Shape, Light, and Material Decomposition from Images using Monte Carlo Rendering and Denoising Supplemental Material

Jon Hasselgren
NVIDIA

Nikolai Hofmann
NVIDIA

Jacob Munkberg
NVIDIA

1 Optimization and Regularization

Image Loss Our renderer uses physically-based shading and produces images with high dynamic range. Therefore, the objective function must be robust to the full range of floating-point values. Following recent work in differentiable rendering [4, 11], our image space loss, L_{image}, computes the L_1 norm on tonemapped colors. As tone map operator, T, we transform linear radiance values, x, according to $T(x) = \Gamma(\log(x+1))$, where $\Gamma(x)$ is the sRGB transfer function [16]:

$$\begin{aligned} \Gamma(x) &= \begin{cases} 12.92x & x \leq 0.0031308 \\ (1+a)x^{1/2.4} - a & x > 0.0031308 \end{cases} \\ a &= 0.055. \end{aligned} \tag{1}$$

Regularizers While it is desirable to minimize regularization, in our setting with multi-view images with constant lighting, we have to rely on several priors to guide optimization towards a result with a good separation of geometry, materials, and lighting. As mentioned in the paper, we rely on smoothness regularizers for albedo, $\boldsymbol{k}_d$, specular parameters, $\boldsymbol{k}_{\text{orm}}$ and geometric surface normal, $\mathbf{n}$, following:

$$L_{\boldsymbol{k}_d} = \frac{1}{|\mathbf{x}_{\text{surf}}|} \sum_{\mathbf{x}_{\text{surf}}} \left| \boldsymbol{k}_d\left(\mathbf{x}_{\text{surf}}\right) - \boldsymbol{k}_d\left(\mathbf{x}_{\text{surf}} + \boldsymbol{\epsilon}\right) \right|, \tag{2}$$

where $\mathbf{x}_{\text{surf}}$ is a world space position on the surface of the object and $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \sigma = 0.01)$ is a small random displacement vector. Regularizing the geometric surface normal (before normal map perturbation) is novel compared to NVDIFFREC and helps enforce smoother geometry, particularly in early training.

We note that normal mapping (surface normals perturbed through a texture lookup) also benefits from regularization. While normal mapping is a powerful tool for simulating local micro-geometry, the decorrelation of the geometry and surface normal can be problematic. In some optimization runs, we observe that with normal mapping enabled, the environment light is (incorrectly) leveraged as a color dictionary, where the normal perturbation re-orients the normals of a geometric element to look up a desired color. To prevent this behavior, we use the following regularizer. Given a normal perturbation $\mathbf{n}'$ in tangent space, our loss is defined as

$$L_{\mathbf{n}'} = \frac{1}{|\mathbf{x}_{\text{surf}}|} \sum_{\mathbf{x}_{\text{surf}}} 1 - \underbrace{\frac{\mathbf{n}'\left(\mathbf{x}_{\text{surf}}\right) + \mathbf{n}'\left(\mathbf{x}_{\text{surf}} + \boldsymbol{\epsilon}\right)}{\left|\mathbf{n}'\left(\mathbf{x}_{\text{surf}}\right) + \mathbf{n}'\left(\mathbf{x}_{\text{surf}} + \boldsymbol{\epsilon}\right)\right|}}_{\text{Half-angle vector}} \cdot (0,0,1). \tag{3}$$

Intuitively, we enforce that normal perturbations, modeling micro-geometry, randomly selected in a small local area, have an expected value of the unperturbed tangent space surface normal, $(0,0,1)$.

Project page: https://nvlabs.github.io/nvdiffrecmc/

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

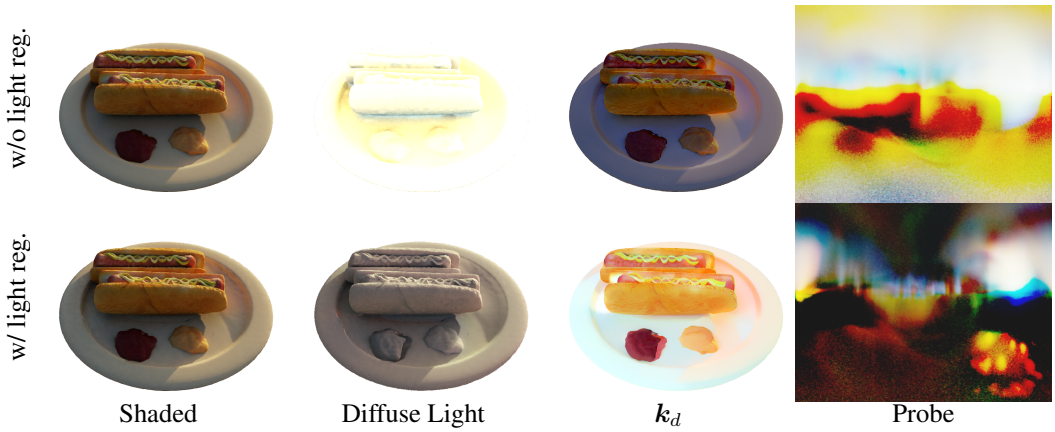


Figure 1: The impact of our light regularizer on the Hotdog scene from the synthetic NeRF dataset. The regularizer is particularly effective in this example, as the dataset contains high frequency lighting with sharp shadows. As shown in the top row, without regularization most of the lighting and shadow is baked into the $\boldsymbol{k}_d$ texture because it is the quickest way for the optimizer to minimize loss. While our regularized result is not perfect (some texture detail on the hotdog bun is incorrectly baked into lighting), the lighting is accurately captured and $\boldsymbol{k}_d$ contains mostly chrominance, as expected. The light probe is also significantly more detailed.

As mentioned in the paper, we additionally regularize based on monochrome image loss between the demodulated lighting terms and the reference image:

$$L_{\text{light}} = |Y\left(T(\mathbf{c}_d + \mathbf{c}_s)\right) - V\left(T(I_{\text{ref}})\right)|_1. \tag{4}$$

Here, T is the tonemap operator described in the previous paragraph. $\mathbf{c}_d$ is demodulated diffuse lighting, $\mathbf{c}_s$ is specular lighting, and I_{ref} is the target reference image. Monochrome images are computed through the simple luminance operator, $Y\left(\mathbf{x}\right) = \left(\mathbf{x}_r + \mathbf{x}_g + \mathbf{x}_b\right)/3$, and HSV-value, $V\left(\mathbf{x}\right) = \max\left(\mathbf{x}_r, \mathbf{x}_g, \mathbf{x}_b\right)$. While the regularizer is limited by our inability to demodulate the reference image, we show in Figure 1 that it greatly increases lighting detail, which is particularly effective in datasets with high frequency lighting. Figure 18 shows the impact of the regularizer on a larger set of scenes.

