Supplementary Material: I^2-SDF: Intrinsic Indoor Scene Reconstruction and Editing via Raytracing in Neural SDFs

Jingsen Zhu[1] Yuchi Huo[2,1*] Qi Ye[3,6] Fujun Luan[4] Jifan Li[1] Dianbing Xi[1]
Lisha Wang[1] Rui Tang[5] Wei Hua[2] Hujun Bao[1] Rui Wang[1*]
[1]State Key Lab of CAD&CG, Zhejiang University [2]Zhejiang Lab [3]Zhejiang University
[4]Adobe Research [5]KooLab, Manycore [6]Key Lab of CS&AUS of Zhejiang Province

https://jingsenzhu.github.io/i2-sdf

1. Network Details

1.1. Architecture

All neural fields in our network are implemented by multi-layer perceptrons (MLPs). For neural SDF field F_d and radiance field F_c, we follow VolSDF [7]'s default setting, where F_d is a 8-layer MLP with hidden dimension 256 and F_c is a 4-layer MLP with hidden dimension 256. The dimension of the latent code output by F_d (*i.e.* $\mathbf{z}(\mathbf{x})$) is 256, and a skip connection is used in the 4$^{\text{th}}$ layer in F_d. The input position $\mathbf{x}$ and view direction $\mathbf{v}$ are encoded by sinusoidal positional encoding [3], with a maximum frequency band of 6 for $\mathbf{x}$ and 4 for $\mathbf{v}$, the same as in VolSDF [7]. The emitter semantic field F_e is a 2-layer MLP with hidden dimension 128, while material fields F_ρ, F_a are 3-layer MLPs with hidden dimension 64.

1.2. Implementation Details

The network model, as well as the training and evaluation scripts, are implemented with Pytorch [4]. The network is trained per-scene on a single NVIDIA Tesla V100 GPU. We adopt two stage training scheme, the training details of the 2 stages are as follows:

Training of geometry and radiance fields. We jointly optimize SDF network F_d, radiance network F_c and emitter semantic network F_e in this stage. We optimize our model for 200k iterations in this stage, which takes about 15 hours for a scene. The training loss

$$\begin{aligned}\mathcal{L}_{\text{geo}} =&\lambda_1\mathcal{L}_{\text{eikonal}} + \lambda_2\mathcal{L}_{\text{depth}} + \lambda_3\mathcal{L}_{\text{normal}} \\ &+ \lambda_4\mathcal{L}_{\text{smooth}} + \lambda_5\mathcal{L}_{\text{bubble}} \qquad (1)\\ \mathcal{L}_1 =&\mathcal{L}_{\text{rgb}} + \lambda_{\text{geo}}\mathcal{L}_{\text{geo}} + \lambda_{\text{emi}}\mathcal{L}_{\text{emi}} \qquad (2)\end{aligned}$$

where the weight hyperparameters are $\lambda_{\text{geo}} = 1$, $\lambda_{\text{emi}} = 0.5$, and $\lambda_2 = 0.1, \lambda_3 = 0.05$, respectively. For $\lambda_1, \lambda_4, \lambda_5$, since the training process is further divided into 3 steps (*i.e.* warm-up, bubble and smooth), their values are adjusted during the training accordingly:

1. In warm-up step, $\lambda_5 = 0$, $\lambda_1 = 0.1$, $\lambda_4 = 0$.
2. In bubble step, $\lambda_1 = \lambda_4 = 0$, $\lambda_5 = 0.5$.
3. In smooth step, $\lambda_5 = 0$, $\lambda_1 = 0.1$, $\lambda_4 = 0.01$.

The number of iterations assigned to the three steps are 50k, 100k and 50k in sequence.

We use error-guided adaptive sampling in bubble step, where the pruning threshold $P_{\min} = 0.05$.

Training of material and emission fields. In this stage, we use importance sampling and Monte Carlo estimation to compute the rendering result. We generate N outgoing rays to perform Monte Carlo integration. In practice, the sample rate N is set to 16, which is a trade-off between quality and performance.

