Neural Point Light Fields

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

Abstract

We introduce Neural Point Light Fields that represent scenes implicitly with a light field living on a sparse point cloud. Combining differentiable volume rendering with learned implicit density representations has made it possible to synthesize photo-realistic images for novel views of small scenes. As neural volumetric rendering methods require dense sampling of the underlying functional scene representation, at hundreds of samples along a ray cast through the volume, they are fundamentally limited to small scenes with the same objects projected to hundreds of training views. Promoting sparse point clouds to neural implicit light fields allows us to represent large scenes effectively with only a single radiance evaluation per ray. These point light fields are as a function of the ray direction, and local point feature neighborhood, allowing us to interpolate the light field conditioned training images without dense object coverage and parallax. We assess the proposed method for novel view synthesis on large driving scenarios, where we synthesize realistic unseen views that existing implicit approaches fail to represent. We validate that Neural Point Light Fields make it possible to predict videos along unseen trajectories previously only feasible to generate by explicitly modeling the scene.

1. Introduction

Learning implicit volumetric scene representations has made it possible to synthesize photo-realistic images of single scenes [20, 24, 27, 39]. The most successful methods combine a conventional volumetric rendering approach with a coordinate-based neural network that predicts density and radiance [24]. As such, instead of explicitly storing density and radiance in a high-dimensional 5D volume, these methods represent this volume as a learned function, that can be further decomposed into radiance and illumination [53, 40, 5]. Although the implicit volumetric representation is highly memory-efficient and differentiable, it also fundamentally requires sampling the volume, that is evaluating the coordinate-based network, hundreds of times for each ray for a given pixel. This mandates long training and small volumetric support inside the volume.

Figure 1: Neural Point Light Fields encode the information of a Light Field representation of a scene on a point cloud capture. An image is rendered for each camera ray based on the local encoding of the Light Field on relevant points.

To tackle these challenges, hybrid representations [13, 19, 15] are used to embed or "bake" local radiance functions on explicit sparse proxy representations such as coarse voxel grids, point clouds or meshes to enable faster rendering by ignoring empty space. While this approach drastically improves rendering speed at test time, it still requires volumetric sampling during training. This is because the scene geometry must be learned during the training process. These methods share the limitations of volumetric approaches during training and, as such, have also been limited to small scenes that are costly to train. Learning representations for large outdoor scenes is an open challenge.

Unfortunately, approaches that are free of implicit representations do not yet offer an alternative. Specifically, explicitly storing features on proxy geometry [34, 33, 17] has not been able to achieve the same quality as volumetric methods when interpolating a view without a nearby training sample. Existing formulations utilize geometry as a projection canvas combined with features extracted from target views, and therefore require a large number of input images near the target view.

In this work, we depart from volumetric models and introduce Neural Point Light Fields, a local implicit representation that encodes a light field on a point cloud. The proposed representation supports novel view synthesis in

large outdoor scenes without strong parallax needed as in volumetric methods. Although recent automotive depth estimation networks make it possible to estimate dense depth point clouds from video data, we assume measured lidar point clouds as input to our method, especially as lidar data is readily available in most outdoor vehicle datasets [42, 10] and recently released smartphones. Although sparse, the lidar geometry provides enough cues to encode a local light field on the point cloud. Instead of a 5D volumetric radiance function, or a conventional 4D light field [18], we propose to formulate a light field only depending on the two dimensional ray direction and a one dimensional index pointing to a point cloud featureThis formulation makes it possible to evaluate a *single* radiance prediction per ray.

We extract features for each point with a learned feature extractor on point cloud projections [11]. For a given camera pose, we shoot rays for each pixel and select a set of close points inside the point cloud. The features from these selected points are then weighted by passing the points relative position to the ray and features through an attention module, resulting in a single ray feature code. The color for each ray is then reconstructed by an implicit light field representation conditioned by this feature code. We assess the proposed method on a large-scale automotive driving dataset [42] and demonstrate novel view synthesis along unseen trajectories with quality unseen before.

Specifically, we make the following contributions

- We introduce Neural Point Light Fields, a representation that implicitly encodes features in a point cloud, requiring only a single radiance evaluation per ray.
- The proposed method lifts the restrictions of volumetric scene representations by exploiting sparse geometry available in estimated or captured point clouds.
- We validate the proposed method on novel video synthesis tasks for large-scale driving scenes, demonstrating the proposed method's capability of generating realistic novel views along trajectories which cannot be handled by existing implicit representation methods.

Our code and trained models are available online.

Scope Even though existing automotive datasets include data from multiple cameras, lidar and radar sensors, we focus on learning from a single camera with a single trajectory per scene, and without highly dynamic scene motion. In contrast to densely observing the scene across a full hemisphere [24], captured images in our case are sparsely distributed along the driving trajectory We note that extending training to multiple camera views is not straightforward, as camera poses, exposure and tone-mapping differences have to be accounted for. Exploiting multiple cameras and adding dynamic object support to the proposed method could constitute exciting future directions.

2. Related Work

Novel View Synthesis. Synthesizing novel views from a set of unstructured images of a scene is a long standing problem in computer vision and graphics. Early work on image-based rendering introduced light fields [18] as a 4D parameterization of light rays and their respective radiance in a scene. Light fields are derived by considering a convex subspace of the 5D plenoptic function [1] that parameterizes a ray by a point in space and a direction. Conventional light field rendering, i.e., interpolation of novel views, requires a large set of densely sampled views of the light field, as traditional optimization methods [47, 48] handle only small parallax changes between the interpolated and measured view. Recently, methods relying on deep learning [23] allowed recovering light field from plane sweep volumes, using 3D convolutional neural networks.

An orthogonal line of work investigates the reconstruction of explicit 3D models from a set of images. By optimizing the reprojection error between features found in all images, multi-view reconstruction methods are capable of reconstructing the underlying scene geometry and camera poses [2, 36]. These methods can reconstruct large scenes, but require many images to achieve high quality, and, in contrast to image-based rendering methods, struggle to synthesize photorealistic novel views.

