# Light Field Networks: Neural Scene Representations with Single-Evaluation Rendering

# –Supplementary Material–

**Vincent Sitzmann**[1,*] sitzmann@mit.edu
**Semon Rezchikov**[2,*] skr@math.columbia.edu
**William T. Freeman**[1,3] billf@mit.edu
**Joshua B. Tenenbaum**[1,4,5] jbt@mit.edu
**Frédo Durand**[1] fredo@mit.edu
[1]MIT CSAIL [2]Columbia University [3] IAFI [4]MIT BCS [5] CBMM
vsitzmann.github.io/lfns/

## Contents



[*]These authors contributed equally to this work.

35th Conference on Neural Information Processing Systems (NeurIPS 2021), Sydney, Australia.

# 1 Additional results on the local two-plane parameterization

In this section, we derive Proposition 1 of the paper. We recall here that the Plucker coordinates parameterize the space $\mathcal{L}$ of oriented lines in $\mathbb{R}^3$ as the set of tuples

$$\mathcal{L} = \{(\mathbf{d}, \mathbf{w}) \mid \mathbf{d}, \mathbf{w} \in \mathbb{R}^3;\ \mathbf{d} \cdot \mathbf{w} = 0, \|\mathbf{d}\| = 1\}. \tag{1}$$

The line through $\mathbf{x}$ in direction $\mathbf{d}$ (with $\|\mathbf{d}\| = 1$), i.e. the line $\overline{\mathbf{x}, \mathbf{x}+\mathbf{d}}$, has normalized Plücker coordinates

$$(\mathbf{d}, \mathbf{x} \times (\mathbf{x} + \mathbf{d})) = (\mathbf{d}, \mathbf{x} \times \mathbf{d}).$$

If the direction vector $\mathbf{d}$ is not normalized, we can still compute Plücker coordinates as above; to normalize the coordinates we must apply the normalization operator

$$N(\mathbf{d}, \mathbf{w}) = (\mathbf{d}, \mathbf{w})/\|\mathbf{d}\|. \tag{2}$$

Thus the ray $\overrightarrow{\mathbf{a}(s)\mathbf{b}(t)}$ has (unnnormalized) Plücker coordinates

$$v(s,t) = (\mathbf{b} - \mathbf{a}, \mathbf{a} \times \mathbf{b}). \tag{3}$$

*Proof of Proposition 1.* Given two coplanar lines

$$\mathbf{a}(s) = \mathbf{x} + s\mathbf{d} \text{ and } \mathbf{b}(t) = \mathbf{x}' + t\mathbf{d} \tag{4}$$

we write

$$\mathbf{c}(s,t) = \Phi\left(\frac{\mathbf{b}-\mathbf{a}}{\|\mathbf{b}-\mathbf{a}\|}, \frac{\mathbf{a}\times\mathbf{b}}{\|\mathbf{b}-\mathbf{a}\|}\right) = \Phi(N(v(s,t)) \tag{5}$$

for the color of the ray $\ell = \overrightarrow{\mathbf{a}(s)\mathbf{b}(t)}$ described by the LFN $\Phi$. Suppose $\ell$ captures the color of a point $\mathbf{p}$ in a 3D scene. For a Lambertian scene, if the ray is not a tangent ray, $\mathbf{c}(s,t)$ should be constant near $\ell$ along the line in the $(s,t)$ plane corresponding to the family of rays through $\mathbf{p}$. *Therefore, the gradient $\nabla_{s,t}\mathbf{c}(s,t)$ is orthogonal to this line in the $(s,t)$ plane.*

Write $J$ for the matrix of rotation by 90 degrees, i.e. $[0,-1;1,0]$. Thus at non-tangent lines, $J\nabla_{s,t}\mathbf{c}(s,t)$ will point along the family of rays through $\mathbf{p}$.