We jointly optimize $F_a, F_\rho, \mathbf{L}[\cdot]$ for 100k iterations. The bottleneck of computational cost lies in the prediction of incident radiance $L^k_{\mathbf{s}}$, which grows proportional to the sample rate N. With $N = 16$, the training lasts for about 2-3 days.

2. Dataset Details

Our synthetic dataset contains 12 scenes in total: 6 bedrooms, 2 living rooms, 2 dining rooms and 2 kitchens. All the scenes are well-designed by artists with detailed geometry and fine textures. We use GPU-accelerated path tracing algorithm [1] to render the images, which can create photo-realistic rendering results with global illumination. All data

*Corresponding author.

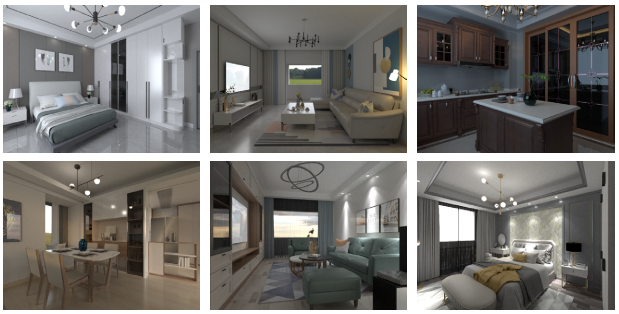

Figure 1. Display of indoor scenes in our dataset.

are rendered on a NVIDIA RTX 3090 GPU, with 4096 samples per pixel (spp). The rendering time is roughly 15 seconds per image. Fig. 1 displays some of the indoor scenes in our dataset.

All images in our dataset are annotated by ground truth camera intrinsics and poses, normal maps, depth maps and emitter semantic masks.

3. BRDF Model

We use GGX microfacet BRDF model [5] to approximate the surface reflection properties by a set of material parameters, including diffuse albedo K_d, specular albedo K_s and roughness ρ. In our implementation, we refer to Unreal Engine [2]'s implementation of microfacet BRDF model. The BRDF (bidirectional reflectance distribution function) $f_r(\mathbf{v}, \mathbf{d}; N, K_d, K_s, \rho)$ (where $\mathbf{v}$ and $\mathbf{d}$ are view and lighting directions, and N is the surface normal) can be decomposed into diffuse and specular components and computed by

$$f_r(\mathbf{v}, \mathbf{d}) = f_d(K_d) + f_s(\mathbf{v}, \mathbf{d}; N, K_s, \rho) \tag{3}$$

$$f_d(K_d) = \frac{K_d}{\pi} \quad \alpha = \rho^2 \tag{4}$$

$$f_s(\mathbf{v}, \mathbf{d}; N, K_s, \rho) = D(\alpha, N, h) G_2(\alpha, N, \mathbf{v}, \mathbf{d}) F(K_s, \mathbf{d}, h) \tag{5}$$

where f_d and f_s are the diffuse and specular components respectively, while D, G_2, F are the distribution, Fresnel and geometric terms, defined as

$$D(\alpha, \mathbf{N}, \mathbf{h}) = \frac{\alpha^2}{\pi((\alpha^2 - 1)(\mathbf{N} \cdot \mathbf{h})^2 + 1)^2} \tag{6}$$

$$S(\alpha, \mathbf{N}, \mathbf{v}, \mathbf{d}) = (\mathbf{N} \cdot \mathbf{d}) \sqrt{\alpha^2 + (\mathbf{N} \cdot \mathbf{v})^2 (1 - \alpha^2)} \tag{7}$$

$$G_2(\alpha, \mathbf{N}, \mathbf{v}, \mathbf{d}) = \frac{1}{2(S(\alpha, \mathbf{N}, \mathbf{v}, \mathbf{d}) + S(\alpha, \mathbf{N}, \mathbf{d}, \mathbf{v}))} \tag{8}$$

$$\mathrm{lum}(C) = 0.213 C.\mathrm{r} + 0.715 C.\mathrm{g} + 0.072 C.\mathrm{b} \tag{9}$$

$$F_{90}(K_s) = \min(\frac{\mathrm{lum}(K_s)}{0.04}, 1) \tag{10}$$

$$F(K_s, \mathbf{N}, \mathbf{d}) = K_s + (F_{90}(K_s) - K_s)(1 - (\mathbf{N} \cdot \mathbf{d}))^5 \tag{11}$$