Neural Scene Representations. An emerging large body of work explores learned representations in scene reconstruction pipelines. These neural rendering approaches are able to generate photo-realistic novel views [20, 27], while reconstructing high-quality scene geometry. Existing methods rely on explicit, implicit, or hybrid representations of the scene. Explicit methods encode texture or radiance on recovered proxy scene geometry, such as meshes [44], multi-planes [8, 21, 23, 41, 54], voxels [38] or points [3, 31]. Instead of jointly recovering geometry and appearance, these methods can focus on recovering image details. Nonetheless, relying on explicit proxy geometry limits the achievable image quality. To overcome the reliance on such geometry, researchers explored implicit representations using coordinate-based networks, e.g. the successful NeRF method [24]. However, achieving photo-realistic quality for diverse tasks [22, 29, 49, 40, 37, 26, 28] comes at the cost of expensive training and testing. The lack of explicit geometric knowledge requires densely evaluating the implicit network within the volume, with the majority of samples located in empty space, and therefore not contributing to the rendered pixel color. Extensions [9] have tackled this issue at test time evaluation by either predicting the sampling regions [25, 4] or explicitly extracting proxy geometry

[19] after training. DS-NeRF [6] uses 3D keypoints reconstructed from COLMAP on a scene to supervise the opacity prediction with those sparse keypoints, which speeds up training. Neural Sparse Voxel Fields (NSVF) [19] use a hybrid representation that stores implicit functions in a voxel grid. NeRF++ proposes to separate background and foreground scene components [51], which help improve the rendering quality, primarily for distant scene objects. However, all of these methods *struggle with large scale outdoor scenes or scenes with very few view directions*. In contrast, the proposed approach allows rendering large outdoor scenes from a sparse set of observations, by introducing a light field parameterization on sparse scene geometry.

Multi-View Structure (MVS) Reconstruction. Reconstructing geometry such as point clouds or meshes from images [36, 35] can guide the training of implicit scene representations [6] or offer a scaffold for learned features [34, 17]. Riegler and Koltun [34, 33] propose such geometric scaffolds living on MVS-meshes. Kopanas et al. [17] showed that optimizing point locations from an initial point cloud, together with their novel view synthesis pipeline, can compensate for errors during reconstruction from MVS. These methods and similar [3] point based approaches use point clouds as a geometric proxy, while following a strict rendering and projection approach. In contrast, we propose a method that uses features not only by projecting them on to a proxy geometry, but encodes them from a 3D point cloud, and requires no input images during test time.

In the context of automotive scene reconstruction, SurfelGAN [50] proposes a representation with discrete textured surface elements (surfels), recovered from captured Lidar and RGB data. Novel views are rendered by a generator network from projections of the surfel RGB data. In contrast, we learn features directly embedded in the captured point cloud.

Encoding features directly on a point clouds has been extensively explored [32] for diverse tasks. Recent work revisited the use of multi-view projections of a point cloud for classification tasks [11, 12], similar to the proposed reconstructions from point clouds, but without using image features. Their method is robust to occlusions [12], and achieves state-of-the-art results on selected downstream tasks. Rather than solving a classification or segmentation task, we show that multi-view point cloud encoding can deliver rich local point features for reconstruction of novel views.

3. Point Light Fields

In this section, we introduce Point Light Fields. A Point Light Field encodes the light field of a scene on sparse point clouds. Assuming a camera-lidar sensor setup typical in robotic and automotive contexts [10], at time step i, the proposed method learns an RGB frame $\mathcal{I}_i$ as input and the corresponding point cloud capture $\mathcal{P}_i$. To learn a light field embedded on the point clouds corresponding to a video sequence, we devise three steps: an encoding step, a feature aggregation, and a point-conditioned light field prediction, all of which we describe in the following.

3.1. Per-point Feature Encoding

We first produce a feature embedding for each point in the point cloud. To do this, we follow the simple strategy presented by Goyal et al. [11]. The input point cloud is projected onto six planes, producing sparse depth images. These images are each fed directly into a convolutional network. We use the initial layers of a vanilla ResNet18 [14] to extract per-pixel features at one-quarter the input resolution. For a given point $\boldsymbol{x}_k$, we retrieve the corresponding feature vector at its projected location in each of the six views. These are concatenated together to produce the final feature encoding $\boldsymbol{l}_k \in \mathbb{R}^{6\times128}$.

We find it sufficient to normalize input point clouds to a canonical cube bounded by $[-1, 1]$ and use the 6 sides of the cube as projection planes. This works robustly even given the complexity of in-the-wild large-scale scenes. We perform ablations comparing features encoded using this strategy against alternative point-based models such as PointNet [32], see Supplementary Material.

The learned per-point features $\boldsymbol{l}_k$ do not depend on any image data and can be trained end-to-end with the full light field rendering. We can introduce augmentations such that the model does not overfit to a particular arrangement of points. This includes sampling different subsets of points from the full captured point cloud, and using point cloud captures from nearby time steps.

3.2. Light Field Feature Interpolation

Given a set of points $\mathcal{P}_i = \{\boldsymbol{x}_0, ..., \boldsymbol{x}_N\}_i$with $\boldsymbol{x}_k \in \mathbb{R}^3$, their encoded features $\boldsymbol{l}_k \in \mathbb{R}^{6\times128}$, and a camera view C_i, defined by its intrinsic $\boldsymbol{K}$, extrinsic $\boldsymbol{E}_i$ and sensor dimensions W and H, we aggregate the features that are relevant for reconstructing the local light field around each ray. For all $W \times H$ pixels from C_i we cast a set of rays $\mathcal{R}_i$ into the scene using a pinhole camera model. Each $\boldsymbol{r}_j \in \mathcal{R}$ is defined by its origin $\boldsymbol{o}_j$ and viewing direction $\boldsymbol{d}_j$.

