Lighthouse: Predicting Lighting Volumes for Spatially-Coherent Illumination Supplementary Materials

Pratul P. Srinivasan*[1] Ben Mildenhall*[1] Matthew Tancik[1]
Jonathan T. Barron[2] Richard Tucker[2] Noah Snavely[2]
[1]UC Berkeley, [2]Google Research

The following sections include details about our included supplementary video, additional implementation details for our method, additional qualitative results, and implementation details for baseline methods.

1. Supplementary Video

We encourage readers to view our included supplementary video for a brief overview of our method and qualitative comparisons between our method and baselines that showcase our method's spatial coherence by inserting specular virtual objects that move along smooth paths.

2. Multiscale Lighting Volume Details

Our multiscale lighting volume consists of 5 scales of $64 \times 64 \times 64$ RGBα volumes. As illustrated in Figure 3 in the main paper, each scale's volume has half the side length of the previous scale's volume.

3. Illumination Rendering Details

For training, we implement the volume rendering procedure described in the main paper by sampling each volume in our multiscale lighting representation on a set of concentric spheres around the target environment map location with trilinear interpolation. We use 128 spheres per scale, sampled evenly in radius between the closest and furthest voxels from the target location. Then, we alpha-composite these spheres ($128 \times 5 = 640$ total spheres) from outermost to innermost to render the environment map at that location.

For fast relighting performance at test time, we implement the volume rendering as standard ray tracing using CUDA. We intersect each camera ray with each virtual object, and then trace rays through the volume from that intersection location to compute the incident illumination to correctly shade that pixel. This means that the illumination we use for relighting virtual objects varies spatially both between different objects as well as across the geometry of each object. This effect is quite difficult to simulate with prior lighting estimation work such as Neural Illumination [7], since this would require running a deep network tens of thousands of times to predict an environment map for each camera ray intersecting the virtual object. In contrast, our method only requires one pass of network inference to predict a multiscale volumetric lighting estimation, and full spatially-varying lighting across inserted objects is then handled by ray tracing through our volumes.

Figure 1 illustrates how our spatially-varying lighting across the surfaces of inserted objects adds realism. In the top image, we render all points on each object's surface using a single environment map predicted by our method at the object's centroid. In the bottom image, we render each point by ray tracing through our predicted lighting volume, so each point is effectively illuminated by a different environment map. We can see that the inserted virtual objects have a more realistic appearance in the bottom image. For example, the armadillo's legs and the Buddha statue's base correctly reflect the floor while the Buddha statue's face correctly reflects the windows instead of the black counter.

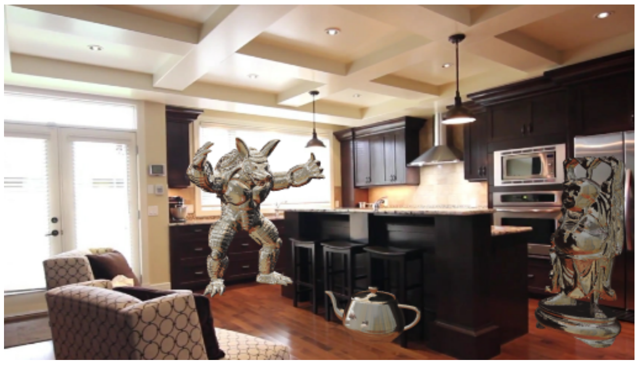
4. Network Architectures

MPI prediction network (Table 1) Our MPI prediction network is a 3D encoder-decoder CNN with skip connections. We use residual blocks [3] with layer normalization [1] for all layers, strided convolutions for downsampling within the encoder, and nearest-neighbor upsampling within the decoder. This network outputs a 3D array of RGBα values and a scalar 3D array of blending weights between 0 and 1.

We compute a "background" image as the average RGB over all depth planes in the output array, and use the "background + blending weights" MPI parameterization [8], where each MPI plane's RGB colors are defined as a convex combination of the input reference image and the predicted "background" image, using the predicted blending weights.

Volume completion network (Table 1) We use the same 3D encoder-decoder CNN architecture detailed above for

* Authors contributed equally to this work.