We compute the final loss as a weighted combination of the image loss and regularizer terms:

$$L = L_{\text{image}} + \underbrace{\lambda_{\boldsymbol{k}_d}}_{=0.1} L_{\boldsymbol{k}_d} + \underbrace{\lambda_{\boldsymbol{k}_{\text{orm}}}}_{=0.05} L_{\boldsymbol{k}_{\text{orm}}} + \underbrace{\lambda_{\mathbf{n}}}_{=0.025} L_{\mathbf{n}} + \underbrace{\lambda_{\mathbf{n}'}}_{=0.25} L_{\mathbf{n}'} + \underbrace{\lambda_{\text{light}}}_{=0.15} L_{\text{light}}, \tag{5}$$

where we show the default weights used for most results (unless otherwise noted).

Optimization details We use Adam [7] (default settings) to optimize parameters for geometry, material, and lighting. Referring to DMTet [14], geometry is parameterized as SDF values on a three-dimensional grid, with a perturbation vector per grid-vertex. Material and lighting are encoded in textures, or neural network (e.g., an MLP with positional encoding) encoded high-frequency functions. We typically use different learning rates for geometry, material, and lighting, with lighting having the highest learning rate and material the lowest. We closely follow the publically available NVDIFFREC code base and refer to that for details. In Figure 2, we visualize two examples of optimizing the light probe, and note that we can capture high-frequency lighting details for specular objects.

We note that the early phases of optimization (starting from random geometry with large topology changes) are crucial in producing a good final result. The shadow test poses a challenge since changes in geometry may drastically change the lighting intensity in a previously shadowed area, causing the optimization process to get stuck in bad local minima. This is shown in Figure 3, where the optimizer fails to carve out the shape of the object. We combat this by incrementally blending in the shadow term as optimization progresses.

Diffuse Specular

Figure 2: We compute direct lighting by integrating the BSDF over the hemisphere using Monte Carlo. For diffuse scenes (left), the BSDF has a wide lobe, and the extracted probe often contains significant noise. For specular scenes, the BSDF lobe is sharper, which allow us to recover high frequency lighting.

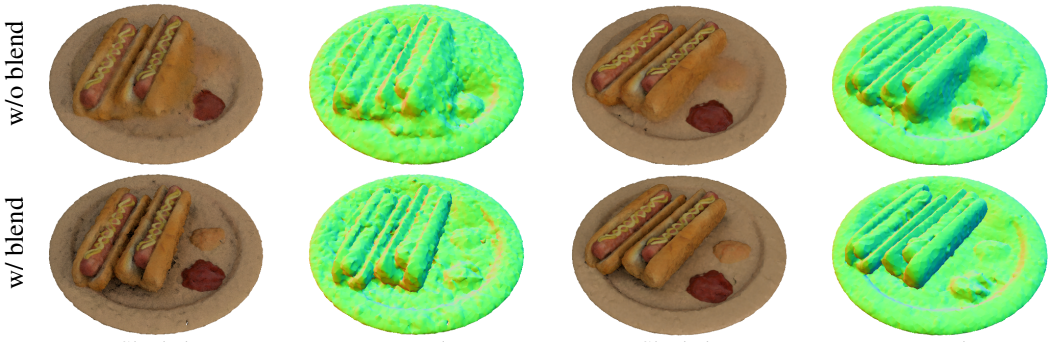


Figure 3: Illustration of the impact of gradually blending in the shadow term in the early stages of optimization. In early stages, there are large changes in topology, and local changes in geometry can have a large (global) impact on color and shading. As shown in the top row, early convergence suffers, and the optimization fails to carve out some geometry. By incrementally ramping up the shadow term, as shown in the bottom row, we improve geometry reconstruction.

Recall from the main paper that we compute the color according to the rendering equation:

$$L(\omega_o) = \int_\Omega L_i(\omega_i) f(\omega_i, \omega_o)(\omega_i \cdot \mathbf{n}) d\omega_i, \tag{6}$$

where $L_i(\omega_i) = L_i'(\omega_i)H(\omega_i)$ can be separated into a lighting term, $L_i'(\omega_i)$, and visibility term, $H(\omega_i)$. Rather than using binary visibility, we introduce a light leakage term, τ, and linearly fade in the shadow contributions over the first 1750 iterations:

$$H(\omega_i, \tau) = \begin{cases} 1-\tau & \text{if intersect\_ray}(\omega_i) \\ 1 & \text{otherwise} \end{cases}. \tag{7}$$

In Figure 3 we note that gradually blending in the shadow term has a large impact on early convergence. In particular, since we start from a random topology, carving out empty space or adding geometry may have a large impact on overall shading, causing spiky and noisy gradients.

The denoiser may also interfere with early topology optimization (blurred visibility gradients). This is particularly prominent when the denoiser parameters are trained along with scene parameters. Therefore, we similarly, ramp up the spatial footprint, σ, in the case of a bilateral denoiser. For neural denoisers, which have no easily configurable filter width, we instead linearly blend between the noisy and denoised images to create a smooth progression.

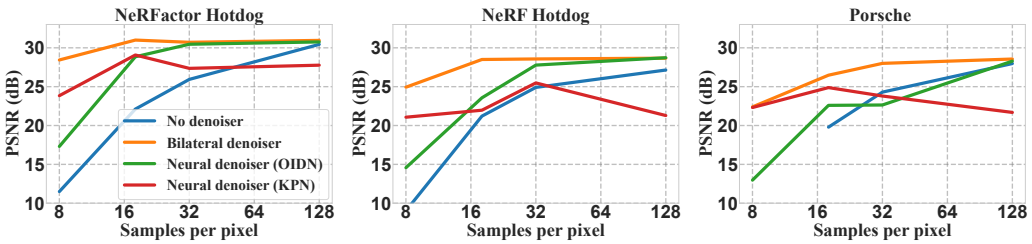


Figure 4: The effect of using different denoising algorithms during optimization at low sample counts on three different scenes of increasing complexity (from left to right). We plot averaged PSNR scores over 200 novel views, rendered without denoising, using high sample counts. The most complex scene (Porsche) failed to converge at 8 spp without denoising. This experiment is identical to the denoising ablation in the paper, except including results from a jointly optimized single-frame version of the kernel prediction network (KPN) architecture from Hasselgren et al. [5]. Inherent problems of live-training the denoiser, such as baking features into the denoiser weights instead of the desired parameters, become apparent from deteriorating results, even at higher sample counts.