Local Point Selection. The local point cloud encoding can explain the scene properties at their sparse locations. Explicitly representing high-frequency light field details from all views would necessitate a dense descriptor. Instead, we implicitly interpolate a representation descriptor for each ray. The work of DeVries et al. [7] shows that the interpolation of local latent descriptors allows for implicit scene representations for large indoor scenes. Unlike their regular grid structure, we want to leverage the information given

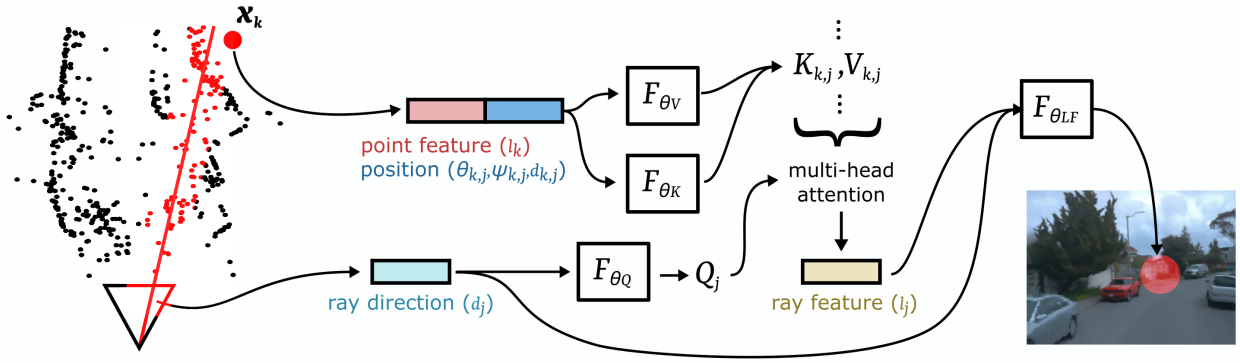


Figure 2: Neural Point Light Field Rendering Pipeline. For each ray $\boldsymbol{r}_j$, a set of K closest points is selected from a point cloud of the scene. From each point $\boldsymbol{x}_k$, a feature vector $\boldsymbol{l}_k$ and the relative location with respect to $\boldsymbol{r}_j$ predict a key and value vectors. The most relevant point features are aggregated for the ray with a multi-head attention module, using the encoded ray direction $\boldsymbol{d}_j$ to form the query vector. A light field function $F_{\theta LF}$ computes the ray color given the ray feature $\boldsymbol{l}_j$ and ray direction $\boldsymbol{d}_j$.

through the geometric properties of the point cloud. We assume that point features $\boldsymbol{l}_k$ hold enough information not only to represent the light field at their exact location, but also in their neighbourhood. For each ray $\boldsymbol{r}_j$, we aggregate a descriptor from a relevant set of sparse points. To this end, we select a set of K points $P_{j,i} \subset \mathcal{P}_i$ inside the viewing frustum of the camera C_i, with the smallest orthogonal distance $d_{k,j}$ between the points and the ray:

$$\cos\left(\varphi_{k,j}\right) = \boldsymbol{d}_{j,i} \cdot \left(\frac{\boldsymbol{x}_{k,i} - \boldsymbol{o}_{j,i}}{||\boldsymbol{x}_{k,i} - \boldsymbol{o}_{j,i}||} \right), \tag{1}$$

$$\begin{aligned} d_{k,j} &= \sin\left(\varphi_{k,j}\right) \cdot \left(\boldsymbol{x}_{k,i} - \boldsymbol{o}_{j,i}\right) \\ \text{with } \sin\left(\varphi_{k,j}\right) &= \sqrt{1 - \cos^2\left(\varphi_{k,j}\right)}. \end{aligned} \tag{2}$$

The ray origin $\boldsymbol{o}_{j,i}$, normalized ray direction $\boldsymbol{d}_{j,i}$, and point $\boldsymbol{x}_{k,i}$ are all given in a local reference frame centered in the captured P_i. A light field descriptor is then generated for each ray, considering all encoded features on the points in $P_{j,i}$.

Ray-centric Point Encoding. There are several immediate choices for the point embeddings of $P_{j,i}$, including average pooling, max pooling or a linear weighting by the distance $d_{j,k}$ of the selected K point features. However, these interpolation methods are ambiguous, i.e. they can deliver the same descriptor for various rays and features on the same set of closest points $P_{i,j}$. In order to ensure a consistent and unique description for each ray from the set $P_{j,i}$, we must use the unambiguous relative position of all points with respect to that ray, with coherence across different time steps i of the same scene.

As illustrated in Fig. 3 and formalized in Eq. 2, 4 and 5, we parameterize a close point using the angle $\theta_{k,j}$ between $\boldsymbol{x}_k$ and ray $\boldsymbol{d}_j$, the orthogonal distance between the point and ray, and the angle ψ, defined as the radial coordinate of a projected $\boldsymbol{x}_k$ onto a plane determined by a projection of the global $\boldsymbol{Y}$-axis and it's cross product with the ray direction $\boldsymbol{d}_j$:

$$\begin{aligned} \begin{bmatrix} x \\ y \end{bmatrix}_{k,j,proj} &= \begin{bmatrix} \boldsymbol{y}_j^T \\ \left(\boldsymbol{d}_j \times \boldsymbol{y}_j\right)^T \end{bmatrix} \boldsymbol{x}_k, \\ \text{with } \boldsymbol{y}_j &= \frac{\boldsymbol{Y} - \left(\boldsymbol{Y} \cdot \boldsymbol{d}_j\right) \boldsymbol{d}_j}{\left\| \boldsymbol{Y} - \left(\boldsymbol{Y} \cdot \boldsymbol{d}_j\right) \boldsymbol{d}_j \right\|} \text{ and } \boldsymbol{y} \in \mathbb{R}^3, \end{aligned} \tag{3}$$

$$\psi_{k,j} = \arctan \frac{x_{k,j,proj}}{y_{k,j,proj}}. \tag{4}$$

The angle between the global point $\boldsymbol{x}_k$ and $\boldsymbol{d}_j$ is computed as

$$\theta_{k,j} = \arccos\left(\boldsymbol{d}_{j,i} \cdot \frac{\boldsymbol{x}_k}{||\boldsymbol{x}_k||} \right). \tag{5}$$

This is computed in world coordinates, independently of local position, unlike $\varphi_{k,j}$ in Eq. 1, that is used for computing the distance.

