Learning Generalizable Light Field Networks from Few Images
Qian Li, Franck Multon, Adnane Boukhayma

To cite this version:

HAL Id: hal-04116795
https://inria.hal.science/hal-04116795
Submitted on 5 Jun 2023

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution 4.0 International License
LEARNING GENERALIZABLE LIGHT FIELD NETWORKS FROM FEW IMAGES

Qian Li, Franck Multon, Adnane Boukhayma
Inria, Univ. Rennes, CNRS, IRISA, M2S, France

Fig. 1: Our method enables fast generation of novel views from sparse input images without 3D supervision in training. We generate above novel views for objects (ShapeNet dataset) and a scene (DTU dataset) never seen at training.

ABSTRACT
We explore a new strategy for few-shot novel view synthesis based on a neural light field representation. Given a target camera pose, an implicit neural network maps each ray to its target pixel’s color directly. The network is conditioned on local ray features generated by coarse volumetric rendering from an explicit 3D feature volume. This volume is built from the input images using a 3D ConvNet. Our method achieves competitive performances on real MVS data with respect to state-of-the-art neural radiance field based competition, while offering a roughly 50 times faster rendering.

Index Terms— Novel view synthesis, neural light field, volumetric rendering

1. INTRODUCTION
The ongoing research in computer vision and artificial intelligence has long sought to enable machines to understand 3D given limited observations [1–6]. This ability is in fact crucial for many downstream 3D based machine learning, vision and graphics tasks. Among these, novel view synthesis is a particularly prominent problem with numerous applications in free viewpoint and virtual reality, as well as image editing and manipulation.

While most traditional approaches require depth information, coarse geometric proxies or dense samplings of the input views, deep learning based approaches rely on deep neural network’s generalization abilities across view points and 3D scenes to achieve novel view synthesis from minimal visual input. In this context, the recently popularized implicit neural representations offer numerous advantages in modelling 3D shape [1] and appearance [2, 4, 7] in comparison to their traditional alternatives. In particular, Neural Radiance Fields [2] (NeRF), notably their generalizable versions (e.g. [5, 7]), provide impressive novel view synthesis performances. However, the rendering of these methods requires sampling hundreds of points along each target pixel ray, and evaluating densities and view-dependent colors for all these points through a multi-layer perceptron (MLP), which increases the time and memory requirements.

To reduce this complexity, we propose to use an implicit neural network operating in ray space rather than the 5D Euclidean × direction space, thus alleviating the need for per ray multi-point evaluation and physical rendering. For a given target pixel, an MLP (i.e. light field network) maps its ray coordinate and ray features to the color directly. Key to efficient generalization, and differently from [4], we build the ray features by computing and merging 3D convolutional feature volumes from the input images. These features are then rendered volumetrically into a coarse ray feature image, as illustrated in figure 2.

Our method is trained end-to-end and evaluated using real multi-view stereo data (DTU [8]). We achieve competitive results in comparison to generalizable encoder-decoder NeRF models, while providing orders of magnitude faster rendering (see table 3).

2. RELATED WORK
We discuss existing work that is most relevant to few-shot novel view synthesis in this section.

Early deep learning based approaches used 2D convolutional encoder-decoder architectures mapping the sparse inputs to the target images [9–11]. These methods were outperformed by 3D aware convolutional approaches [12–14]. Although many of these could learn to generate 360-degree views from very sparse inputs especially for synthetic central object data, most of them could not scale to high resolution images, complex scenes, and real data such as MVS datasets (DTU [8]).

Implicit neural radiance fields (NeRF) [2] emerged later on as a powerful representation for novel view synthesis. It presented initially however a few limitations such as compu-
3. METHOD

Given one or few images \( \{ I_i \} \) of a scene or an object with their known camera parameters, i.e., camera poses \( \{ R_i, T_i \} \), \( R_i \in SO(3) \), \( T_i \in \mathbb{R}^3 \), and intrinsics \( K \in \mathbb{R}^{3 \times 3} \), our goal is to generate images \( \{ I_i \} \) for novel target views, i.e., new camera poses \( \{ R_t, T_t \} \). A summary of our method is illustrated in Figure 2. We present in the remaining of this section the components of the two stages of our method, namely the convolutional stage, and the neural light field network.

3.1. Feature volume re-sampling

Following seminal work (e.g., [5, 13]), we build an explicit volume of features from an input image \( I_i \) using a fully convolutional neural network \( E \) consisting of a succession of a 2D convolutional U-Net and several 3D convolutional blocks:

\[
F_i = E(I_i),
\]

where \( I_i \in \mathbb{R}^{H \times W \times 3} \), \( H \) and \( W \) being the height and width of the input RGB image, and \( F_i \in \mathbb{R}^{H_V \times W_V \times D \times C} \), \( H_V \), \( W_V \), \( D \) and \( C \) being respectively the height, width, depth, and the number of channels of the 3D feature volume.