2 Denoising

Bilateral Denoiser As a representative bilateral denoiser, we adapt the spatial component of Spatiotemporal Variance-guided Filtering [13] (SVGF), Our filter is given by

$$\hat{\mathbf{c}}\left(p\right) = \frac{\sum_{q \in \Omega} \mathbf{c}\left(q\right) \cdot w\left(p, q\right)}{\sum_{q \in \Omega} w\left(p, q\right)}, \tag{8}$$

where $\mathbf{c}$ are the pixel colors, $w\left(p, q\right)$ are the bilateral weights between pixels p and q, and Ω denotes the filter footprint. The bilateral weight is split into three components as follows

$$w\left(p, q\right) = \underbrace{e^{-\frac{|p-q|^2}{2\sigma^2}}}_{w} \underbrace{e^{-\frac{|z(p)-z(q)|}{\sigma_z |\nabla z(p) \cdot (p-q)|}}}_{w_z} \underbrace{\max(0, \mathbf{n}\left(p\right) \cdot \mathbf{n}\left(q\right))^{\sigma_{\mathbf{n}}}}_{w_{\mathbf{n}}}. \tag{9}$$

The spatial component, w, is a Gaussian filter with its footprint controlled by σ. We linearly increase σ from 1e-4 to 2.0 over the first 1750 iterations. The depth component, w_z implements an edge stopping filter based on image space depth. It is a soft test, comparing the depth differences between pixels $z(p)$ and $z(q)$ with a linear prediction based on the local depth gradient $\nabla z(p)$. The final component, $w_{\mathbf{n}}$, is computed as the scalar product over surface normals $\mathbf{n}(p)$ and $\mathbf{n}(q)$. Following SVGF, we use $\sigma_z = 1$ and $\sigma_{\mathbf{n}} = 128$. We also omitted the Á-Trous wavelet filtering of SVGF, which is primarily a run-time performance optimization, and evaluate the filter densely in the local footprint. We propagate gradients back to the renderer, i.e., the noisy input colors, but no gradients to the filter weights.

Neural denoisers For OIDN, we use the pre-trained weights from the official code [1], and wrap the model (a direct-prediction U-Net) in a PyTorch module to allow for gradient propagation to the input colors. We also propagate gradients to the network weights, in order to fine-tune the denoiser per-scene, but unfortunately, the direct-prediction architecture of OIDN made live-training highly unstable in our setting.

For the kernel-predicting neural denoiser, we leverage the architecture from Hasselgren et al. [5], but omit the adaptive sampling part and the recurrent feedback loop. The network weights were initialized to random values using Xavier initialization. The hierarchical kernel prediction in this architecture constrains the denoiser, and enables live-trained neural denoising in our setting. That said, in our evaluations, the live-trained variant produce worse reconstructions than the pre-trained denoisers.

We study the impact of different denoising algorithms on scene reconstruction, including jointly optimizing the denoising network, in Figure 4. Unfortunately, when fine-tuning the denoiser to the specific scene as part of optimization, we note a tendency for features to get baked into the denoiser's network weights instead of the desired scene parameters (overfitting), negatively impacting

Reference	No denoising	Bilateral	OIDN (pre-trained)	KPN (live-trained)
8 spp	26.24	36.15	35.94	35.21
18 spp	33.98	35.86	35.74	35.36
32 spp	35.87	35.72	35.65	35.19
128 spp	36.08	35.43	35.51	35.21

Figure 5: The effect of denoising when optimizing HDR environment lighting, using a diffuse version of the Hotdog scene. In this test, geometry and material are fixed to study the effect of optimizing the lighting in isolation. **Top**: Resulting light probes at 8 spp for each method tabulated above. **Bottom**: Tabulated average PSNR over 200 novel views using high sample count without denoising.

Reference	No denoising	Bilateral	OIDN (pre-trained)	KPN (live-trained)
8 spp	11.50	28.43	17.31	23.84
18 spp	22.12	30.99	28.85	29.07
32 spp	25.93	30.71	30.46	27.36
128 spp	30.44	30.95	30.76	27.76

Figure 6: Impact of denoising when jointly optimizing geometry, materials and environment lighting, using the NeRFactor Hotdog scene. **Top**: Reconstruction results at 32 spp for each of the methods tabulated above. **Bottom**: Tabulated average PSNR (dB) over 200 novel views using high sample count without denoising.

the reconstruction quality. We showcase the isolated effect of denoising on light probe optimization in Figure 5. Figure 6 illustrates the effect of denoising when jointly optimizing geometry, material and lighting.

3 Visibility Gradients from Shadows

In Figure 7 we show the impact of the visibility gradients [9, 2] for the shadow test on a few targeted reconstruction examples. In this ablation, we disabled denoising, and applied a full shadow term directly (instead of linearly ramping up the shadow contribution, as discussed in Section 1). While clearly beneficial in the simpler examples, e.g., single-view reconstruction and targeted optimization (finding the light position), for multi-view reconstruction with joint shape, material, and environment light optimization, the benefits of the shadow visibility gradients is less clear. In our experiments, gradients from the shadow rays in the evaluation of direct lighting in the hemisphere, typically have negligible impact compared to gradients from primary visibility. Below, we report the view-interpolation scores over the validation sets for three scenes.

	PSNR↑			SSIM↑		
	Chair	Lego	Porsche	Chair	Lego	Porsche
w/ gradients	29.15	27.03	27.83	0.948	0.918	0.941
w/o gradients	29.06	26.83	28.21	0.948	0.918	0.945

In contrast, we observed that the shadow ray gradients sometimes increase noise levels and degrade reconstruction quality. We show a visual example in Figure 8.

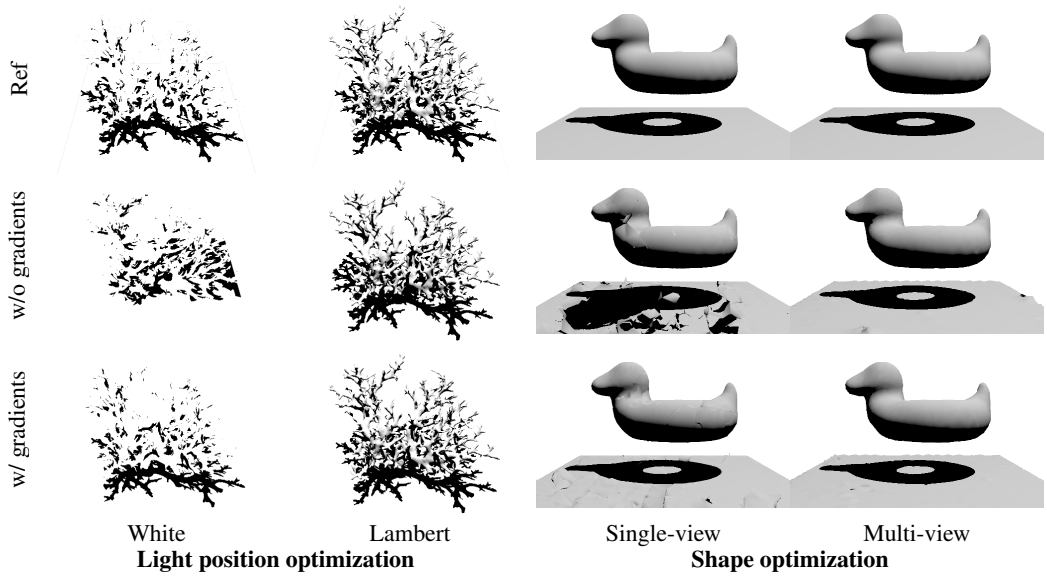


Figure 7: We show a few examples illustrating the benefits of shadow visibility gradients for inverse rendering setups. *White*: we optimize the position of a point light source, using a pure white material. Here, shadow gradients are necessary for the optimization to succeed. *Lambert*: we redo the same experiment, but with a Lambertian material including an $\mathbf{n} \cdot \mathbf{l}$ shading term. Now, both the shadow and shading terms provide gradients to the light position, and the optimization succeeds even without shadow gradients. *Single-view*: we optimize shape from a single view (purposely chosen to hide the hole in the model), using a known point light source. Here, the shadow gradients provide useful information, and improves the reconstruction to capture the hole in the model. *Multi-view*: the same setup, but optimized using multiple views. Here, there is no clear benefit of using shadow gradients (the primary visibility gradients from each view dominate the geometry reconstruction).