Ray Feature Attention. Instead of applying an arbitrary weighting for the ray features, we propose a learned multi-head attention module (depicted in Fig. 4) to compute ray feature vector $\boldsymbol{l}_j$. We propose a variant of the multi-head attention module presented by Vaswani et al. [45]. We compare the chosen attention based weighting with linear interpolation schemes In the experimental Sec. 4. The two angular distances $\theta_{k,j}$ and $\psi_{k,j}$, as well as $d_{k,j}$ are transformed using a positional encoding $\gamma\left(s\right) = [..., \sin\left(2^t \pi s\right), \cos\left(2^t \pi s\right), ...]$ with $t = 0, \ldots, T$ and $T = 4$ [24, 43] to interpolate high frequency data from a low frequency input domain. The point feature vectors $\boldsymbol{l}_k$ and the positional encoded distances are concatenated to form a unique descriptor $\boldsymbol{v}_{k,j} = \left(\boldsymbol{l}_k \oplus \gamma\left(\theta_{k,j}\right) \oplus \gamma\left(\psi_{k,j}\right) \oplus \gamma\left(d_{k,j}\right)\right)$ corresponding to ray $\boldsymbol{r}_j$ and point k, that encompasses the positional encoding and the feature vector of that point. The descriptor $\boldsymbol{v}_{k,j}$ is then passed through two double-layer MLPs that predict a key $K_{k,j}$ and value $V_{k,j}$ for each of the K point ray pairs.

$$V_{k,j} = \boldsymbol{F}_{\theta_V}\left(\boldsymbol{v}_{k,j}\right), K_{k,j} = \boldsymbol{F}_{\theta_K}\left(\boldsymbol{v}_{k,j}\right) \tag{6}$$

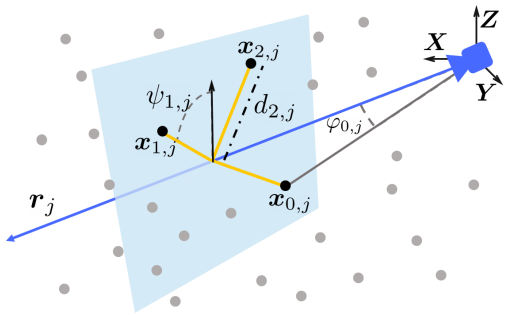


Figure 3: Ray-point distances are illustrated for a ray j and the $k = 3$ closest points. For better visualization, the ray and points are translated with $-\boldsymbol{o}_j$ into the scenes coordinate frame, and all points are projected into a single plane instead of 3 parallel planes.

$$Q_j = \boldsymbol{F}_{\theta_Q}\left(\gamma\left(\boldsymbol{d}_j\right)\right) \tag{7}$$

Query vector Q_j is derived from the positionally encoded ray direction $\gamma\left(\boldsymbol{d}_j\right)$. Ray direction $\boldsymbol{d}_j$ is again presented in world coordinates, to make it independent of any local reference coordinate system. The multi-head attention learns to predict a weight for all $V_{k,j}$ given $K_{k,j}$, for each selected point ray pair (k, j) and query ray Q_j. The aggregated output of the multi-head attention module comprises a feature code $\boldsymbol{l}_j \in \mathcal{R}^{128}$, that describes the light fields for each ray $\boldsymbol{r}_j$:

$$\text{multi-head attention: } \boldsymbol{l}_j = \boldsymbol{F}_{\theta_{attn}}(Q_j, K_{k,j}, V_{k,j}). \tag{8}$$

Points Beyond the Point Cloud. Point clouds in most automotive datasets only capture the scene geometry from the ground plane up to a few meters height. This results in scene regions which are not explicitly captured in the point cloud data, such as high building structures and the sky. We therefore set a threshold d_∞ below which we consider rays to intersect with the point cloud. The value d_∞ is chosen as the maximum distance between two points in any P_i after ignoring outlying points. For points that exceed d_∞, we concatenate $\boldsymbol{v}_{k,j}$ with a learned global feature code $\boldsymbol{l}_\infty$, such that the attention module can leverage both a global and a local point feature representation, as point features may contain relevant context and geometry for structures that rise above the point cloud, and may therefore still be useful.

3.3. RGB Prediction

After predicting a feature vector $\boldsymbol{l}_j$ for any ray $\boldsymbol{r}_j$ from encodings on a sparse point cloud, we are finally able to reconstruct the color $\boldsymbol{C}_j$ corresponding to any arbitrary ray in our global scene, that is

$$\boldsymbol{C}_j = \boldsymbol{F}_{\theta_{LF}}\left(\boldsymbol{d}_j \oplus \boldsymbol{l}_j\right). \tag{9}$$

Here $\boldsymbol{F}_{\theta_{LF}}$ is an 8-layer MLP (with 256 channels) that takes the concatenation of ray direction $\boldsymbol{d}_j$ and feature vector $\boldsymbol{l}_j$ corresponding to the ray at index j, to predict output color $\boldsymbol{C}_j$. Implementation details for this and all other modules are provided in the supplementary materials.