(a) Single environment map per object

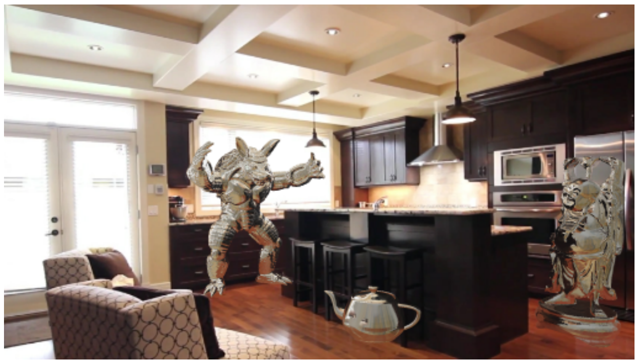

(b) Fully spatially varying lighting

Figure 1: Visualization of our method's ability to realistically render spatially-varying lighting across each object's surface. In (a), we use a single environment map predicted by our method at the object's centroid to illuminate the entire object. In (b), we illuminate each point on each object's surface by tracing rays through our predicted volume, so each point is effectively illuminated by a different environment map. This results in more realistic lighting effects, as can be seen in the reflection of the floor in the armadillo's legs and the Buddha's statue's base and the reflection of the window in the Buddha statue's face. This difference is less pronounced in smaller objects such as the teapot.

all 5 scales of our volume completion network, with separate weights per scale. At each scale, the network predicts an RGBα volume as well as a scalar volume of blending weights between 0 and 1. We parameterize each scale's output as a convex combination of the input resampled volume at that scale and the network's output RGBα volume, using the network's predicted blending weights.

Discriminator network (Table 2) We use a 2D CNN PatchGAN [4] architecture with spectral normalization [6].

	Encoder	
1	$3 \times 3 \times 3$ conv, 8 features	$H \times W \times D \times 8$
2	$3 \times 3 \times 3$ conv, 16 features, stride 2	$H/2 \times W/2 \times D/2 \times 16$
3-4	($3 \times 3 \times 3$ conv, 16 features) $\times 2$, residual	$H/2 \times W/2 \times D/2 \times 16$
5	$3 \times 3 \times 3$ conv, 32 features, stride 2	$H/4 \times W/4 \times D/4 \times 32$
6-7	($3 \times 3 \times 3$ conv, 32 features) $\times 2$, residual	$H/4 \times W/4 \times D/4 \times 32$
8	$3 \times 3 \times 3$ conv, 64 features, stride 2	$H/8 \times W/8 \times D/8 \times 64$
9-10	($3 \times 3 \times 3$ conv, 32 features) $\times 2$, residual	$H/8 \times W/8 \times D/8 \times 64$
11	$3 \times 3 \times 3$ conv, 128 features, stride 2	$H/16 \times W/16 \times D/16 \times 128$
12-13	($3 \times 3 \times 3$ conv, 32 features) $\times 2$, residual	$H/16 \times W/16 \times D/16 \times 128$
	Decoder	
14	$2\times$ nearest neighbor upsample	$H/8 \times W/8 \times D/8 \times 128$
15	concatenate 14 and 10	$H/8 \times W/8 \times D/8 \times (128 + 64)$
16	$3 \times 3 \times 3$ conv, 64 features	$H/8 \times W/8 \times D/8 \times 64$
17-18	($3 \times 3 \times 3$ conv, 64 features) $\times 2$, residual	$H/8 \times W/8 \times D/8 \times 64$
19	$2\times$ nearest neighbor upsample	$H/4 \times W/4 \times D/4 \times 64$
20	concatenate 19 and 7	$H/4 \times W/4 \times D/4 \times (64 + 32)$
21	$3 \times 3 \times 3$ conv, 32 features	$H/4 \times W/4 \times D/4 \times 32$
22-23	($3 \times 3 \times 3$ conv, 32 features) $\times 2$, residual	$H/4 \times W/4 \times D/4 \times 32$
24	$2\times$ nearest neighbor upsample	$H/2 \times W/2 \times D/2 \times 32$
25	concatenate 24 and 4	$H/2 \times W/2 \times D/2 \times (32 + 16)$
26	$3 \times 3 \times 3$ conv, 16 features	$H/2 \times W/2 \times D/2 \times 16$
27-28	($3 \times 3 \times 3$ conv, 16 features) $\times 2$, residual	$H/2 \times W/2 \times D/2 \times 16$
29	$2\times$ nearest neighbor upsample	$H \times W \times D \times 16$
30	concatenate 29 and 1	$H \times W \times D \times (32 + 16)$
26	$3 \times 3 \times 3$ conv, 16 features	$H \times W \times D \times 16$
27	$3 \times 3 \times 3$ conv, 5 features (sigmoid)	$H \times W \times D \times 5$