Using the the input feature volume \( F_i \) aligned with the input image, we would like to create a feature volume \( F_{t/i} \), aligned to the target image, that could be used subsequently to render a target feature image given the target camera pose \( \{ R_t, T_t \} \). Following the principles of volumetric rendering [2], in order to recreate a target image of dimensions \( H_V \times W_V \), we need to evaluate \( N \) points \( \{ p_{u,v}^k \}_{k=1}^N \) along each ray \( r_{u,v} \) with direction \( d_{u,v} \), where \( u \in [1, H_V] \) and \( v \in [1, W_V] \):

\[
d_{u,v} = R_t K^{-1} \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} + T_t, \quad p_{u,v}^z = T_t + t_z \frac{d_{u,v}}{|d_{u,v}|},
\]

where \( t_z \sim \mathcal{U} [z_n + \frac{1}{z_f} (z_f - z_n), z_n + \frac{1}{z_f} (z_f - z_n)] \) following [2], \( z_n \) and \( z_f \) being the depth near and far bounds of the visual frustum. \( K \) is the intrinsic camera matrix. The target volume \( F_{t/i} \) is obtained then as the resampling of input volume \( F_i \) with trilinear interpolation, using points \( \{ p_{u,v}^z \} \) aligned rigidly to the input camera coordinate frame:

\[
F_{t/i}(u, v, z) = F_i(R_i^T (p_{u,v}^z - T_i)),
\]

where \( F_{t/i} \in \mathbb{R}^{H_V \times W_V \times N \times C} \) and \( \{ R_t, T_t \} \) is the input camera pose. In practice, we normalize the aligned points’ coordinates prior to sampling as \( F_i \) is assumed to represent features in the input view normalized device coordinate (NDC) space.

3.2. Feature Aggregation and rendering

As different input views provide different information about the observed scene, we merge subsequently the 3D features obtained from the various inputs. We note that all target feature volumes \( \{ F_{t/i}^k \} \) provided by input images \( \{ I_i^k \} \) are represented in the same target view camera coordinate frame. Inspired by attention mechanisms, we propose to learn a 3D confidence measure per input view in the form of a weight volume \( W_i \in \mathbb{R}^{H_V \times W_V \times D} \). This volume is obtained as one of the channels of the input volume features \( W_i = F_i(1) \) (i.e., \( W_{t/i} = F_{t/i}(1) \)). After resampling the input features \( \{ F_{t/i}^k \} \) into the target ones \( \{ F_{t/i}^k \} \), we use the resampled weights \( \{ W_{t/i}^k \} \) normalized with Softmax across the input views to...
compute a weighted average of the target volumes:

\[ F_t = \sum_k \text{Softmax}(W^{k}_{t/i}F^{k}_{t/i}(\|1, C\|)), \]

where index \( k \) is over the number of input views, and \( F \in \mathbb{R}^{H \times W \times N \times C-1} \). This aggregation allows our method to use an arbitrary number of input views at both training and testing.

Following volumetric rendering \([2]\), we generate a target feature image \( \bar{F} \) for a given target view differentiably using \( \alpha \)-compositing of the target feature volume \( F_t \) along the depth dimension. We assume one of the target feature channels to represent volume density \( \sigma = F_t(1) \in \mathbb{R}^{H \times W \times D} \). We recall that the dimensions of tensor \( F_t \) span the pixels of the target feature resolution \( H_x \times W_x \) in the first two dimensions, and \( N \) points sampled along each ray for the third dimension. The rendered target feature image then writes:

\[ \bar{F} = \sum_{z=1}^{N} T_z \alpha_z F_t(\|1, C - 1\|), \quad (5) \]
\[ T_z = e^{-\sum_{j=1}^{z-1} \sigma(j) \delta_j} \quad \alpha_z = 1 - e^{\sigma(z) \delta_z} \quad (6) \]

where \( T \) represents transmittance, \( \delta_z = t_{z+1} - t_z \) and \( \bar{F} \in \mathbb{R}^{H \times W \times C - 2} \). In order to reduce the memory cost and increase the rendering speed of our method, the size of the rendered feature image is chosen to be lower than the size of the target image resolution, i.e. \( H_V = H/4 \) and \( W_V = W/4 \).

### 3.3. Neural Light Field

The convolutional rendered features produce a low resolution feature image representative of all rays making up the target view. We propose to learn a light field function \( f \) to upsample and refine these first stage results.