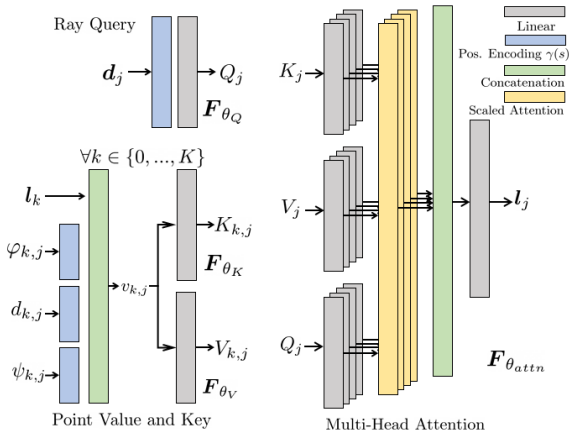


Figure 4: The multi-headed self-attention module aggregates the feature vector $\boldsymbol{l}_j$ of ray j given the ray direction $\boldsymbol{d}_j$ from the information of the K closest points. For each point k an embedding $\boldsymbol{v}_{k,j}$ is computed from the point's feature and the positionally encoded location relative to $\boldsymbol{r}_j$ for each ray-point pair (j, k). $\boldsymbol{F}_{\boldsymbol{\theta_K}}$ and $\boldsymbol{F}_{\boldsymbol{\theta_V}}$ compute the key K and value V vectors from $\boldsymbol{v}_{k,j}$. The query vector Q is predicted for the ray's direction $\boldsymbol{d}_j$.

For each predicted ray color $\hat{C}\left(\boldsymbol{r_j}\right)$ we can compute the mean-squared error image loss

$$\mathcal{L} = \sum_{j \in \mathcal{R}} \left\| \hat{C}\left(\boldsymbol{r_j}\right) - C\left(\boldsymbol{r_j}\right) \right\|_2^2. \tag{10}$$

Training All model parameters, namely $\theta_{ResNet18}$, θ_K, θ_V, θ_Q, θ_{attn} and θ_{LF}, are jointly optimized by minimizing the loss in Eq. 10 using the Adam optimizer [16] with a linear learning rate decay, where at each step we randomly sample 8192 rays from a small batch of frames.

4. Assessment

To assess the proposed method and evaluate its complexity, we train neural point light fields on an automotive driving dataset. We compare against state-of-the-art neural rendering methods by generating novel views interpolating between poses on the driven trajectory, as well as extrapolating to completely new trajectories. Moreover, we analyze how architecture and parameter choices in the proposed method affect reconstruction quality.

4.1. Complexity

Volumetric neural rendering methods require a large number of samples per ray for obtaining accurate results. Even though existing methods allow speeding up rendering times [15], training often requires hundreds of ray samples. We report the measured time and evaluations count corresponding to processing a single ray during training and inference in Tab. 1. To ignore differences related to specific

Figure 5: Scene Reconstruction. We present results for reconstructing images for poses seen during training of NeRF [23], DS-NeRF [6], GSN [7] and Neural Point Light Fields. All methods were trained on the same set of scenes from the Waymo Open Dataset [42]. NeRF (even with substantially increased model capacity) and DS-NeRF show similar blurriness and other artifacts, while the depth supervision allows DS-NeRF to improve over existing methods. GSN produces fewer artifacts while struggling to reconstruct fine details, and fails for sparsely observed views (center scene). Neural Point Light Fields most faithfully reconstructs the image from the data set, see also Tab. 2.

implementation speed-ups (such as rays pre-caching), evaluation time is measured after the ray sampling step for a respective PyTorch [30] implementation of the method. Measured times include encoding and decoding steps (e.g., point encoding in our method or convolution refinement in GSN), normalized by the number of image pixels to correspond to a single ray.

In contrast to volumetric scene representations, that need a high number of sampling points, even when supported by local feature vectors, Neural Point Light Fields only require a single evaluation per ray during rendering. This leads to a two times speedup, despite the overhead incured due to extraction of point features.

Cost		NeRF [24]	DS-NeRF [6]	GSN [7]	Ours
No. of Evaluations	↓	192	192	64	**1**
Time per ray, training (in μs)	↓	146	146	37	**34**
Time per ray, inference (in μs)	↓	49	49	17	**10**

Table 1: Complexity per ray during training and inference. All volumetric approaches require multiple evaluations per ray. Neural Point Light Fields (Ours) has a complexity of $O(1)$ per rendered ray. Despite an added complexity in the feature extraction step, this allows for shorter training and inference.

4.2. Experimental Setup

We quantitatively and qualitatively validate the proposed method on two tasks, namely view reconstruction and novel view synthesis, where we compare against Generative Scene Networks (GSN), NeRF and depth-supervised NeRF (DS-NeRF). GSN has been successfully applied to large scale indoor scenes [7] and takes advantage of a local embedding of the scene that is jointly learned with the scene. In contrast to our sparse point features, the latent codes are located on a sparse 2D floorplan. We evaluate NeRF [24] as a state-of-the-art volumetric scene representation. Additionally we evaluate DS-NeRF [6], which takes advantage of an additional depth supervision for the opacity prediction. In the Supplementary Document, we present additional comparisons to NeRF++ [51] and Free View Synthesis [33], which employs features on a mesh proxy geometry as discussed in Sec. 2. All methods were trained with their official publicly available code, by choosing the configuration closest to our outdoor/free moving scene scenario. For our method we use a maximum of $N = 20000$ randomly sampled points, $K = 8$ closest points, 128 dimensional point and ray embedding $\boldsymbol{l}_k$ and $\boldsymbol{l}_j$, and 8 heads in the multi-head attention module.

All methods except GSN were trained on 6 scenes from the Waymo Open Dataset [42] with a length ≤ 200 frames, see Supplemental Document. To allow training on a single GPU, we downsample the captured images by a factor of 8, resulting in a resolution of 240×160 pixels. For GSN, a convolutional refinement step requires the models to be trained on the full image resolution, and the code provided hard-coded settings that required us (after consulting with the authors) to use 64×64 image crops. For a fair evaluation, we report GSN results for 3 scenes, while calculating metrics on downsampled dataset images. Note that GSN has an advantage in all quantitative evaluations as a smaller FOV at lower resolution needs to be synthesized. All models were trained until convergence on each scene on a mixture of NVIDIA TITAN Xp and NVIDIA V100

Figure 6: Novel View Interpolation. We predict views for unseen poses held-out from the training data. images in middle row are taken from the longest selected scenes (200 frames), while the rest are taken from shorter ones (80 frames). NeRF and DS-NeRF show blurry and overly smooth results, but perform better on smaller scenes. NeRF synthesizes the details on the small scenes better, while failing completely on larger scenes, even when substantially increasing the model's capacity. GSN performs consistently across all scenes, but exhibits artifacts and lacks detail. Our Neural Point Light Fields representation allows high-quality synthesis for novel view interpolation.