In importance sampling and Monte Carlo integration, we also need to calculate the PDF value $p(\mathbf{v}, \mathbf{d})$ corresponding to the view and lighting direction:

$$w_d = \frac{\mathrm{lum}(K_d)}{\mathrm{lum}(K_d) + \mathrm{lum}(K_s)} \tag{12}$$

$$p(\mathbf{v}, \mathbf{d}) = w_d p_d(\mathbf{v}, \mathbf{d}) + (1 - w_d) p_s(\mathbf{v}, \mathbf{d}) \tag{13}$$

$$p_d(\mathbf{v}, \mathbf{d}) = \frac{N \cdot \mathbf{d}}{\pi} \tag{14}$$

$$G_1(\alpha, N, \mathbf{v}) = \frac{2}{\sqrt{1 + \frac{\alpha^2 (1 - (\mathbf{N} \cdot \mathbf{v})^2)}{(\mathbf{N} \cdot \mathbf{v})^2}} + 1} \tag{15}$$

$$p_s(\mathbf{v}, \mathbf{d}) = \frac{D(\alpha, N, h) G_1(\alpha, N, \mathbf{v})}{4(\mathbf{N} \cdot \mathbf{v})} \tag{16}$$

where p_d and p_s are the diffuse and specular components, which are mixed according to the luminance of K_d and K_s.

4. Details of Bubbling and Adaptive Sampling

Intermediate results in bubble step. Fig. 2 presents the process of how the missing objects are reconstructed by our bubbling method. The chandelier is missing initially at 50k iterations. In the early stage of bubble step, the light balls are reconstructed rapidly (55k), since they are relatively large inside the chandelier. On the other hand, thin components (*e.g.* poles) grow slowly (70k). Eventually (150k) the entire chandelier is successfully reconstructed.

Visualization of error-guided sampling map during training. As shown in Fig. 3, at the training iteration of 50k (*i.e.* the start of bubble step), the chandelier is completely ignored and thus the corresponding pixels in the PDF map have high values. As the training proceeds to 80k iterations, the light balls have already been well-reconstructed, with chandelier poles still missing. Therefore, the value of pixels corresponding to light balls are reduced to 0, whereas chandelier pole pixels still need to be further sampled.

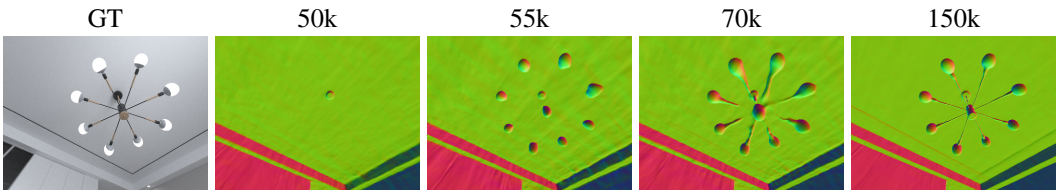

Figure 2. **Intermediate results in bubble step.**

Table 1. **Comparisons of per-scene novel view PSNR.**

	Ours	NeRF	Instant-NGP	NeuRIS	MonoSDF
syn_1	28.03	26.24	25.8	25.53	27.37
syn_2	30.09	29.27	27.34	24.30	24.83
syn_3	27.46	25.73	26.58	24.30	24.48
syn_4	29.64	27.99	25.95	26.93	26.67
syn_5	27.71	26.93	16.82	24.38	26.03
syn_6	28.55	27.27	16.27	25.21	26.49
syn_7	28.04	27.65	24.67	25.18	24.67
syn_8	27.83	25.31	27.72	24.36	25.39
mean	**29.70**	27.09	23.89	25.02	25.74
real_1	26.63	26.48	19.10	25.91	26.21
real_2	28.01	27.21	26.32	23.87	24.38
real_3	24.58	24.11	23.29	23.31	22.72
real_4	21.39	20.86	19.23	19.82	20.05
mean	**25.15**	24.66	21.99	23.22	23.34