Table 1: **3D CNN network architecture used for MPI prediction and volume completion networks.** All convolutional layers use a ReLu activation, except for the final layer which uses a sigmoid activation.

	Encoder	
1	4×4 conv, 64 features, stride 2	$H/2 \times W/2 \times D/2 \times 64$
2	4×4 conv, 128 features, stride 2	$H/4 \times W/4 \times D/4 \times 128$
3	4×4 conv, 256 features, stride 2	$H/8 \times W/8 \times D/8 \times 256$
4	4×4 conv, 512 features, stride 2	$H/16 \times W/16 \times D/16 \times 512$
5	4×4 conv, 1 feature	$H/16 \times W/16 \times D/16 \times 1$

Table 2: **2D CNN discriminator network architecture.** All convolutional layers use a Leaky ReLu activation with $\alpha = 0.2$, except for the final layer.

5. Baseline Method Details

DeepLight [5] and Garon *et al.* [2] output HDR environment maps with unknown scales, since camera exposure is a free parameter. For fair comparisons, we scale their environment maps so that their average radiance matches the average radiance of the ground-truth environment maps for our quantitative and qualitative results when using the InteriorNet dataset. There is no ground-truth environment map for our real photograph results, so we scale their predicted environment map so that their average radiance matches the average radiance of the reference image.

Neural Illumination [7] does not have an available implementation, so we implement and train a generous baseline version of their method. Their published method trains a 2D CNN network to predict per-pixel geometry from a single input image, uses this geometry to warp input image pixels into the target environment map, trains another 2D CNN to complete the unobserved areas of this environment map, and trains a final 2D CNN to convert this en-