	NeRF [24]	DS-NeRF [6]	GSN [7]	Ours
Reconstruction				
PSNR ↑	<u>29.48</u>	26.53	17.98	**31.52**
SSIM ↑	<u>0.815</u>	0.778	0.512	**0.882**
LPIPS ↓	0.289	0.306	<u>0.136</u>	**0.110**
Novel View Synthesis				
PSNR ↑	22.47	<u>26.15</u>	16.83	**29.96**
SSIM ↑	0.700	<u>0.772</u>	0.464	**0.868**
LPIPS ↓	0.389	0.310	<u>0.174</u>	**0.119**

Table 2: We report PSNR, SSIM and LPIPS on 5 static scenes from the Waymo Open Dataset [42] using images from the front camera for NeRF [24], DS-NeRF [6], GSN [7] and Neural Point Light Fields. For PSNR and SSIM, higher is better; for LPIPS lower is better. The best values are emphasized in **bold**, while the next best are <u>underlined</u>. Our method outperforms all methods in all metrics. While NeRF shows only slightly worse reconstruction performance, DS-NeRF provides better novel view synthesis capabilities.

GPUs. Complexity evaluations were computed on the same hardware. The lower resolution requirements on GSN for over-fitting on a single scene resulted in a training time of 2 days, while the other models trained for 2 to 3 days, depending on the number of scene frames.

Quantitative Evaluation. We train all methods using the same 90% of all driven trajectory frames, leaving the remaining 10% for evaluating interpolation of unseen views within the observed trajectory. Tab. 2 reports quantitative results for both tasks using the PSNR, SSIM [46] and LPIPS [52] metrics. GSN makes an overall worse impression than the other methods in both tasks. The proposed method outperforms all other methods in all metrics. While NeRF performs significantly worse in the Novel View Synthesis task, DS-NeRF exhibit only a slight performance drop compared to its reconstruction results, probably benefiting from a better opacity prediction when trained on a sparse set of images. Our method performs the best in the view synthesis task as well, exhibiting only a minor performance degradation compared to the reconstruction task, in contrast to NeRFs results.

Scene Reconstruction. The results shown in Fig. 5 support the quantitative evaluation from Tab. 2. While NeRF produces inconsistent and blurry predictions for the large scenes we address in this work, it is still able to recover some details on straight scenes. We hypothesize that the blurriness arises from the requirements of an accurate pose information and the sparse set of training views on long scene trajectories. DS-NeRF shows a similar behavior, but lacks some detail that has been reconstructed in NeRF, while producing smooth artifacts. Renderings of the depth map of the trained scene suggest that the point cloud capture is too smooth for DS-NeRF representation and, as such, suppresses high frequency features. In contrast, GSN produces an overall consistent reconstruction, independent of scene length. Nevertheless results show smoothing even in the significantly downsampled resolution accepted by GSN. In contrast, Neural Point Fields allows reconstructing all structures, independent of their position and appearance across frames resulting on only few artifacts on very fine structures (e.g. individual tree branches, leafs). Please also

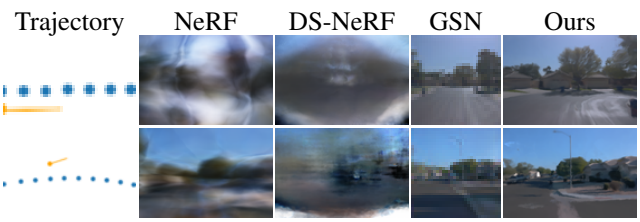


Figure 7: Novel View Trajectory Extrapolation. Extrapolating views (orange) using the training trajectory (blue). While NeRF and DS-NeRF fail to synthesize views far from the training trajectory, the proposed method produces high quality results, similar to its performance in the reconstruction and view interpolation tasks.

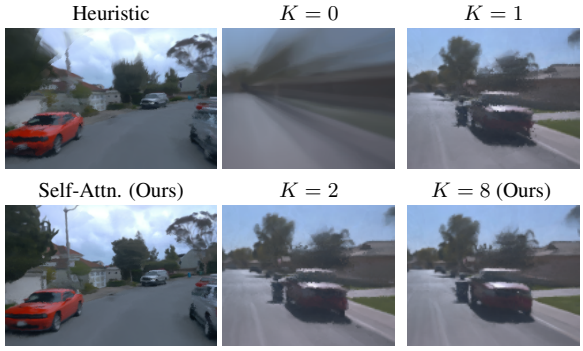
see the video in the Supplementary Materials.

Novel View Trajectory Interpolation. We next compare views synthesized for frames excluded from the training data in Fig. 6. DS-NeRF suffers from blur and ghosting in the interpolation task. NeRF shows similar, though slightly weaker artifacts on the few scenes it was able to converge on. Our method produces high quality renderings when handling both short (top and bottom rows) and long (middle row) scenes. The results validate that these existing methods are not able to effectively synthesize scenes just from a sparse set of images. GSN, which uses local support, seems to be more consistent, producing similar output quality in both tasks, regardless of scene length. Neural Point Light Fields encode the scenes features on a sparse set of points, hence achieve high-quality novel view interpolation, even for long sequences.

Novel View Trajectory Extrapolation. The results shown in Fig. 7 report visual extrapolation experiments. We present a map of the novel view camera poses with respect to the training trajectory. Our method is able to generate a set of novel trajectories and scenes, that can hardly be differentiated from the interpolation and reconstruction results. This is possible within certain regions of the scene which are at least partially covered by the training images see Supplemental Material. Views into scene regions which were not seen during training, e.g., the back of a vehicle only seen from the front, result in imaginary objects, probably hallucinated from points similar to the observed objects. Incorporating information from additional cameras (possibly covering 360°) may allow synthesizing such occluded scene regions in the future.