We write $D$ for the distance between the lines $\mathbf{a}$ and $\mathbf{b}$; so $D = \|\mathbf{x}' - \mathbf{x}\|$ for $\mathbf{a}, \mathbf{b}$ as in (4). Let $\mathbf{a}_0, \mathbf{b}_0$ be the closest points to $\mathbf{p}$ on $\mathbf{a}(s)$ and $\mathbf{b}(t)$, respectively. Write $\mathbf{a}_1 = \mathbf{a}(s)$ and $\mathbf{b}_1 = \mathbf{b}(t)$ for the intersection of $\ell$ with $\mathbf{a}, \mathbf{b}$; then a nearby ray on the pencil of rays through $\mathbf{p}$ is given by $\ell' = \overrightarrow{\mathbf{a}_1'\mathbf{b}_1'}$ for some $\mathbf{a}_1' = \mathbf{a}(s-\delta s), \mathbf{b}_1' = \mathbf{b}(t+\delta t)$. We have similar triangles $\triangle\mathbf{p}\mathbf{a}_0\mathbf{a}_1$, $\triangle\mathbf{p}\mathbf{a}_0\mathbf{a}_1'$, $\triangle\mathbf{p}\mathbf{b}_0\mathbf{b}_1$, and $\triangle\mathbf{p}\mathbf{b}_0\mathbf{b}_1'$. Since the $(s,t)$ coordinates of $\ell'$ differ from the $(s,t)$ coordinates of $\ell$ by a multiple of $J\nabla_{s,t}\mathbf{c}(s,t)$, the similarity relationships between the triangles above imply that

$$\frac{\delta s}{\delta t} = \frac{-(J\nabla\mathbf{c})_s}{(J\nabla\mathbf{c})_t} = \frac{d}{D-d}.$$

This simplifies to

$$\frac{\partial_t \mathbf{c}}{\partial_s \mathbf{c}} = \frac{d}{D-d}.$$

Rearranging, we have

$$d = D\frac{\partial_t \mathbf{c}}{\partial_s \mathbf{c} + \partial_t \mathbf{c}}. \tag{6}$$

□

Let $\delta$ be the distance from $\mathbf{a}(s)$ to $p$; then $\delta = \sqrt{s^2 + d^2}$.

Let's now describe how to get an estimate of a point generating the color of a ray in Plücker coordinates. Say we have (normalized) Plücker coordinates $(\mathbf{d}, \mathbf{w})$. Write $\mathbf{x} = \mathbf{d} \times \mathbf{w}$; this is the closest point on the line $\ell_{(\mathbf{d},\mathbf{w})}$ to the origin. Now choose auxiliary $\mathbf{d}'$ with $\|\mathbf{d}\| = 1$. (Ideally $\mathbf{d}$ is

not close to $\mathbf{d}'$.) Write $\mathbf{x}' = \mathbf{x} + D\mathbf{d}$ for some positive number $D$. Then we consider the two-plane parametrization through

$$\mathbf{a}(s) = \mathbf{x} + s\mathbf{d}', \mathbf{b}(t) = \mathbf{x}' + t\mathbf{d}' \tag{7}$$

Deine $\mathbf{c}$ via (5) where we define $v$ as in (3) using $\mathbf{a}, \mathbf{b}$ as in (7). The line $\overrightarrow{\mathbf{x}, \mathbf{x} + \mathbf{d}} = \ell_{(\mathbf{d},\mathbf{w})}$ intersects $\mathbf{a}(s)$ at $\mathbf{x} = \mathbf{a}(0)$ and intersects $\mathbf{b}(t)$ at $\mathbf{x}' = \mathbf{x} + D\mathbf{d} = \mathbf{b}(0)$. Compute $\partial_s \mathbf{c}, \partial_t \mathbf{c}$ — a procedure that involves computing analytical derivatives of the neural network $\Phi$ — then compute $d(s, t)$ via (6), and then

$$\mathbf{p} = \mathbf{x} + \delta(0, 0)\mathbf{d}.$$

## 2 Reproducibility

In the following, we provide all the details necessary to reproduce the experiments outlined in the paper. *All code and datasets will be made publicly available.*

### 2.1 Hardware

Each model was separately trained on a single NVIDIA RTX 6000 GPU with 24 GB of memory. Overall, we used up to 4 GPUs in parallel.