Table 2. **Comparisons of per-scene normal angular error and depth L_1 loss.**

	Normal-Angular-L_1 ↓			Depth-L_1 ↓		
	Ours	NeuRIS	MonoSDF	Ours	NeuRIS	MonoSDF
syn_1	0.040	0.051	0.036	0.014	0.240	0.010
syn_2	0.030	0.041	0.035	0.019	0.331	0.048
syn_3	0.054	0.080	0.073	0.021	0.319	0.061
syn_4	0.053	0.071	0.065	0.068	0.312	0.103
syn_5	0.064	0.096	0.058	0.025	0.227	0.034
syn_6	0.057	0.082	0.054	0.051	0.355	0.043
syn_7	0.057	0.071	0.076	0.065	0.335	0.099
syn_8	0.072	0.071	0.064	0.016	0.274	0.033
mean	**0.053**	0.070	0.058	**0.035**	0.299	0.054

5. Additional Experimental Results

Novel view synthesis. Tab. 1 displays quantitative results of per-scene novel view PSNR. It turns out that our method outperforms all of the baselines, benefiting from our precise reconstruction of small objects and proper handling of shape-radiance ambiguity. Qualitative results are presented in Fig. 6. NeRF, NeuRIS and MonoSDF fails to reconstruct small objects such as chandeliers and lamp poles, while NeRF and Instant-NGP also sufffers from fractured reconstruction results. While Instant-NGP usually capture most high-frequency details, it likely overfits to single-view radiance and fails to ensure multi-view geometry consistency in indoor scenes, leading to poor novel view synthesis results with floating artifacts.

Geometry Reconstruction. Tab. 2 displays quantitative comparisons of per-scene normal angular L_1 error and depth L_1 error between our method and baselines. The definition of normal angular L_1 error is

$$\mathcal{L}_{\text{normal}} = \|1 - \hat{N} \cdot N\|_1 \tag{17}$$

Our method also outperforms NeuRIS and MonoSDF, indicating superior 3D reconstruction quality. Fig. 7 presents

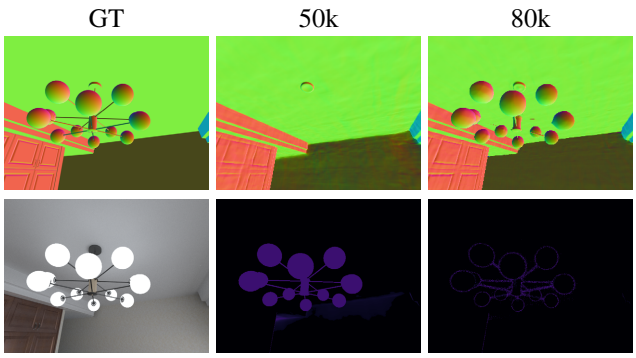

Figure 3. **Visualization of error-guided sampling map during training.** The first row displays intermediate normal reconstruction results, while the second row displays the corresponding error PDF map.

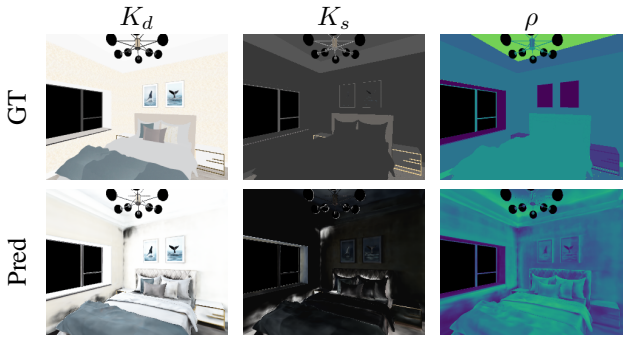

Figure 4. **Comparisons between GT material and predictions.** Note that our optimization does not require supervision from GT material dataset but we still produce plausible results.

qualitative comparisons of the reconstructed normal maps. Our method can even recover high-frequency details on geometry, such as the spikes on the ball.

Material Decomposition. Figs. 4 and 8 presents qualitative results of the decomposed diffuse albedo K_d, specular albedo K_s and roughness ρ. The material parameters are not directly supervised by GT labels, but we produce plausible results.