4.3. Ablations

We analyze architecture and parameter choices in Fig. 8. Choosing self-attention for aggregating ray features proves to be crucial, as we find that a heuristic weighting or naive summation over all point features are not able to achieve similar results. While merely summing prohibits training completely, heuristic weighting of each point feature by the inverse distance $d_{k,j}$ achieves better results. However, this



	Naive Sum	Heuristic	$K = 0$	$K = 1$	$K = 2$	Ours
PSNR ↑	4.84	24.56	18.88	29.83	30.95	**31.52**

Figure 8: Ablation studies. Qualitative and quantitative comparisons of using different numbers K of closest points per ray and different feature aggregation approaches.

weighting still lacks details and suffers from artifacts and noisy scene reconstruction. We propose to index a set of points, in contrast to methods that purely parameterize a ray. In addition, we compare between using a different number of points K per ray in Fig. 8, indicating that a substantial number of points is essential for learning large scene light fields. Additional ablation studies are reported in the Supplementary Materials.

5. Conclusion

We introduce an implicit representation that encodes a local light field on a point cloud. Departing from volumetric representations that require querying radiance estimates at hundreds of sample points along each ray, we learn realistic radiance fields with only a single radiance sample per ray. Neural point light fields are functions of the ray direction and local point feature neighborhood, which allows us to interpolate the light field conditioned training images without densely captured input views. As such, the method allows for novel view synthesis in large-scale automotive scenarios, with only a few sparse view directions available during a drive-by capture. We validate the proposed method for novel view synthesis when interpolating and extrapolating along unseen trajectories, where existing implicit representation methods fail. While it is typical in automotive scenarios to have point cloud captures available, in the future we plan to jointly recover point positions and local features of the proposed neural point light fields.

Acknowledgements. We thank ServiceNow for providing compute resources for this project with the ServiceNow Toolkit. Felix Heide was supported by an NSF CAREER Award (2047359), a Sony Young Faculty Award, and a Project X Innovation Award. Yuval Bahat was supported by the MSCA COFUND STAR fellowship.

References

[1] Edward H Adelson, James R Bergen, et al. *The plenoptic function and the elements of early vision*, volume 2. Vision and Modeling Group, Media Laboratory, Massachusetts Institute of , 1991.

[2] Sameer Agarwal, Yasutaka Furukawa, Noah Snavely, Ian Simon, Brian Curless, Steven M. Seitz, and Richard Szeliski. Building rome in a day. *Commun. ACM*, 54(10):105112, Oct 2011.

[3] Kara-Ali Aliev, Artem Sevastopolsky, Maria Kolos, Dmitry Ulyanov, and Victor Lempitsky. Neural point-based graphics. pages 696–712, 2020.

[4] Relja Arandjelović and Andrew Zisserman. Nerf in detail: Learning to sample for view synthesis. *arXiv preprint arXiv:2106.05264*, 2021.

[5] Mark Boss, Raphael Braun, Varun Jampani, Jonathan T Barron, Ce Liu, and Hendrik Lensch. Nerd: Neural reflectance decomposition from image collections. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12684–12694, 2021.

[6] Kangle Deng, Andrew Liu, Jun-Yan Zhu, and Deva Ramanan. Depth-supervised nerf: Fewer views and faster training for free. *arXiv preprint arXiv:2107.02791*, 2021.

[7] Terrance DeVries, Miguel Angel Bautista, Nitish Srivastava, Graham W Taylor, and Joshua M Susskind. Unconstrained scene generation with locally conditioned radiance fields. *arXiv preprint arXiv:2104.00670*, 2021.

[8] John Flynn, Michael Broxton, Paul Debevec, Matthew DuVall, Graham Fyffe, Ryan Overbeck, Noah Snavely, and Richard Tucker. Deepview: View synthesis with learned gradient descent. *Proceedings IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun 2019.

[9] Stephan J Garbin, Marek Kowalski, Matthew Johnson, Jamie Shotton, and Julien Valentin. Fastnerf: High-fidelity neural rendering at 200fps. *arXiv preprint arXiv:2103.10380*, 2021.

[10] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[11] Ankit Goyal, Hei Law, Bowei Liu, Alejandro Newell, and Jia Deng. Revisiting point cloud shape classification with a simple and effective baseline. *arXiv preprint arXiv:2106.05304*, 2021.

[12] Abdullah Hamdi, Silvio Giancola, and Bernard Ghanem. Mvtn: Multi-view transformation network for 3d shape recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1–11, 2021.

[13] Zekun Hao, Arun Mallya, Serge Belongie, and Ming-Yu Liu. Gancraft: Unsupervised 3d neural rendering of minecraft worlds. *arXiv preprint arXiv:2104.07659*, 2021.

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[15] Peter Hedman, Pratul P Srinivasan, Ben Mildenhall, Jonathan T Barron, and Paul Debevec. Baking neural radiance fields for real-time view synthesis. *arXiv preprint arXiv:2103.14645*, 2021.

[16] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2015.

[17] Georgios Kopanas, Julien Philip, Thomas Leimkühler, and George Drettakis. Point-based neural rendering with per-view optimization. In *Computer Graphics Forum*, volume 40, pages 29–43. Wiley Online Library, 2021.

[18] Marc Levoy and Pat Hanrahan. Light field rendering. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 31–42, 1996.

[19] Lingjie Liu, Jiatao Gu, Kyaw Zaw Lin, Tat-Seng Chua, and Christian Theobalt. Neural sparse voxel fields. *arXiv preprint arXiv:2007.11571*, 2020.

[20] Stephen Lombardi, Tomas Simon, Jason Saragih, Gabriel Schwartz, Andreas Lehrmann, and Yaser Sheikh. Neural volumes: Learning dynamic renderable volumes from images. *ACM Trans. Graph.*, 38(4):65:1–65:14, Jul 2019.