### 2.2 Architecture Details

All LFNs are implemented as six-layer fully connected neural networks with ReLU nonlinearities and 256 hidden units per layer. Before each layer, we leverage layer normalization *without affine transforms*, i.e., no additional parameters are introduced. All hypernetworks are implemented as three-layer fully connected neural networks with ReLU nonlinearities and layer normalization *with* affine transform. The hidden layer size of the hypernetworks is 256. Thus, the parameter $\ell$ in Equation 10 of the main text is $\sim 400,000$. The size of latent codes $\mathbf{z}$ is 256.

### 2.3 360-degree light field reconstruction experiments (Figure 5)

**Cars dataset.** We use the dataset proposed by Sitzmann et al. [9], hosted by the authors of pixelNeRF [11] under https://drive.google.com/drive/folders/1PsT3uKwqHHD2bEEHkIXB99AlIjtmrEiR.

**Rooms dataset.** Using the assets provided by the authors of GQN [2], hosted under https://github.com/deepmind/lab/tree/master/assets/textures/map/lab_games and a modified version of the Blender Shapenet rendering script hosted under https://github.com/vsitzmann/shapenet_renderer/blob/master/shapenet_spherical_renderer.py, we rendered $10,000$ rooms. The rooms have a sidelength of 7. Cameras are placed exclusively in the center $2 \times 2$ square of the room. Objects are placed exclusively in the outer $1.5$ perimeter of the room, such that the *camera is only placed in unobstructed free space*. Colors and types of objects are chosen at random. Every room features between $1$ and $5$ objects. We render 30 observations per room.

**Experiment Details.** We train separate models for the "cars" and "rooms" classes. We train in two resolution stages. First, at a resolution of $64 \times 64$ and a batch size of 300, then, at a batch size of 75 at a resolution of $128 \times 128$ until convergence for a total of approx. 3 days, both using the ADAM optimizer with a step size of $10^{-4}$. We choose the parameter $\lambda_{lat}$ as $1e2$. At test time, we freeze the network parameters, initialize all latent codes to the prior mean (i.e., all zeros), and optimize the latent codes until convergence. Hyperparameters were discovered via unstructured search.

### 2.4 Multi-class single shot reconstruction

**Dataset.** We use the dataset proposed by Kato et al. [3], available for download under https://s3.eu-central-1.amazonaws.com/avg-projects/differentiable_

volumetric_rendering/data/NMR_Dataset.zip (hosted by the authors of Differentiable Volumetric Rendering [5]).

**Experiment Details.** We train a single model on all 13 classes. We train until convergence (approx. two days) at a resolution of $64 \times 64$ and a batch size of 300 until convergence for a total of approx. 3 days using the ADAM optimizer with a step size of $10^{-4}$. We choose the parameter $\lambda_{lat}$ as $1e2$. At test time, we freeze the network parameters, initialize all latent codes to the prior mean (i.e., all zeros), and optimize the latent codes until convergence. Leveraging an aggressive learning rate schedule, auto-decoding converges within 200 iterations or approximately 1 second for a batch of 300 shapes. Hyperparameters were discovered via unstructured search.

### 2.5 Single-class single shot reconstruction

**Dataset.** We use the dataset proposed by Sitzmann et al. [9], hosted by the authors of pixelNeRF [11] under https://drive.google.com/drive/folders/1PsT3uKwqHHD2bEEHkIXB99AlIjtmrEiR.