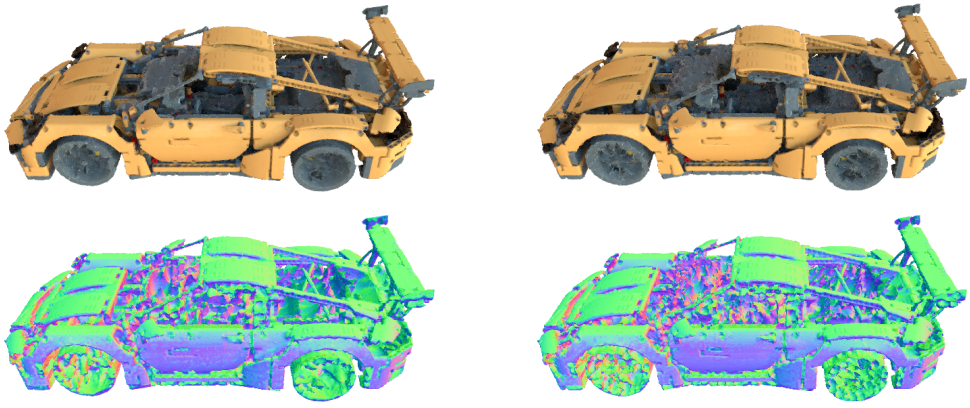
4 Sampling

In Figure 9, we examine the effect of importance sampling strategies on reconstruction quality. We compare cosine sampling, BSDF importance sampling [6], light probe importance sampling (using piecewise-constant 2D distribution sampling [12]), and multiple importance sampling [17] (MIS) using the balance heuristic. MIS consistently generates high quality reconstruction for a wide range of materials.

For unbiased results, the forward and backward passes of the renderer should use independent (*decorrelated*) random samples. However, we observe drastically reduced noise levels, hence improved convergence rates, when using *correlated* samples, i.e., re-using the same random seeds in the forward and backward pass, especially for low sample count optimization runs. This effect mainly stems from gradients flowing back to the exact same set of parameters which contributed to the forward rendering, instead of randomly selecting another set of parameters, which naturally increases variance. In Figure 10, we compare the two approaches over a range of sample counts.

5 Geometry

In Table 1 we relate our method to Table 8 (supplemental) of the NVDIFFREC paper [11], and note similar geometric quality as NVDIFFREC. The Lego scene is an outlier caused by our normal smoothness regularizer from Eq. 3. The scene is particularly challenging for geometric smoothing since it contains very complex geometry full of holes and sharp edges. Please refer to Figure 11 for a visual comparison.



PSNR: 28.21 dB, SSIM: 0.945
w/o shadow gradients

PSNR: 27.82 dB, SSIM: 0.941
w/ shadow gradients

Figure 8: Comparing the influence of shadow visibility gradients in our full pipeline with joint optimization of shape, materials, and lighting. We observe slightly higher noise levels on the geometry reconstructions with shadow visibility gradients enabled, as can be seen in the visualization of surface normals. The error metrics are average view interpolation scores over 200 novel views.

	 Diffuse	 Plastic	 Metallic
Cosine sampling	46.81	34.88	25.61
BSDF sampling	46.81	34.78	31.66
Light sampling	42.70	34.23	24.38
MIS	46.50	35.40	35.82

Figure 9: Comparison of different imaportance sampling strategies, using a diffuse, plastic and metal version of the Spot model. Lighting and materials are optimized, and geometry is kept fixed. The hemisphere is sampled using 32 spp, and denoising is disabled for this test. Tabulated average validation PSNR of the forward renderings using high sample count and no denoising over 200 frames.

6 Relighting and View Interpolation

We present relighting results with per scene breakdown for all synthetic datasets in Table 2. Figures 12, 13, and 14 show visual examples of the NeRFactor synthetic dataset. This dataset is intentionally designed to simplify light and material separation (low frequency lighting, no prominent shadowing, direct lighting only). Figures 15, 16, and 17 show visual examples from the NeRF synthetic dataset. This dataset was designed view-interpolation, and contains particularly hard scenes with all-frequency lighting. Table 3 presents per-scene view interpolation scores for the synthetic datasets.

Table 1: Chamfer L1 scores on the extracted meshes. Lower score is better.

	Chair	Hotdog	Lego	Materials	Mic
PhySG	0.1341	0.2420	0.2592	N/A	0.2712
NeRF (w/o mask)	0.0185	4.6010	0.0184	0.0057	0.0124
NeRF (w/ mask)	0.0435	0.0436	0.0201	0.0082	0.0122
NVDIFFREC	0.0574	0.0272	0.0267	0.0180	0.0098
Our	0.0566	0.0297	0.0583	0.0162	0.0151

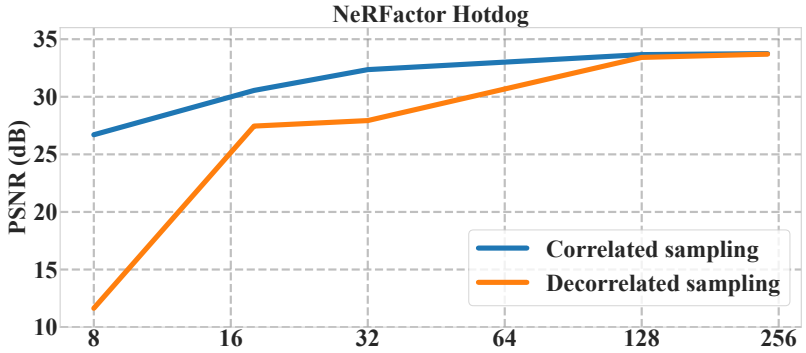


Figure 10: Correlated vs. decorrelated samples during gradient backpropagation. Although clearly *biased*, convergence at lower sample counts improves drastically when explicitly re-using samples in the backward pass (correlated), as gradients are propagated to the exact same set of parameters which contributed to the forward rendering.

