Neural Point Light Fields

Supplementary Material

Julian Ost[1] Issam Laradji[2] Alejandro Newell[3] Yuval Bahat[3] Felix Heide [1,3]

[1]Algolux [2]McGill [3]Princeton University

In this supplemental document, we present additional details and results in support of the findings from the main manuscript. Specifically, we include

- Additional Epipolar Light Field Visualizations (Section 1),
- Implementation Details (Section 2),
- Training Details (Section 3),
- Ablation Studies (Section 4),
- Ablation Comparisons (Section 5),
- Additional Results (Section 6), and
- Additional Discussion on Method Limitations (Section 7).

1. Additional Epipolar Light Field Visualizations.

Traditionally, view-dependent effects captured in light fields have been analyzed by presenting epipolar plane images [1] (EPI), which capture 2D slices of a light field that interpolating between two extremal views. Typically light field reconstruction methods have access to multiple views of the same scene captured from a similar distance, which can then be used for training or directly interpolating the light field. In contrast, Neural Point Light Fields is trained on images taken on a *single capturing trajectory* for each scene. Nevertheless the proposed method is capable of recovering accurate view-dependent effects. In Fig. 1 we present EPIs from trained Neural Point Light Fields representations on two scenes. These show that Neural Point Light Fields is able to perform view extrapolation with accurate view-dependent light field effects.

Scene Epipolar Image Scene Epipolar Image

Figure 1. Epipolar Plane Images. The green line on the images from the scene is equivalent to the cyan line on the EPIs. View extrapolation on two scenes only observed on a single trajectory accurately captures view-dependent light field effects.

2. Implementation Details

Sec. 3 of the main manuscript describes our method that reconstructs a Neural Point Light Field, given a sparse set of images and corresponding point cloud data. We next provide additional implementation and training details. We note that, to allow for reproducibility, we will publish our source code, and the corresponding pre-trained models: https://light.princeton.edu/neural-point-light-fields

Point Cloud Feature Encoding. Point cloud features are extracted following Goyal et al. [4]. And as described in the main paper, the point cloud is scaled into a $[1, -1]$ cube. The N points are then projected onto the six planes of the cube, of size $H \times W = 128 \times 128$.

$$\boldsymbol{Proj} : \mathbb{R}^{N,3} \rightarrow \mathbb{R}^{6,128,128} \tag{1}$$

These depth projections per plane are then fed into a convolutional feature extractor $\boldsymbol{F}_{\theta_{ResNet18}}$, a vanilla ResNet 18 [5], yielding a $16 \times 16 \times 128$ feature map on each cube side. We reproject the resulting six feature maps onto their corresponding points in the point cloud. This results in an embedding code $\boldsymbol{l}_k \in \mathcal{R}^{6\times 128}$ for each point $\boldsymbol{x}_k,\ k \in [1, N]$:

$$\boldsymbol{ReProj} : \mathbb{R}^{6,16,16,128} \rightarrow \mathbb{R}^{N,6,128} \tag{2}$$

Ray Feature Attention. As we outline in Sec. 3.2 of the main manuscript, a multi-head self-attention module computes a weighted ray feature from its relevant point features. We first predict values $V_{k,j}$ and keys $K_{k,j}$ for each point-ray pair from the embedding vector $\boldsymbol{v}_{k,j}$ with $\boldsymbol{F}_{\theta_V}$ and $\boldsymbol{F}_{\theta_K}$. For each ray j, we predict a query vector $Q_{k,j}$ with $\boldsymbol{F}_{\theta_Q}$. For all three functions we use a single linear layer of width 128 and a ReLu activation. For the following multi-headed self-attention module we follow Vaswani et al. [12], using 8 heads and the scaled dot product from Eq. 3 in each head i, that is

$$\text{out}_{\text{head}_i} = \text{softmax}\left(\frac{\left(QW_i^Q\right)\left(KW_i^K\right)^T}{\sqrt{d_k}}\right)\left(VW_i^V\right) \tag{3}$$

Color Prediction. For each ray $\boldsymbol{r}_j$ and latent embedding $\boldsymbol{l}_j$ we predict a color value $\boldsymbol{C}_j$, using an eight layer MLP described in Tab. 1.