**Experiment Details.** We train separate models for the "cars" and "chairs" classes. We train in two resolution stages. First, at a resolution of $64 \times 64$ and a batch size of 300 for 200k steps, then, at a batch size of 75 at a resolution of $128 \times 128$ until convergence (total approx. 3 days), both using the ADAM optimizer with a step size of $10^{-4}$. We choose the parameter $\lambda_{lat}$ as $1e2$. At test time, we freeze the network parameters, initialize all latent codes to the prior mean (i.e., all zeros), and optimize the latent codes until convergence, as in the multi-class single-shot experiment. Hyperparameters were discovered via unstructured search.

## 3 Additional Results

In fig. 1 we show novel views of training set objects together with extracted epipolar plane images and depth maps. Please see the supplemental video for extensive qualitative results.

In Table 1 we show additional results comparing rendering complexity of LFNs to other existing methods, for larger, $800 \times 800$ pixel images.

**Comparison to conditioning-by-concatenation.** We performed an ablation study comparing hypernetworks to the simpler alternative of conditioning via concatenation. We parameterized the LFN identically to the MLPs in pixelNeRF and DVR: A 5-layer, fully-connected residual MLP with ReLU activations and 512 hidden units. The latent code was directly concatenated to the input ray coordinate. The remaining experimental settings were identical to those in Section 2.4 above. Both methods were trained for approximately 3 days on a single RTX 6000 GPU. In this setting, conditioning via concatenation yields significantly worse performance, as shown in Table 2. This is in line with the insight from Pi-GAN [1], where FILM-conditioning [6], which is an intermediate between a hypernetwork and conditioning via concatenation, was found to perform significantly better than conditioning via concatenation. Similarly, Facebook AI has found the hypernetwork-conditioned SRN to outperform concatenation-conditioned DVR and NeRF alternatives (see minute 16:30 of video associated to [7]). We further note that conditioning via concatenation is a special case of a hypernetwork, namely a single linear layer outputting the biases of the first LFN layer - see for instance appendix of [8] for a proof.

We note that we do not claim that the proposed method of conditioning is superior to any other method of conditioning. Rather, the proposed LFN framework is compatible with any conditioning method. We merely argue that to learn multi-view consistent light fields, we require some form of meta-learning, and see the analysis of the optimal conditioning method as an avenue for future work.

**Ablating overfitting vs. generalization.** To validate our claim that meta-learning on a dataset of 3D shapes enables consistent novel view synthesis on test objects by learning a multi-view consistency prior in light-fields, we benchmarked PSNR of novel views on 10 objects in the 10 set, reconstructed from a single observation using models trained on 1, 10, 100, 1k, and up to 2500 shapes, both in the single-class and multi-class settings. Experimental settings are as in Sections 2.4 and 2.5. Please

see Table 3 for results. The trend in PSNR is consistent improvement, suggesting that additional training-set objects would likely improve performance further.

We further note that meta-learning across a dataset of 3D scenes serves two distinct roles: learning a space of multi-view consistent light fields, and learning a prior over 3D scenes to enable few-shot reconstruction. To better disentangle these two roles, we ran a second experiment. For each instance in the training set, which was drawn from the ShapeNet chair class, we randomly picked 5 views, such that on average, almost every surface point of each chair has been observed, but not every oriented ray. I.e., while the 3D surfaces of the chairs are densely sampled, their light fields are not. We trained LFNs on 10, 100, 1000, and all (4.5k) chair instances, including the 10 chairs previously mentioned, and evaluated view synthesis performance in PSNR on a held-out set of novel views for each of these 10 chairs. This tests only the first property of the meta-learning approach - i.e., how the meta-learning learns multi-view consistency as an abstract property - without entangling it with the capability to perform few-shot reconstruction. Experimental parameters were as in 2.5. Please see Table 4 for results. The performance similarly improves logarithmically.