Scene editing. With the intrinsic decomposition results, we can enable photo-realistic scene editing tasks such as material editing and relighting. Fig. 9 shows qualitative results of scene editing results in both real and synthetic data. We explore mirror insertion (top-left and bottom-right), texture editing (top-right and mid-left), object insertion (mid-right) and relighting (bottom-left). Note that the edited specular reflections (on mirrors and inserted metal ball) are consistent with the surroundings. On account of our raytracing algorithm, our method is capable of casting shadows of the inserted object (see the shadows of the inserted ball on the sofa).

References

[1] James T Kajiya. The rendering equation. In *Proceedings of the 13th annual conference on Computer graphics and interactive techniques*, pages 143–150, 1986. 1

[2] Brian Karis and Epic Games. Real shading in unreal engine 4. *Proc. Physically Based Shading Theory Practice*, 4(3):1, 2013. 2

[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] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019. 1

[5] Bruce Walter, Stephen R Marschner, Hongsong Li, and Kenneth E Torrance. Microfacet models for refraction through rough surfaces. *Rendering techniques*, 2007:18th, 2007. 2

[6] Jiepeng Wang, Peng Wang, Xiaoxiao Long, Christian Theobalt, Taku Komura, Lingjie Liu, and Wenping Wang. Neuris: Neural reconstruction of indoor scenes using normal priors. *arXiv preprint*, 2022. 5

[7] Lior Yariv, Jiatao Gu, Yoni Kasten, and Yaron Lipman. Volume rendering of neural implicit surfaces. *Advances in Neural Information Processing Systems*, 34:4805–4815, 2021. 1

[8] Zehao Yu, Songyou Peng, Michael Niemeyer, Torsten Sattler, and Andreas Geiger. Monosdf: Exploring monocular geometric cues for neural implicit surface reconstruction. *arXiv:2022.00665*, 2022. 5