Layer	note	Input Size	Output Size
0	Encoded Ray Direction + Ray Feature	27 + 128	256
1	Fully Connected ReLU	256	256
2	Fully Connected ReLU	256	256
3	Fully Connected ReLU	256	256
4	Fully Connected ReLU	256	256
5	Fully Connected ReLU + Skip	256 + 27 + 128	256
6	Fully Connected ReLU	256	256
7	Fully Connected ReLU	256	256
8	Fully Connected Sigmoid	256	3

Table 1. Ray color network architecture. This MLP predicts the color of a ray, given an encoded ray direction and a ray feature embedding.

3. Training Details

We train all models with the same set of hyperparameters. All parameters of the model are optimized using the the Adam optimizer [6], and setting the learning rate to 1×10^{-3}. At each training step, we use a batch of 8192 rays, which correspond to 4096 pixels randomly sampled from 2 images.

Point Cloud Augmentation. As we mention in Sec. 3.1 of the main manuscript, we proposed data augmentation schemes to promote consistency along a trajectory along the scene. Depth projections from adjacent frames are similar and intrinsically offer consistency between the encodings for different time steps. To enforce consistency in the point features, i.e., encode the ones that capture the same location of the underlying scene geometry, we propose the following data augmentation and training scheme.

To this end, we assume our method is applied after the completion of captures, such that the information provided from adjacent frames is available. We align and merge the set of points from adjacent frames to explicitly capture larger parts of the scene, and to introduce consistent data points across frames. Alignment is done by applying a variant of the iterative closest points (ICP) algorithm [10] provided in the Open3D library [14]. The ICP algorithm is applied on a set $P_{i,m} =$

Figure 2. Qualitative Ablation Experiments. A purely distance-based encoding (left column) of the ray features from Eq. 5 results in artifacts, especially for the longer scene in the first row. The quality of the synthesized outputs of the method increases with the number of point N, until saturating around $N = 5000$. Too few points result in lack of high frequency details.

$\{\mathcal{P}_{i-m}, ..., \mathcal{P}_i, ..., \mathcal{P}_{i+m}\}$ of $2m + 1$ adjacent frames, with $m = 10$ in all our experiments. The resulting point cloud is large and contains overlapping regions, making it inefficient and redundant for the purpose of feature extraction. Therefore at each training step we randomly sample N points from the augmented point cloud, resulting in a new set of points $\tilde{P}_{i,m} = \{\boldsymbol{x}_0, ..., \boldsymbol{x}_N\} \subset P_{i,m}$.

In addition, we randomly choose a point cloud data $P_{i+h,m}$with $h \in \{-H, ..., H\}$ from the H neighbouring time step during training, to further promote a consistent reconstruction of the large scene only encoded from local scene sections. In our experiments we choose $H = 10$.

List of Scenes. We demonstrate our approach on the Waymo Open Dataset [11]. We select a set of scenes from this dataset in our experiments that do not include highly dynamic scene components. Tab. 2 lists the starting and ending frames corresponding to each of the scenes we used for training.

Set	Segment Name/ID	First Frame	Last Frame
validation 0000	segment-10247954040621004675_2180_000_2200_000_with_camera_labels	0	80
validation 0000	segment-10713922294950850 36_1844_790_1864_790_with_camera_labels	135	197
validation 0000	segment-11037651371539287009_77_670_97_670_with_camera_labels	0	164
validation 0001	segment-13469905891836363794_4429_660_4449_660_with_camera_labels	0	197
validation 0002	segment-14333744981238305769_5658_260_5678_260_with_camera_labels	0	198
validation 0002	segment-14663356589561275673_935_195_955_195_with_camera_labels	0	197

Table 2. List of all training scenes and corresponding frames, from the Waymo Open Dataset [11].

Point Cloud and Depth Maps for Compared Methods. The depth images for GSN and the per ray depth information for DS-NeRF are directly projected from the recorded point cloud data, allowing these methods to see point cloud data in addition to RGB images.

4. Ablation Studies

In Sec. 4.3 of the main manuscript we report a set of ablation experiments indicating that a learned self-attention module leads to higher quality representations, compared to using a distance-based heuristic and a sum over all point features. This weighing is necessary since we exploit multiple points for each ray aid the quality of the feature encoding.