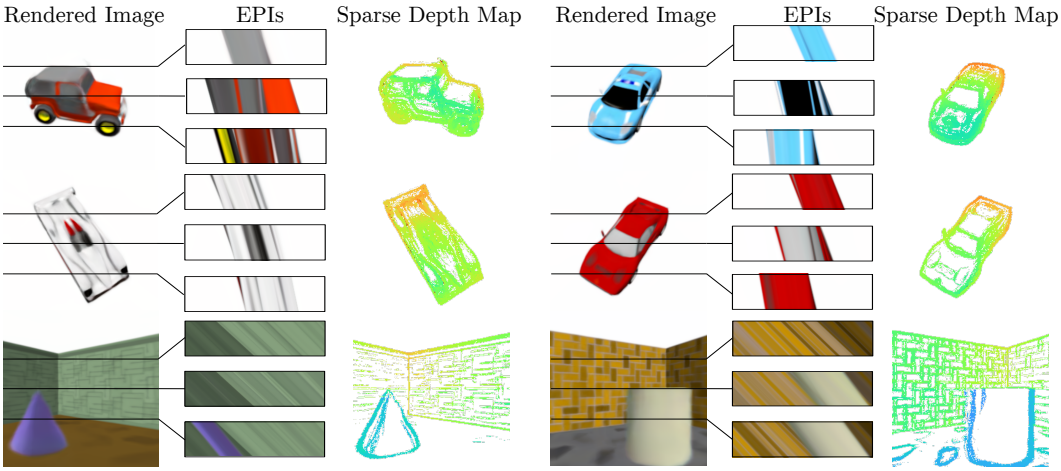


Figure 1: Novel views of cars and rooms with Epipolar Plane Images and depth maps.

Table 1: **Comparison of rendering complexity, continued.** Data for all architectures aside from LFN taken from [7]; see 17:00 of associated video.

| | LFNs | SRNs [9] | DVR [5] | IDR [10] | NeRF [4] |
|---|---|---|---|---|---|
| clock time for 800 × 800 image (ms) | 20.5 | 1e3 | 9e4 | 1e5 | 2e4 |

Table 2: **Comparison of meta-learning approach.** We compared conditioning via a hypernetwork, used in the paper, to conditining-by-concatenation (CvC below) in the 13-class ShapeNet dataset. All results are PSNR in dB.

| Method | plane | bench | cbnt. | car | chair | disp. | lamp | spkr. | rifle | sofa | table | phone | boat | mean |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Hypernetwork | 29.95 | 23.21 | 25.91 | 28.04 | 22.94 | 20.64 | 24.56 | 22.54 | 27.50 | 25.15 | 24.58 | 22.21 | 27.16 | 24.95 |
| CvC | 24.08 | 19.43 | 21.01 | 23.57 | 19.81 | 17.47 | 19.05 | 19.40 | 22.24 | 20.85 | 19.81 | 17.64 | 24.02 | 20.64 |

**Overfitting a densely sampled 3D scene.** We demonstrate that LFNs can in principle represent high-frequency information. We fit an LFN to all the views of the "Fern" scene from the NeRF [4] dataset, which consists of photos of a real-world scene captured with a DSLR camera. We add positional encodings to the plucker embedding, following NeRF. Please see qualitative results in Fig 2, as well as the supplemental video. LFNs succeed in reproducing all context images perfectly, proving that in principle, an LFN is capable of parameterizing realistic, 4d high-frequency content. As expected, rendering out the intermediate views leads to basically random images, please see this video, as there is no mechanism to enforce multi-view consistency, in contrast to the experiments in the paper, where this prior was learnt using meta-learning. We note that fundamentally, there are

Table 3: **Multi-view consistency vs. training set size.** As we scale the size of the training set, the multi-view consistency prior improves consistently, both in the single-class and multi-class setting. All results are PSNR in dB. Top row denotes number of object instances in the traninig set. The cars dataset contains approximately 2500 objects, while the NMR dataset contains approximately 2000 objects per class.