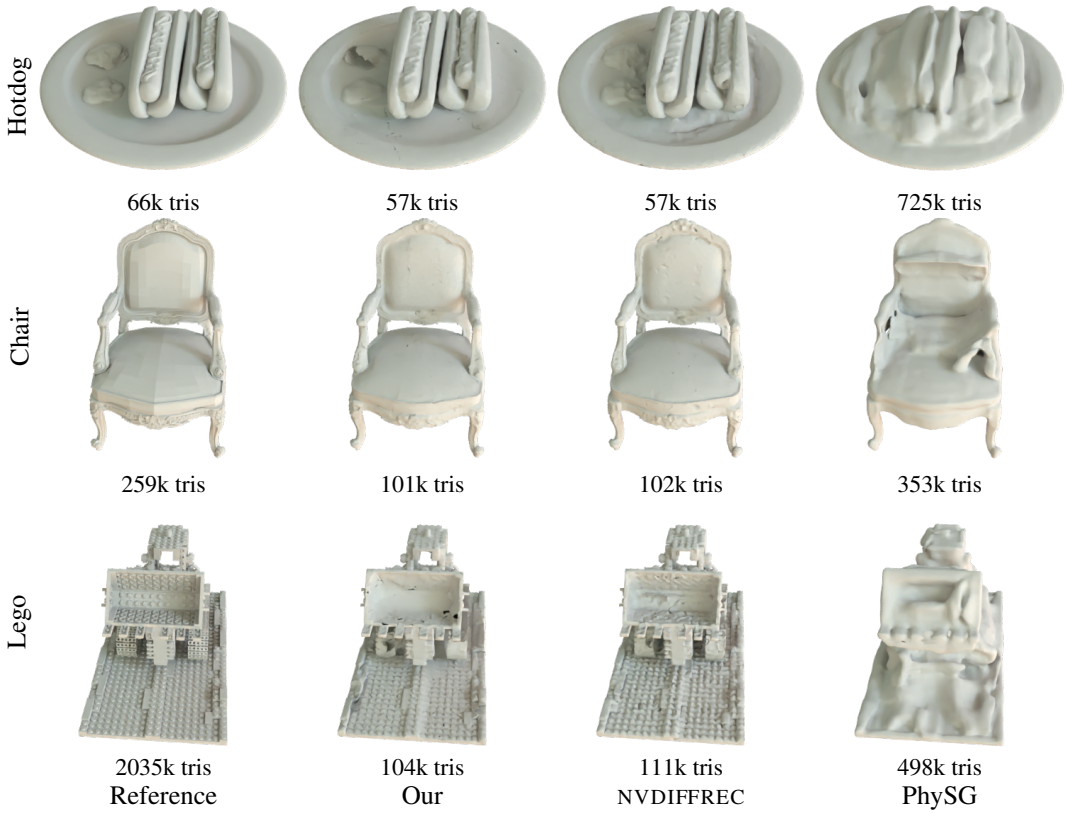


Figure 11: Extracted mesh quality visualization examples using the synthetic NeRF `Chair`, `Hotdog`, and `Lego` datasets. For these visualizations, the normal mapping shading term which simulates small geometric detail (used by Our and NVDIFFREC) is disabled.

Table 2: Relighting results for all synthetic datasets with per-scene quality metrics. The synthetic NeRFactor and NeRF train sets use 100 images from random viewpoints, and our dataset use 200 images for `Apollo` and 256 images for `Porsche` and `Roller`. The test sets used to compute relighting quality use eight novel views. Each view is relit by eight low resolution (32×16 pixel) light probes for NeRFactor, and four high resolution (2048×1024 pixels) light probes for NeRF and our dataset.

NeRFactor synthetic dataset

	PSNR↑				SSIM↑				LPIPS↓			
	Drums	Ficus	Hotdog	Lego	Drums	Ficus	Hotdog	Lego	Drums	Ficus	Hotdog	Lego
Our	23.6	29.3	30.0	21.2	0.926	0.968	0.952	0.853	0.062	0.025	0.040	0.113
NVDIFFREC	22.4	27.8	29.5	19.4	0.915	0.962	0.936	0.825	0.066	0.028	0.048	0.109
NERFACTOR	21.6	21.6	25.5	20.3	0.911	0.921	0.916	0.839	0.065	0.075	0.087	0.121

NeRF synthetic dataset

	PSNR↑					SSIM↑					LPIPS↓				
	Chair	Hotd.	Lego	Mats.	Mic	Chair	Hotd.	Lego	Mats.	Mic	Chair	Hotd.	Lego	Mats.	Mic
Our	27.0	25.9	23.4	25.1	30.9	0.946	0.929	0.890	0.923	0.972	0.044	0.073	0.088	0.047	0.025
NVDIFFREC	25.1	20.3	21.4	19.9	29.7	0.921	0.853	0.861	0.847	0.967	0.060	0.126	0.089	0.079	0.026

Our synthetic dataset

	PSNR↑			SSIM↑			LPIPS↓		
	Apollo	Porsche	Roller	Apollo	Porsche	Roller	Apollo	Porsche	Roller
Our	25.6	28.2	27.5	0.931	0.965	0.952	0.045	0.030	0.027
NVDIFFREC	24.7	26.1	20.4	0.926	0.952	0.898	0.049	0.029	0.067

Table 3: View interpolation results for all synthetic datasets with per-scene quality metrics. The synthetic NeRFactor and NeRF training sets use 100 images from random viewpoints, and our dataset use 200 images for `Apollo` and 256 images for `Porsche` and `Roller`. The quality metric for a scene is the arithmetic average over all 200 test images.

NeRFactor synthetic dataset

	PSNR↑				SSIM↑			
	Drums	Ficus	Hotdog	Lego	Drums	Ficus	Hotdog	Lego
Our	26.6	30.8	33.9	29.1	0.940	0.975	0.967	0.930
NVDIFFREC	28.5	31.2	36.3	30.7	0.959	0.978	0.981	0.951
NERFACTOR	24.6	23.1	31.6	28.1	0.933	0.937	0.948	0.900

NeRF synthetic dataset

	PSNR↑					SSIM↑				
	Chair	Hotdog	Lego	Mats.	Mic	Chair	Hotdog	Lego	Mats.	Mic
Our	29.0	30.4	26.8	25.7	29.0	0.947	0.942	0.917	0.913	0.963
NVDIFFREC	31.6	33.0	29.1	26.7	30.8	0.969	0.973	0.949	0.923	0.977

Our synthetic dataset

	PSNR↑			SSIM↑		
	Apollo	Porsche	Roller	Apollo	Porsche	Roller
Our	23.4	28.4	25.0	0.908	0.953	0.941
NVDIFFREC	23.9	28.5	25.1	0.923	0.958	0.951