We report the results of additional ablation experiments in Tab. 3 and Fig. 2, which provide insights on the effect of several design choices, as we discuss next.

Point Features. We extract features for each point $\boldsymbol{x}_k$ using a CNN backbone applied on the projection of those points into the an enclosing cube. This is in contrast to the commonly used coordinate based features extractors. We experimented with such feature extractors using PointNet [8], but were unable to train our method effectively using the extracted features, despite

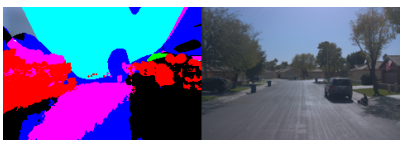

Figure 4. Using PointNet to encode features on the point cloud does fail. We present a rendering using features encoded with PointNet and the respective reference frame from the dataset.

Figure 3. Reconstruction examples for two additional scenes. Row 1 and 2 show examples from a night scene. Row 3 and 4 show examples from a scene that was captured under direct sunlight. The reconstruction quality of scenes in challenging lighting conditions is similar across all scene, except NeRF that fails for large scenes.

	raw point cloud	$N = 10$ (random)	$N = 50$ (random)	$N = 100$	$N = 500$	$N = 1000$	$N = 5000$	PointNet (Loc.+Glob.)	PointNet (Loc.)	Naive Sum	Heuristic	$K = 0$	$K = 1$	$K = 2$	Dist. Enc.	Ours (K=8, N=20000)
Reconstruction																
PSNR ↑	27.71	26.37	27.15	29.93	30.44	30.88	**31.37**	5.47	6.24	4.84	24.56	18.88	29.83	30.95	26.49	31.32
SSIM ↑	0.818	0.799	0.811	0.868	0.875	0.881	0.889	0.121	0.0104	0.108	0.800	0.650	0.851	0.871	0.785	**0.890**
LPIPS ↓	0.185	0.243	0.211	0.150	0.126	0.119	0.105	0.799	0.779	1.110	0.191	0.543	0.146	0.122	0.193	**0.101**
Novel View Synthesis																
PSNR ↑	26.69	26.05	26.54	29.11	29.53	30.07	**30.46**	5.93	6.27	4.84	23.64	18.73	29.411	30.02	25.35	30.29
SSIM ↑	0.801	0.799	0.804	0.860	0.867	0.873	**0.880**	0.157	0.011	0.110	0.784	0.643	0.845	0.865	0.767	**0.880**
LPIPS ↓	0.193	0.245	0.216	0.155	0.131	0.126	0.112	0.786	0.777	1.107	0.201	0.548	0.150	0.126	0.204	**0.109**

Table 3. We evaluate the sensitivity of the proposed method to a set changes. For a different feature encoder such as the chosen variation of PointNet [8] and ray features from a non-weighted sum over all point features, our method is not able to learn a representation at all. A heuristic weighting based on a points distance is inferior to the ray self-attention. Until $N = 5000$ all metrics improve. A similar behavior can be reported for K, that improves a lot unril $K = 2$. All ablation experiments were evaluated on two scenes.

performing an extensive hyper-parameter search. We began by using the output of the last layer of the PointNet segmentation network, including concatenation with global features per point, which produced the results shown in Fig. 4. We also experimented with using per-point features, in which case we do not concatenate global and local point features in the segmentation network to enforce per-point encodings. However, this did not have a significant effect on the results, see Tab.3.

Ray Features from Positions. The extracted point features $\boldsymbol{l}_k$ are one part of the point-ray embedding $\boldsymbol{v}_{k,j}$ presented in Eq. 4. In addition we also add a distance based component, that locates each point with respect to the relevant ray.

$$\boldsymbol{v}_{k,j} = (\boldsymbol{l}_k \oplus \gamma(\theta_{k,j}) \oplus \gamma(\psi_{k,j}) \oplus \gamma(d_{k,j})) \tag{4}$$

Since the feature extraction adds a significant computational complexity to our method by encoding features from the full point cloud when only using a small number of closest points per ray. An alternative approach could ignore $\boldsymbol{l}_k$ and only extract a point-ray embedding $\boldsymbol{v}_{k,j}$ from the relative locations of the ray and its closest points:

$$\boldsymbol{v}_{k,j} = (\gamma(\theta_{k,j}) \oplus \gamma(\psi_{k,j}) \oplus \gamma(d_{k,j})) \tag{5}$$

Experimenting with this alternative approach shows that this information is not enough to reconstruct the Light Field of a given scene, and results in heavy artifacts, as demonstrated in the first column of Fig. 2.