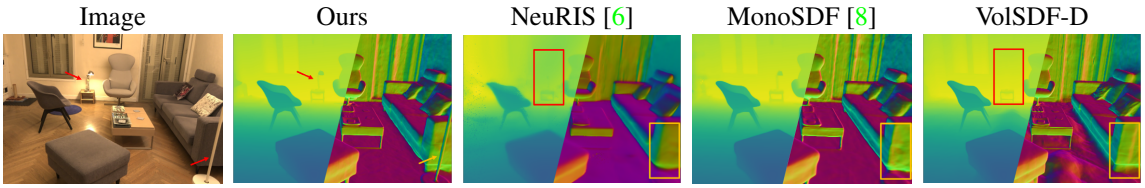

Figure 5. **Qualitative comparisons of reconstructed depth map and normal map.**

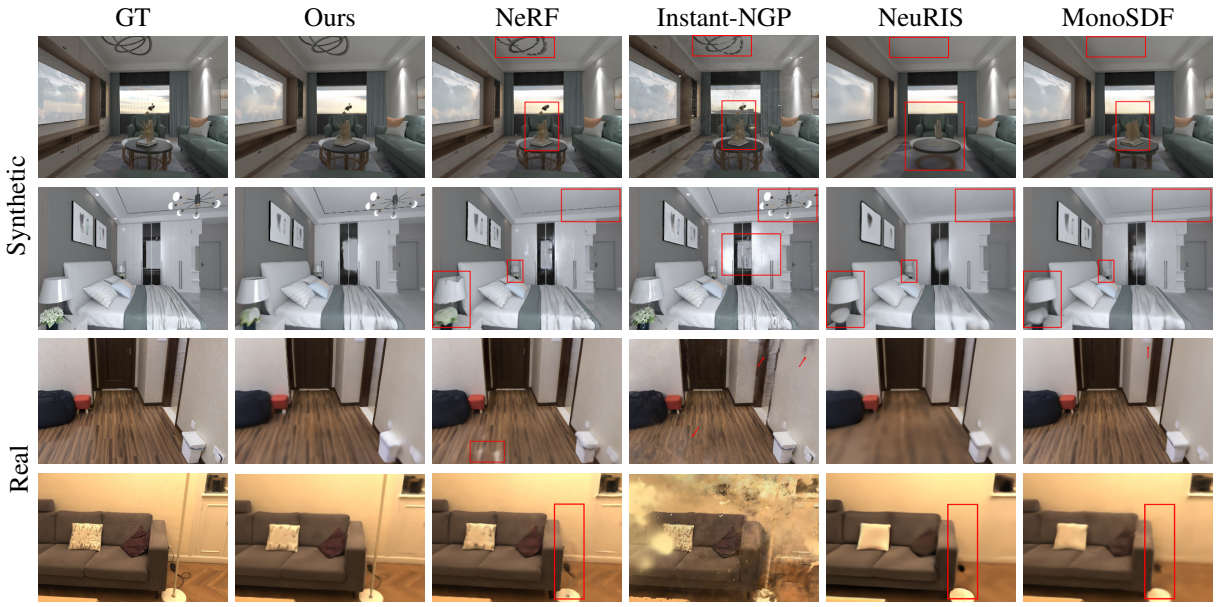

Figure 6. **Qualitative comparisons of novel view synthesis** on synthetic data and real data. Zoom in for details.

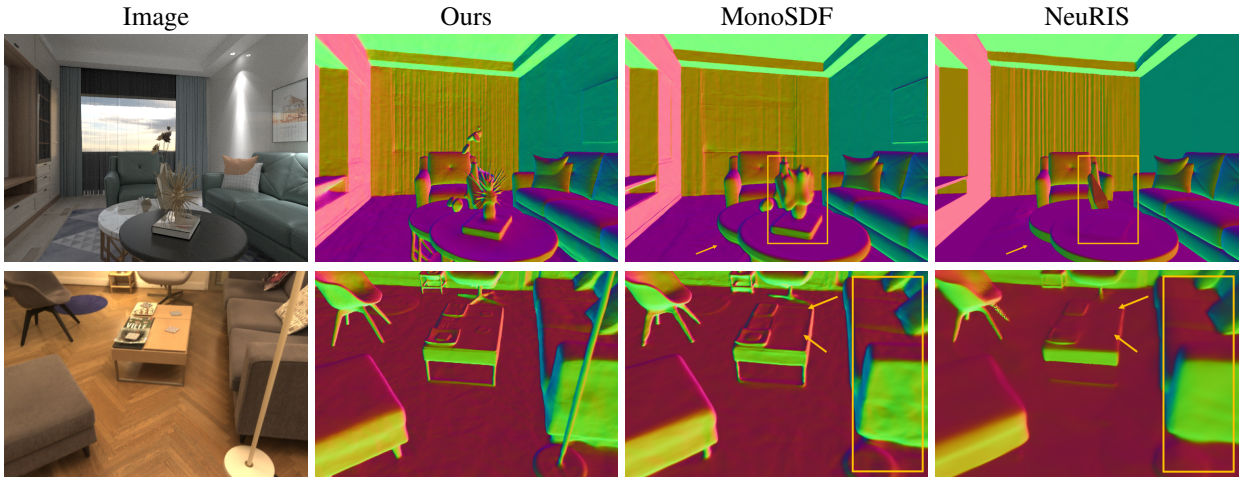

Figure 7. **Qualitative comparisons of normal estimation** on synthetic data and real data.