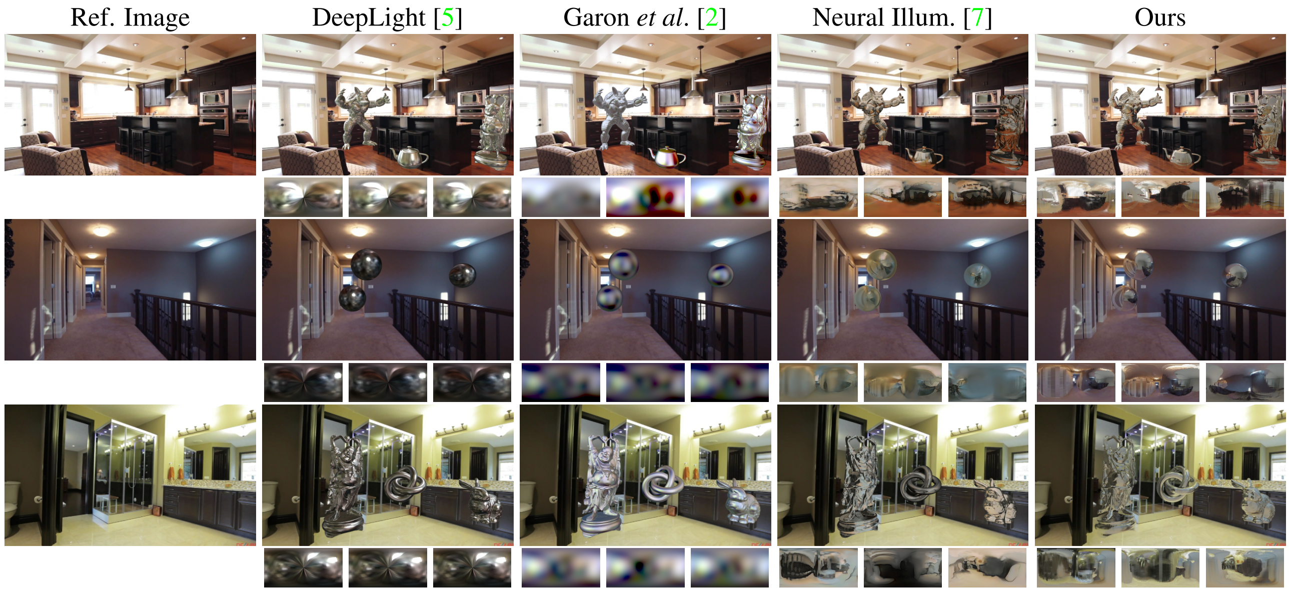


Figure 2: Qualitative comparison of real images from the RealEstate10K dataset [8] with relit inserted virtual objects and corresponding environment maps. DeepLight [5] only estimates a single environment map for the entire scene, so virtual objects at different locations do not realistically reflect scene content. Garon *et al*. [2] estimate a low-dimensional lighting representation at each pixel, so their lighting does not realistically vary across 3D locations along the same camera ray. Furthermore, their low-dimensional representation does not contain sufficient high frequency detail for rendering specular objects, so inserted objects have a more diffuse appearance. Neural Illumination [7] has trouble correctly preserving scene content colors from the input image. Additionally, their method separately predicts unobserved content for the environment map at each 3D location so their predicted lighting is not as spatially-coherent. Our results contain more plausible spatially-coherent reflections, and we can see that the colors of virtual objects in our results are more consistent with the scene content in the original image.

vironment map to HDR. To enable a generous comparison with our method, which uses a stereo pair of images as input, we remove the first CNN from the Neural Illumination method, and instead use the ground-truth depth to reproject input image pixels into the target environment map for all quantitative and qualitative comparisons on our InteriorNet test set (we use our method's estimated MPI geometry for qualitative results on the RealEstate dataset where no ground truth is available). Additionally, since we assume InteriorNet renderings are captured with a known invertible tone-mapping function, we omit Neural Illumination's LDR to HDR conversion network and instead just apply the exact inverse tone-mapping function to their completed LDR environment maps. For a fair comparison with our method's results, we use the same architecture as Table 1 for the environment map completion network, but with 2D kernels instead of 3D, and the same discriminator as Table 2. We train this generous Neural Illumination baseline on the same InteriorNet training dataset that we use to train our method.

6. Additional Results

Figure 2 contains additional qualitative results for specular virtual objects relit with baseline methods and our algorithm. We can see that our results are more spatially-coherent and contain realistic reflections of scene content.

References

[1] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization. *arXiv:1607.06450*, 2016. 1

[2] Mathieu Garon, Kalyan Sunkavalli, Sunil Hadap, Nathan Carr, and Jean-Francois Lalonde. Fast spatially-varying indoor lighting estimation. *CVPR*, 2019. 2, 3

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning. *CVPR*, 2016. 1

[4] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. Image-to-image translation with conditional adversarial networks. *CVPR*, 2017. 2

[5] Chloe LeGendre, Wan-Chun Ma, Graham Fyffe, John Flynn, Laurent Charbonnel, Jay Busch, and Paul Debevec. Deeplight: Learning illumination for unconstrained mobile mixed reality. *CVPR*, 2019. 2, 3

[6] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. *ICLR*, 2018. 2

[7] Shuran Song and Thomas Funkhouser. Neural illumination: Lighting prediction for indoor environments. *CVPR*, 2019. 1, 2, 3

[8] Tinghui Zhou, Richard Tucker, John Flynn, Graham Fyffe, and Noah Snavely. Stereo magnification: Learning view synthesis using multiplane images. *SIGGRAPH*, 2018. 1, 3