Number of Points and Lidar Point Cloud Quality. We next evaluate the effect of points N in the point cloud used to encode the scene representation. While the effect on computation time for the encoding step is negligible when using a projection based feature extractor, results in Fig. 2 and Tab. 3 (using $N = 10, 50, 100, 500, 1000, 5000, 20000$) indicate that rendering quality drops when using drastically too few points. Too few points result in smoother and noisy renderings. This is due to the low number of points which are relevant for the rendering of a ray in this scenario. The rendering quality increases with N, until it saturates beyond certain N values, $N = 5000$ in our case. Point sampling from the input point cloud is specific to a dataset and scene and may vary with scene complexity and point cloud quality. We uniformly sample N point in the 3D space, out of the given point cloud. For $N < 50$, we first uniformly sample 100 points in space, to ensure a relatively uniform spatial distribution, before re-sampling N points. This allows experimenting with extreme number of points ($N = 10$ or $N = 50$, reported in Tab. 3), which corroborate that the rendering quality degrades gracefully even for extremely sparse per-view point clouds.

Figure 5. View reconstruction and extrapolation for NeRF++ and FVS, compared to Neural Point Light Fields.

5. Additional Comparisons to Mesh Based Approaches and NeRF++

We compare in the main text against NeRF [7], its variant DS-NeRF [2] which exploits additional point cloud information, and against the generative method GSN [3]. We additionally compare against NeRF++ [13] and Free View Synthesis (FVS) [9]. Fig. 5 show that our method significantly outperforms NeRF++ in view extrapolation, and compares favorably to FVS in both view extrapolation and reconstruction. Note that these mesh-based step-wise methods fundamentally differ from the proposed method. While our method is an end-to-end neural rendering approach, FVS generates a mesh first in a pre-processing step using traditional multi-view stereo methods. FVS then projects adjacent images onto a mesh proxy geometry. As such, the rendering quality depends on the availability of these adjacent views and the mesh accuracy (see artefacts in Fig. 5 around vehicles (top center) and image periphery (bottom center)). This is because our method does not require nearby images for view synthesis. In Tab. 5 and Fig. 8, we present additional reconstruction and novel view synthesis results for all methods. The more advanced method NeRF++ shows slightly better performance compared to NeRF in the reconstruction and view synthesis task. FVS has not been trained on the selected scenes, but has access to all frames except the left out frame during test time. We only evaluate the novel view synthesis task. As expected from the performance on the extrapolation task, it shows stable and comparable results to other baseline methods trained on the scene.

	NeRF [7]	DS-NeRF [2]	GSN [3]	FVS [9]	NeRF++ [13]	Ours
Reconstruction						
PSNR ↑	29.48	26.53	17.98	-	<u>30.65</u>	**31.52**
SSIM ↑	0.815	0.778	0.512	-	<u>0.825</u>	**0.882**
LPIPS ↓	0.289	0.306	<u>0.136</u>	-	0.159	**0.110**
Novel View Synthesis						
PSNR ↑	22.47	<u>26.15</u>	16.83	26.01	24.08	**29.96**
SSIM ↑	0.700	0.772	0.464	0.842	<u>0.850</u>	**0.868**
LPIPS ↓	0.389	0.310	0.174	<u>0.123</u>	0.143	**0.119**

Table 4. We report PSNR, SSIM and LPIPS results on 5 static scenes from the Waymo Open Dataset [11] using images from the front camera for NeRF [7], depth-supervised NeRF [2], generative scene networks [3], NeRF++ [13], and Free View Synthesis [9] and Neural Point Light Fields. For PSNR and SSIM, higher is better; for LPIPS lower is better. The best values are written in **bold**, the next best are <u>underlined</u>.