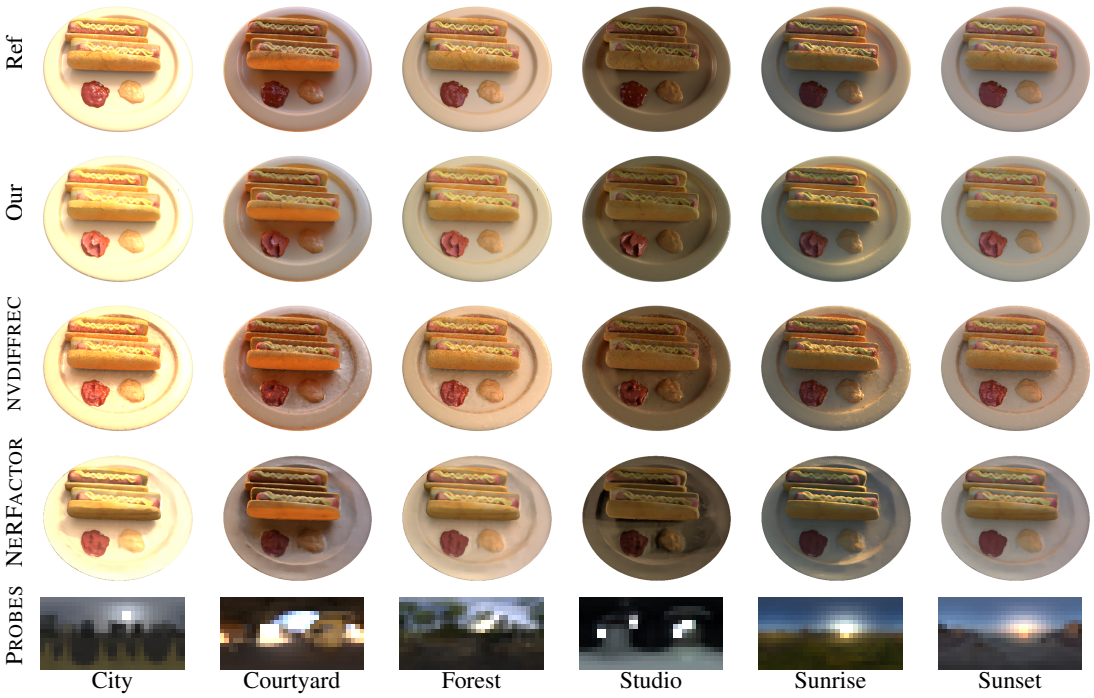


Figure 12: Relighting examples from `Hotdog` scene of the NeRFactor synthetic dataset. We show a single view relit by six of the eight different low frequency light probes from the test set.

7 Scene Credits

Mori Knob from Yasotoshi Mori (CC BY-3.0). Bob and Spot models (CC0) by Keenan Crane. Rollercoaster and Porsche scenes from LDraw resources (CC BY-2.0) by Philippe Hurbain. Apollo and Damicornis models courtesy of the Smithsonian 3D repository [15] (CC0). The Family scene is part of the Tanks&Temples dataset [8] (CC BY-NC-SA 3.0), the Character scene is part of the BlendedMVS dataset [18] (CC BY-4.0) and the Gold Cape scene is part of the NeRD dataset [3] (CC BY-NC-SA 4.0). The NeRF [10] and NeRFactor [20] datasets (CC BY-3.0) contain renders from modified blender models located on blendswap.com: chair by 1DInc (CC0), drums by bryanajones (CC-BY), ficus by Herberhold (CC0), hotdog by erickfree (CC0), lego by Heinzelnisse (CC-BY-NC), materials by elbrujodelatribu (CC0), mic by up3d.de (CC0), ship by gregzaal (CC-BY-SA). Light probes from Poly Haven [19]: Aerodynamics workshop and Boiler room by Oliksiy Yakovlyev, Dreifaltkeitsberg by Andreas Mischok, and Music hall by Sergej Majboroda (all CC0). The probes provided in the NeRFactor dataset are modified from the probes (CC0) shipped with Blender.

References

[1] Attila T. Áfra. Open Image Denoise, 2022. `https://www.openimagedenoise.org/`.

[2] Sai Bangaru, Tzu-Mao Li, and Frédo Durand. Unbiased Warped-Area Sampling for Differentiable Rendering. *ACM Trans. Graph.*, 39(6):245:1–245:18, 2020.

[3] Mark Boss, Raphael Braun, Varun Jampani, Jonathan T. Barron, Ce Liu, and Hendrik P.A. Lensch. NeRD: Neural Reflectance Decomposition from Image Collections. In *International Conference on Computer Vision (ICCV)*, pages 5855–5864, 2021.

[4] Jon Hasselgren, Jacob Munkberg, Jaakko Lehtinen, Miika Aittala, and Samuli Laine. Appearance-driven automatic 3d model simplification. In *Eurographics Symposium on Rendering*, 2021.

[5] Jon Hasselgren, Jacob Munkberg, Anjul Patney, Marco Salvi, and Aaron Lefohn. Neural Temporal Adaptive Sampling and Denoising. *Computer Graphics Forum*, 39(2):147–155, 2020.

[6] Eric Heitz. Sampling the GGX Distribution of Visible Normals. *Journal of Computer Graphics Techniques (JCGT)*, 7(4):1–13, 2018.

[7] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. In *Proceedings of the 3rd International Conference for Learning Representations*, 2015.

Figure 13: Relighting examples from `Ficus` scene of the NeRFactor synthetic dataset. We show a single view relit by six of the eight different low frequency light probes of the test set.

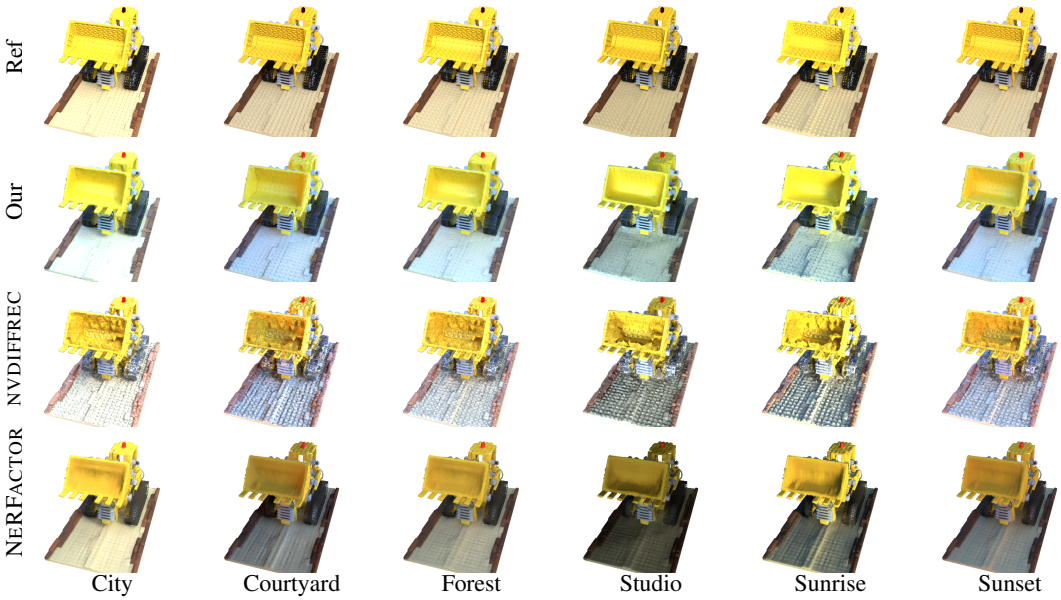


Figure 14: Relighting examples from `Lego` scene of the NeRFactor synthetic dataset. We show a single view relit by six of the eight different low frequency light probes of the test set.