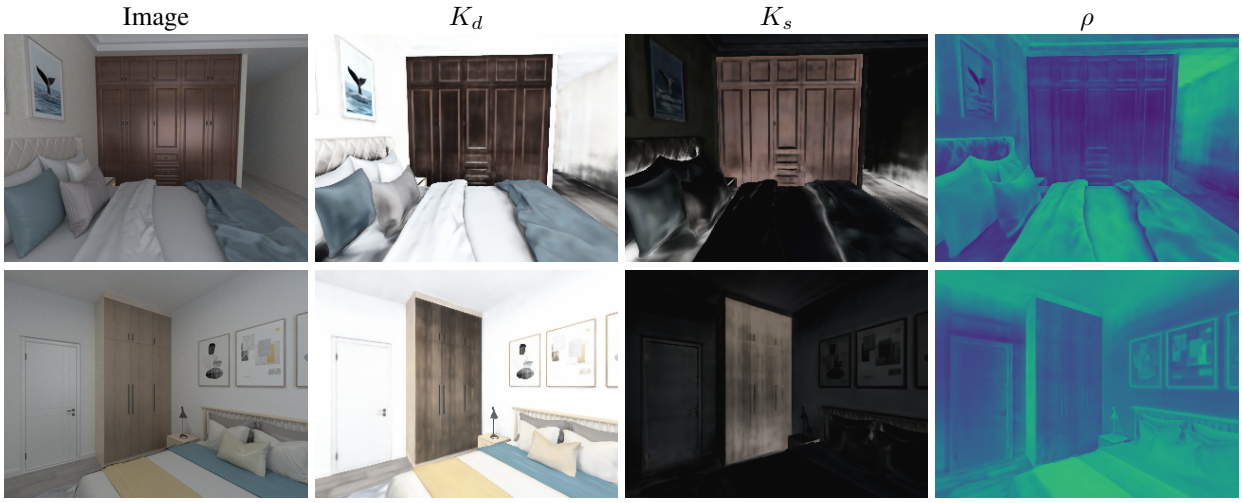

Figure 8. **Qualitative results of decomposed materials.**

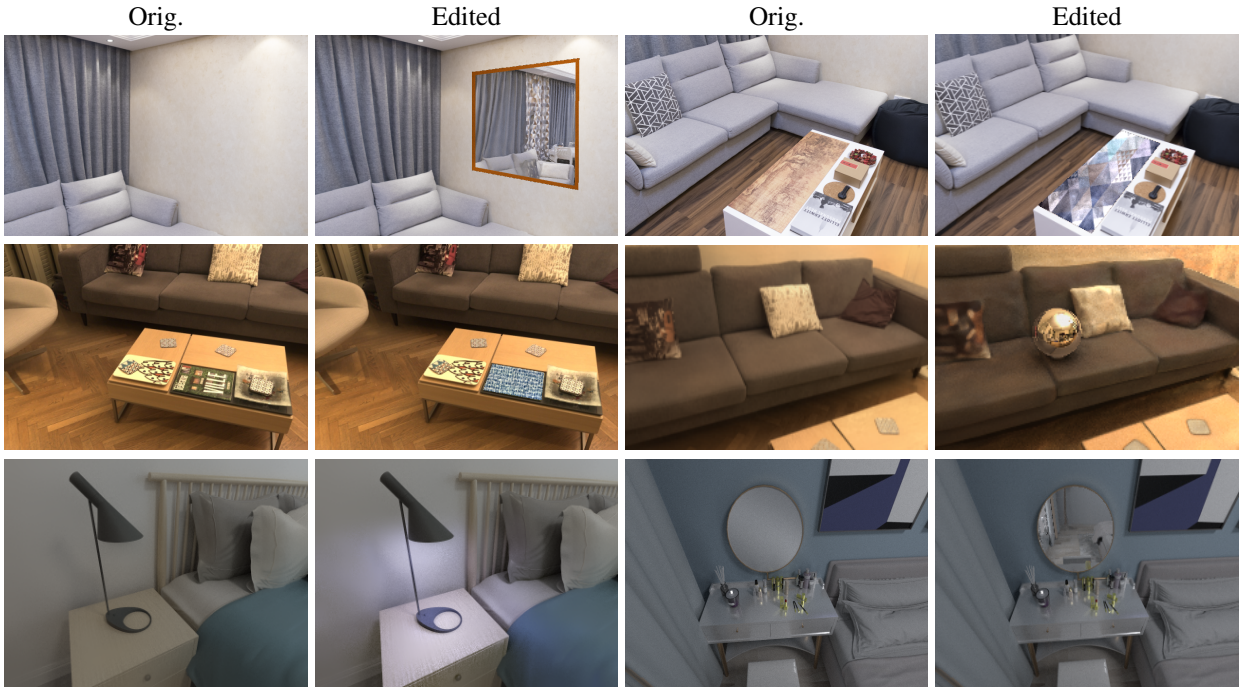

Figure 9. **Qualitative results of scene editing and relighting.**