6. Additional View Synthesis Results

The first four rows of Fig. 3 present reconstruction results for two additional scenes with challenging lighting conditions, including night time capturing and direct sunlight. NeRF and DS-NeRF achieve similarly blurry outputs on the night scene, with artifacts which do not allow to recognize individual scene objects. Neural Point Light Fields, in contrast, is able to accurately reconstruct large portions of the scene. In the short scene with direct sun light (two bottom rows) NeRF is able to capture a little more detail than Neural Point Light Fields in the third row. However, NeRF fails for larger scenes, as the reconstruction in the 4th row taken around 140 frames shows. This matches the reconstruction results in Tab. 2 in Sec. 4 of the main paper. In Fig. 8 we present additional novel view interpolation results with a qualitative trend that matches the reconstruction results from Fig. 3. We refer the reader to the supplemental video for additional full trajectory results and additional extrapolation results.

7. Additional Discussion on Method Limitations

Generalization. While the method accurately recovers views, that have a partial overlap with the training views, even along a single trajectory, completely unseen scene portions, e.g., the back of a car that we observed only from the front can only be hallucinated by the proposed method. An example for view extrapolation into such regions of a scene is presented in Fig. 6. Renderings in those regions result in imaginary objects conditioned on points similar to seen objects such as a tree as well as artifacts point-ray combinations outside the training distribution. We note that this behavior is expected, as our current model is not generalizing across scenes and has no knowledge on how to interpret new point ray combinations.

Illumination. Our method does not handle rapid illumination changes (see, e.g., supplementary video around 1:34 and Fig.7), which often result from changes in direct sunlight illumination in the captured data. This might be resolved using a learned exposure and tone-mapping module, which we relegate to future work.

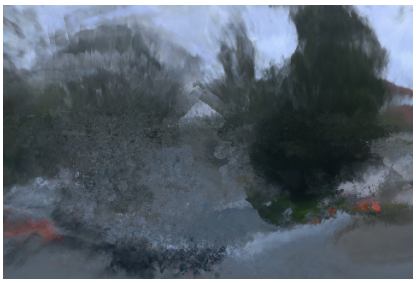

Figure 6. Extrapolation to completely unseen scene portions, where the proposed method has to hallucinate occluded object regions never observed along a training trajectory.

Figure 7. Rapid illumination changes and different exposure values in adjacent frames are not represented by point features.

Figure 8. Novel View Interpolation. Additional results on novel view interpolation While NeRF provides acceptable results for short scenes (bottom), it fails for long sequences.

References

[1] R. C. Bolles, H. H. Baker, and D. H. Marimont. Epipolar-plane image analysis: An approach to determining structure from motion. *International journal of computer vision*, 1(1):7–55, 1987. 1

[2] K. Deng, A. Liu, J.-Y. Zhu, and D. Ramanan. Depth-supervised nerf: Fewer views and faster training for free. *arXiv preprint arXiv:2107.02791*, 2021. 5

[3] T. DeVries, M. A. Bautista, N. Srivastava, G. W. Taylor, and J. M. Susskind. Unconstrained scene generation with locally conditioned radiance fields. *arXiv preprint arXiv:2104.00670*, 2021. 5

[4] A. Goyal, H. Law, B. Liu, A. Newell, and J. Deng. Revisiting point cloud shape classification with a simple and effective baseline. *arXiv preprint arXiv:2106.05304*, 2021. 2

[5] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 2

[6] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2015. 2

[7] 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, 2020. 5

[8] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 652–660, 2017. 3, 4

[9] G. Riegler and V. Koltun. Free view synthesis. In *European Conference on Computer Vision*, pages 623–640. Springer, 2020. 5

[10] S. Rusinkiewicz and M. Levoy. Efficient variants of the icp algorithm. In *Proceedings third international conference on 3-D digital imaging and modeling*, pages 145–152. IEEE, 2001. 2

[11] P. Sun, H. Kretzschmar, X. Dotiwalla, A. Chouard, V. Patnaik, P. Tsui, J. Guo, Y. Zhou, Y. Chai, B. Caine, et al. Scalability in perception for autonomous driving: Waymo open dataset. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2446–2454, 2020. 3, 5

[12] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017. 2

[13] K. Zhang, G. Riegler, N. Snavely, and V. Koltun. Nerf++: Analyzing and improving neural radiance fields. *arXiv preprint arXiv:2010.07492*, 2020. 5

[14] Q.-Y. Zhou, J. Park, and V. Koltun. Open3d: A modern library for 3d data processing. *arXiv preprint arXiv:1801.09847*, 2018. 2