[8] Arno Knapitsch, Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. Tanks and temples: Benchmarking large-scale scene reconstruction. *ACM Trans. Graph.*, 36(4):78:1–78:13, 2017.

[9] Guillaume Loubet, Nicolas Holzschuch, and Wenzel Jakob. Reparameterizing Discontinuous Integrands for Differentiable Rendering. *ACM Trans. Graph.*, 38(6):228:1–228:14, 2019.

[10] 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.

[11] Jacob Munkberg, Jon Hasselgren, Tianchang Shen, Jun Gao, Wenzheng Chen, Alex Evans, Thomas Mueller, and Sanja Fidler. Extracting Triangular 3D Models, Materials, and Lighting From Images. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8270–8280, 2022.

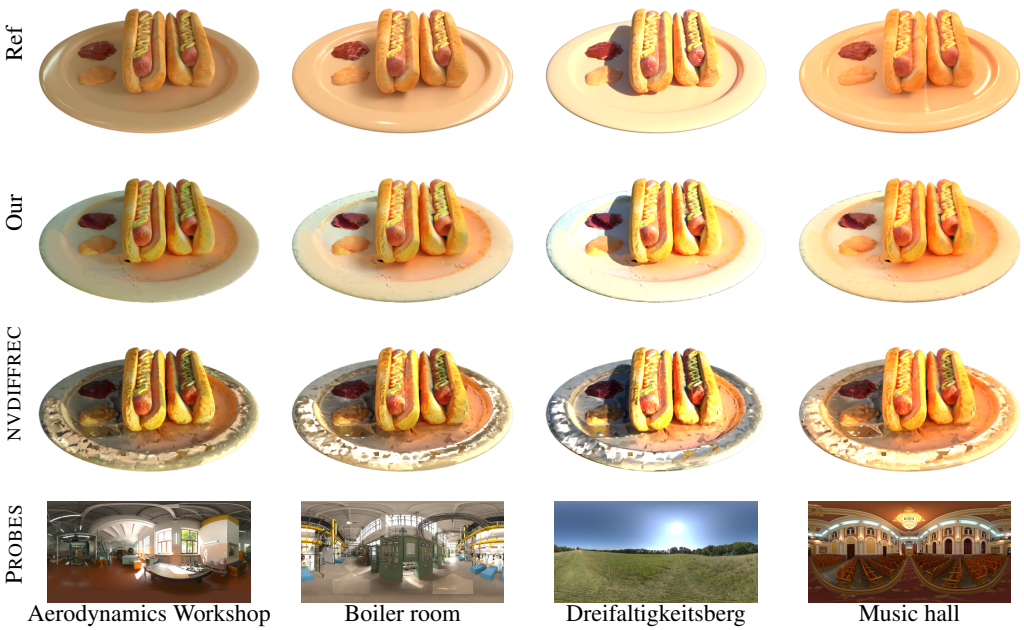


Figure 15: Relighting examples from `Hotdog` scene of the NeRF synthetic dataset. We show a single view relit by the four different high frequency light probes from Poly Haven (CC0).

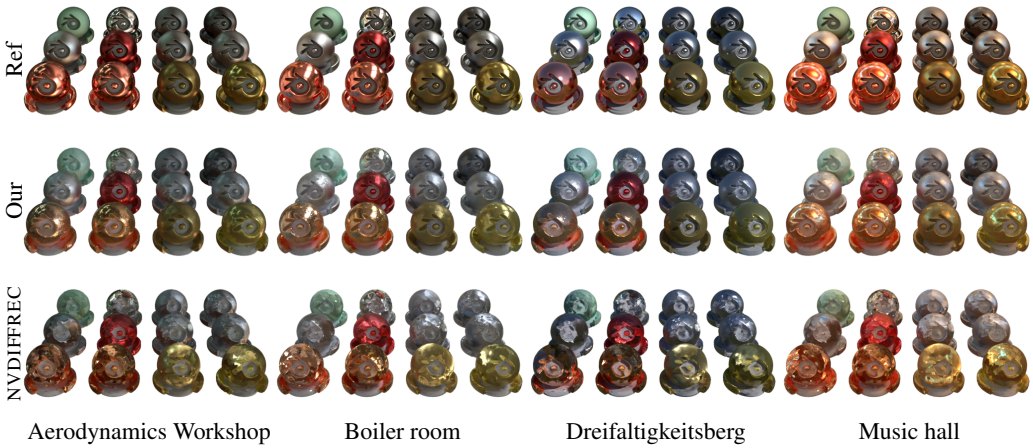


Figure 16: Relighting examples from `Materials` scene of the NeRF synthetic dataset. We show a single view relit by the four different high frequency light probes.

[12] Matt Pharr and Greg Humphreys. *Physically Based Rendering, Second Edition: From Theory To Implementation*. Morgan Kaufmann Publishers Inc., 2nd edition, 2010.

[13] Christoph Schied, Anton Kaplanyan, Chris Wyman, Anjul Patney, Chakravarty R. Alla Chaitanya, John Burgess, Shiqiu Liu, Carsten Dachsbacher, Aaron Lefohn, and Marco Salvi. Spatiotemporal Variance-guided Filtering: Real-time Reconstruction for Path-traced Global Illumination. In *Proceedings of High Performance Graphics*, pages 2:1–2:12, 2017.

[14] Tianchang Shen, Jun Gao, Kangxue Yin, Ming-Yu Liu, and Sanja Fidler. Deep Marching Tetrahedra: a Hybrid Representation for High-Resolution 3D Shape Synthesis. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 6087–6101, 2021.

[15] Smithsonian. Smithsonian 3D Digitization, 2018. https://3d.si.edu/.

[16] Michael Stokes, Matthew Anderson, Srinivasan Chandrasekar, and Ricardo Motta. A Standard Default Color Space for the Internet - sRGB, 1996.