[21] Erika Lu, Forrester Cole, Tali Dekel, Weidi Xie, Andrew Zisserman, David Salesin, William T. Freeman, and Michael Rubinstein. Layered neural rendering for retiming people in video. *ACM Trans. Graph.*, 39(6), nov 2020.

[22] Ricardo Martin-Brualla, Noha Radwan, Mehdi S. M. Sajjadi, Jonathan T. Barron, Alexey Dosovitskiy, and Daniel Duckworth. NeRF in the Wild: Neural Radiance Fields for Unconstrained Photo Collections. In *arXiv*, 2020.

[23] Ben Mildenhall, Pratul P. Srinivasan, Rodrigo Ortiz-Cayon, Nima Khademi Kalantari, Ravi Ramamoorthi, Ren Ng, and Abhishek Kar. Local light field fusion: Practical view synthesis with prescriptive sampling guidelines. *ACM Transactions on Graphics (TOG)*, 2019.

[24] 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 *Proceedings of the IEEE European Conf. on Computer Vision (ECCV)*, pages 405–421. Springer, 2020.

[25] Thomas Neff, Pascal Stadlbauer, Mathias Parger, Andreas Kurz, Joerg H. Mueller, Chakravarty R. Alla Chaitanya, Anton S. Kaplanyan, and Markus Steinberger. DONeRF: Towards Real-Time Rendering of Compact Neural Radiance Fields using Depth Oracle Networks. *Computer Graphics Forum*, 40(4), 2021.

[26] Michael Niemeyer and Andreas Geiger. Giraffe: Representing scenes as compositional generative neural feature fields. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[27] Michael Niemeyer, Lars Mescheder, Michael Oechsle, and Andreas Geiger. Differentiable volumetric rendering: Learning implicit 3d representations without 3d supervision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[28] Julian Ost, Fahim Mannan, Nils Thuerey, Julian Knodt, and Felix Heide. Neural scene graphs for dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2856–2865, 2021.

[29] Keunhong Park, Utkarsh Sinha, Jonathan T. Barron, Sofien Bouaziz, Dan B Goldman, Steven M. Seitz, and Ricardo

Martin-Brualla. Nerfies: Deformable neural radiance fields. *Proceedings of the IEEE International Conference on Computer Vision*, 2021.

[30] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[31] Francesco Pittaluga, Sanjeev J Koppal, Sing Bing Kang, and Sudipta N Sinha. Revealing scenes by inverting structure from motion reconstructions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 145–154, 2019.

[32] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas 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.

[33] Gernot Riegler and Vladlen Koltun. Free view synthesis. In *Proceedings of the IEEE European Conf. on Computer Vision (ECCV)*, pages 623–640. Springer, 2020.

[34] Gernot Riegler and Vladlen Koltun. Stable view synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12216–12225, 2021.

[35] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-Motion Revisited. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[36] Johannes Lutz Schönberger, Enliang Zheng, Marc Pollefeys, and Jan-Michael Frahm. Pixelwise View Selection for Unstructured Multi-View Stereo. In *Proceedings of the IEEE European Conf. on Computer Vision (ECCV)*, 2016.

[37] Katja Schwarz, Yiyi Liao, Michael Niemeyer, and Andreas Geiger. Graf: Generative radiance fields for 3d-aware image synthesis. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[38] Vincent Sitzmann, Justus Thies, Felix Heide, Matthias Nießner, Gordon Wetzstein, and Michael Zollhöfer. Deepvoxels: Learning persistent 3d feature embeddings. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[39] Vincent Sitzmann, Michael Zollhöfer, and Gordon Wetzstein. Scene representation networks: Continuous 3d-structure-aware neural scene representations. In *Advances in Neural Information Processing Systems*, 2019.

[40] Pratul P Srinivasan, Boyang Deng, Xiuming Zhang, Matthew Tancik, Ben Mildenhall, and Jonathan T Barron. Nerv: Neural reflectance and visibility fields for relighting and view synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7495–7504, 2021.

[41] P. P. Srinivasan, R. Tucker, J. T. Barron, R. Ramamoorthi, R. Ng, and N. Snavely. Pushing the boundaries of view extrapolation with multiplane images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 175–184, 2019.

[42] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin 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.

[43] Matthew Tancik, Pratul P. Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan T. Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. *Advances in Neural Information Processing Systems*, 2020.

[44] Justus Thies, Michael Zollhfer, and Matthias Niener. Deferred neural rendering. *ACM Transactions on Graphics*, 38(4):112, Jul 2019.

[45] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[46] Z. Wang, E.P. Simoncelli, and A.C. Bovik. Multiscale structural similarity for image quality assessment. In *The Thrity-Seventh Asilomar Conference on Signals, Systems Computers, 2003*, volume 2, pages 1398–1402 Vol.2, 2003.

[47] Sven Wanner and Bastian Goldluecke. Globally consistent depth labeling of 4d light fields. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 41–48. IEEE, 2012.

[48] Sven Wanner and Bastian Goldluecke. Variational light field analysis for disparity estimation and super-resolution. *IEEE transactions on pattern analysis and machine intelligence*, 36(3):606–619, 2013.

[49] Wenqi Xian, Jia-Bin Huang, Johannes Kopf, and Changil Kim. Space-time neural irradiance fields for free-viewpoint video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9421–9431, 2021.

[50] Zhenpei Yang, Yuning Chai, Dragomir Anguelov, Yin Zhou, Pei Sun, Dumitru Erhan, Sean Rafferty, and Henrik Kretzschmar. Surfelgan: Synthesizing realistic sensor data for autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11118–11127, 2020.

[51] Kai Zhang, Gernot Riegler, Noah Snavely, and Vladlen Koltun. Nerf++: Analyzing and improving neural radiance fields. *arXiv preprint arXiv:2010.07492*, 2020.

[52] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, pages 586–595, 2018.

[53] 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. *arXiv preprint arXiv:2106.01970*, 2021.

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