasekar, and M. Stokes. Proposal for a standard default color space for the internet - srgb. In *4th Color and Imaging Conference, CIC 1996*, 1996.

[2] Blender Online Community. *Blender - a 3D modelling and rendering package*. Blender Foundation, Blender Institute, Amsterdam,

[3] M. Deserno. How to generate equidistributed points on the surface of a sphere. 2004.

[4] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[5] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *Proc. of the International Conf. on Learning Representations (ICLR)*, 2015.

[6] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *Proc. of the International Conf. on Learning Representations (ICLR)*, 2014.

[7] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2012.

[8] L. Mescheder, M. Oechsle, M. Niemeyer, S. Nowozin, and A. Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[9] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng. NeRF: Representing scenes as neural radiance fields for view synthesis. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2020.

[10] K. Park, K. Rematas, A. Farhadi, and S. M. Seitz. Photoshape: Photorealistic materials for large-scale shape collections. In *ACM Trans. on Graphics*, 2018.

[11] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[12] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, 2015.

[13] X. Sun, J. Wu, X. Zhang, Z. Zhang, C. Zhang, T. Xue, J. B. Tenenbaum, and W. T. Freeman. Pix3d: Dataset and methods for single-image 3d shape modeling. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018.