| Dataset | 1 | 10 | 100 | 1k | all |
|---|---|---|---|---|---|
| Cars (Single-class) | 16.21 | 18.98 | 20.76 | 23.09 | 23.56 |
| NMR (Multi-Class) | 18.03 | 19.77 | 20.82 | 23.87 | 24.95 |

Table 4: **Disentangling multi-view consistency vs. object prior.** Columns indicate number of instances of chair ShapeNet class. Each LFN was trained on 5 random views of each instance and evaluated on 10 novel views of a fixed subset of 10 training set chairs.

| | 10 | 100 | 1k | all |
|---|---|---|---|---|
| PSNR (dB) on novel views | 15.88 | 16.83 | 19.12 | 22.15 |

Figure 2: **Overfitting a single 3D scene.** We overfit an LFN on the context views of the "Fern" scene from NeRF [4]. Context views are reproduced almost perfectly, demonstrating that an LFN can in principle fit high-frequency detail (left). Rendering out intermediate views with unobserved rays fails, as no multi-view consistency prior is enforced (right).

two different regimes of interest. (1) The overfitting regime, where a neural scene representation is fit to a 3D scene that is completely observed, i.e., every surface point is observed enough times to enable triangulation. (2) The prior-based reconstruction regime, where we aim to reconstruct a 3D scene given incomplete observations. We note that problem (2) is of great interest in the field of artificial intelligence, where we regularly infer neural scene representations from incomplete observations. LFNs are immediately applicable to this regime, opening avenues of future work in scene understanding, scene decomposition, reinforcement learning (where a tractable inverse graphics model is of critical importance), etc.

## 4 References


[1] E. R. Chan, M. Monteiro, P. Kellnhofer, J. Wu, and G. Wetzstein. pi-gan: Periodic implicit generative adversarial networks for 3d-aware image synthesis. *Proc. CVPR*, 2020.

[2] S. A. Eslami, D. J. Rezende, F. Besse, F. Viola, A. S. Morcos, M. Garnelo, A. Ruderman, A. A. Rusu, I. Danihelka, K. Gregor, et al. Neural scene representation and rendering. *Science*, 360(6394):1204–1210, 2018.

[3] H. Kato, Y. Ushiku, and T. Harada. Neural 3d mesh renderer. In *Proc. CVPR*, pages 3907–3916, 2018.

[4] 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. ECCV*, 2020.

[5] M. Niemeyer, L. Mescheder, M. Oechsle, and A. Geiger. Differentiable volumetric rendering: Learning implicit 3d representations without 3d supervision. In *Proc. CVPR*, 2020.

[6] E. Perez, F. Strub, H. de Vries, V. Dumoulin, and A. C. Courville. Film: Visual reasoning with a general conditioning layer. In *AAAI*, 2018.

[7] J. Reizenstein, R. Shapovalov, P. Henzler, L. Sbordone, P. Labatut, and D. Novotny. Common objects in 3d: Large-scale learning and evaluation of real-life 3d category reconstruction. In *Proc. ICCV*, pages 10901–10911, 2021.

[8] V. Sitzmann, E. R. Chan, R. Tucker, N. Snavely, and G. Wetzstein. Metasdf: Meta-learning signed distance functions. *Proc. NeurIPS*, 2020.

[9] V. Sitzmann, M. Zollhöfer, and G. Wetzstein. Scene representation networks: Continuous 3d-structure-aware neural scene representations. In *Proc. NeurIPS 2019*, 2019.

[10] L. Yariv, Y. Kasten, D. Moran, M. Galun, M. Atzmon, B. Ronen, and Y. Lipman. Multiview neural surface reconstruction by disentangling geometry and appearance. *Proc. NeurIPS*, 2020.

[11] A. Yu, V. Ye, M. Tancik, and A. Kanazawa. pixelnerf: Neural radiance fields from one or few images. *Proc. CVPR*, 2020.