[17] Eric Veach and Leonidas J. Guibas. Optimally Combining Sampling Techniques for Monte Carlo Rendering. In *Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques*, page

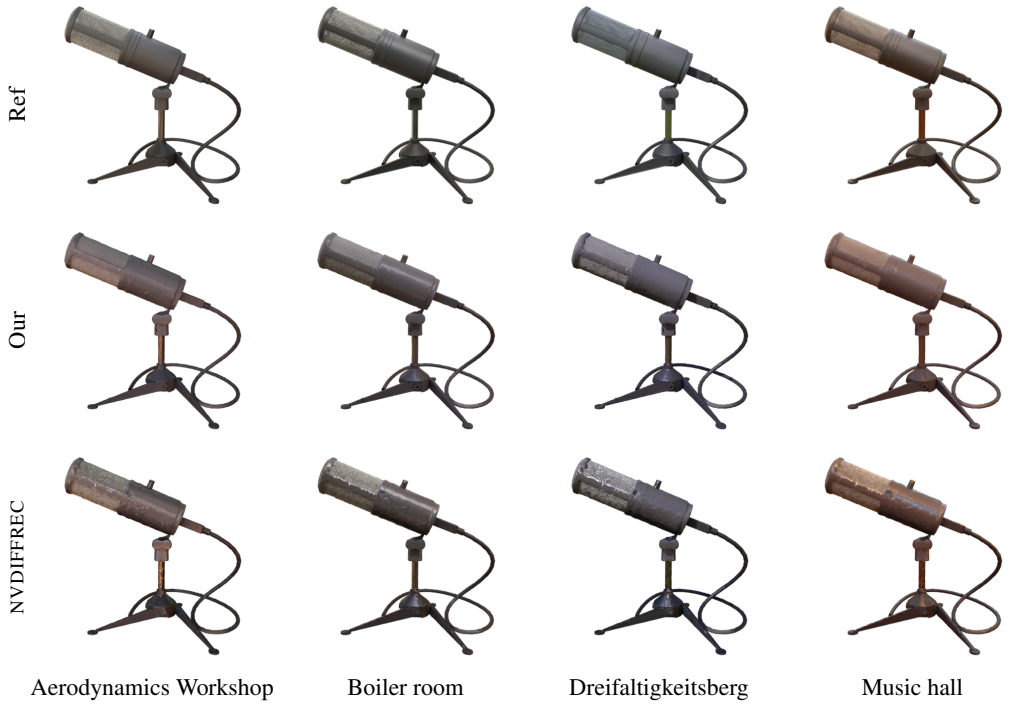


Figure 17: Relighting examples from `Microphone` scene of the NeRF synthetic dataset. We show a single view relit by the four different high frequency light probes.

419–428, 1995.

[18] Yao Yao, Zixin Luo, Shiwei Li, Jingyang Zhang, Yufan Ren, Lei Zhou, Tian Fang, and Long Quan. BlendedMVS: A Large-scale Dataset for Generalized Multi-view Stereo Networks. *Computer Vision and Pattern Recognition (CVPR)*, 2020.

[19] Greg Zaal. polyhaven, 2021. https://polyhaven.com/hdris.

[20] Xiuming Zhang, Pratul P. Srinivasan, Boyang Deng, Paul Debevec, William T. Freeman, and Jonathan T. Barron. NeRFactor: Neural Factorization of Shape and Reflectance under an Unknown Illumination. *ACM Trans. Graph.*, 40(6):237:1–237:18, 2021.

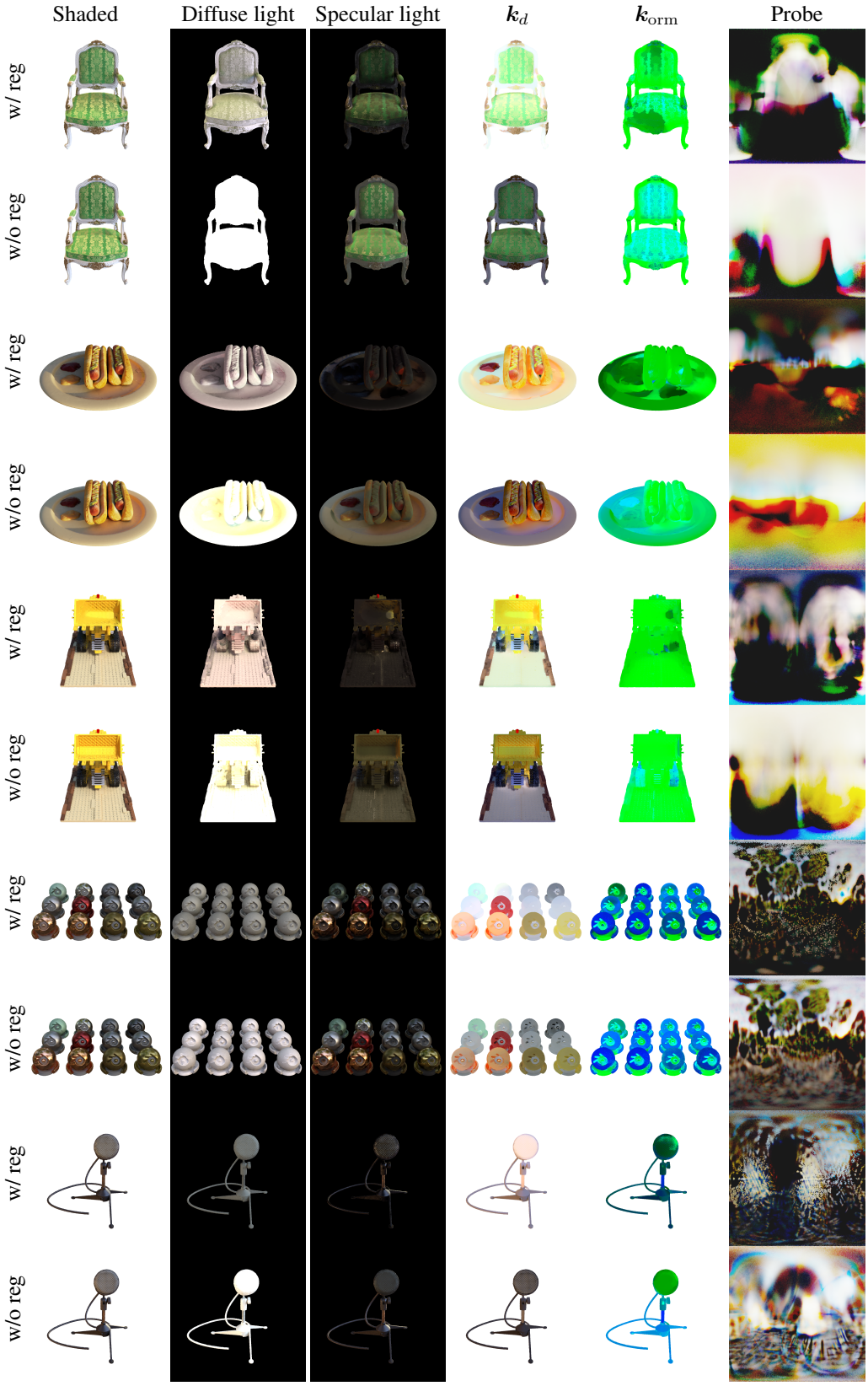


Figure 18: Breakdown of shading terms with and without our light regularizer. Note that, for complex meshes (Hotdog and Lego), shadows are more accurately baked into lighting terms, effectively delighting the $\boldsymbol{k}_d$ textures. For scenes with simple visibility (Chair, Materials, Microphone), the regularizer arguably contrast enhance the light probes too much, but we still note smoother $\boldsymbol{k}_d$.