Given a ray \( r_{u,v} \) with direction \( d_{u,v} \) corresponding to the target image pixel coordinates \( (u, v) \), with \( (u, v) \in [1, H] \times [1, W] \), we encode rays using Plücker coordinates similarly to Sitzmann et al. \([4]\):

\[ r_{u,v} = \frac{(d_{u,v}, T_t \times d_{u,v})}{||d_{u,v}||}, \quad (7) \]

where \( r_{u,v} \in \mathbb{R}^6 \). This representation ensures a unique ray encoding when the origin \( T_t \) moves along direction \( d_{u,v} \). We recall that the expression of \( d_{u,v} \) as a function of the target camera pose \( \{R_t, T_t\} \) can be found in equation 2.

The feature \( F_{u,v} \) of a ray \( r_{u,v} \) at the final image resolution \( H \times W \) is obtained from the lower resolution rendered feature image \( \bar{F} \in \mathbb{R}^{H \times W \times C - 2} \) through a learned upsampling. Specifically, the rendered feature image undergoes two successive 2D convolutions and up samplings to produce a feature image at the desired resolution \( \bar{F} \in \mathbb{R}^{W \times H \times C - 2} \). The final target RGB image \( I_t = \{c_{u,v}\}_{u \in [1, H], v \in [1, W]} \) is predicted from the concatenation of the ray coordinate and its feature with an MLP accordingly:

\[ c_{u,v} = f(r_{u,v}, F_{u,v}), \quad (8) \]

Notice that while convolution equipped NeRF \([2]\) methods (e.g. \([5, 7]\)) require querying \( H \times W \times N \) 3D points through their implicit neural radiance fields, our light field network only needs to evaluate \( H \times W \) rays, which enables our method to train potentially faster, and render orders of magnitude faster compared to \([5, 7]\) (see Table 3).

### 3.4. Training Objective

Our model is fully differentiable and trained end-to-end. We optimize the parameters of the convolutional network \( E \) and the light field network \( f \) jointly, by back-propagating a combination of a fine loss \( L_r \) and two coarse losses \( \tilde{L}_r \) and \( \tilde{L}_d \):

\[ L = L_r + \tilde{L}_r + \tilde{L}_d, \quad (9) \]

\( L_r \) and \( \tilde{L}_r \) are the L2 reconstruction losses of the final light field predicted image \( I_t \) and the first stage prediction \( \tilde{I}_t \) respectively:

\[ L_r = ||I_t - \tilde{I}_r||^2_2, \quad \tilde{L}_r = ||\tilde{I}_t - \tilde{I}_r||^2_2. \quad (10) \]

We additionally regularize the gradient of the low resolution depth image \( \tilde{d}_t \) rendered from the density volume \( \sigma \) of the first stage thusly:

\[ \tilde{L}_d = \frac{1}{H \times W} \sum_{u,v} \|\partial_u \tilde{d}_t|e^{-||\partial_u \tilde{I}_r||}| + \|\partial_v \tilde{d}_t|e^{-||\partial_v \tilde{I}_r||}|, \quad (11) \]

\[ \tilde{d}_t = \frac{1}{\sum_{z=1}^{N} T_z \alpha_z t_z} \sum_{z=1}^{N} T_z \alpha_z t_z, \quad (12) \]

where \( T \) and \( \alpha \) are detailed in equation 6.

### 4. EXPERIMENTS

#### 4.1. Implementation details

We implemented our method with the PyTorch framework on a Quadro RTX 5000 gpus. We optimize with the Adam solver using learning rate \( 10^{-4} \) in training and \( 10^{-5} \) in fine-tuning. The depth of the convolutional feature volume is set to \( D = 32 \), and the number of channels \( C = 32 \).

#### 4.2. Comparison on DTU dataset

We demonstrate the capability of our method to generate novel views from sparse input views using the DTU benchmark \([8]\). Following the PixelNeRF\([7]\) experimental settings, the data is split into 88 training scenes and 16 testing scenes. Each scene contains 49 views, including 4 views for testing.
as suggested by MVSNeRF[5] and GeoNeRF[22]. Our train-
ing does not require mask supervision, thus all evaluation are performed on full resolution image(400 × 300) rather than only foreground.

For quantitative comparison, we report the peak signal-to-
to-noise ratio (PSNR), structural similarity (SSIM) and learned perceptual image patch similarity (LPIPS) reconstruction metrics in Table 1 for 3 and 6 view inputs averaged across all testing scenes. We report numbers of PixelNeRF(PN) and MVSNeRF(MN) from RegNeRF[23]. We also show qualitative comparisons for 6 view inputs in figure 3. While our method is robust and competitive with NeRF based counter-
parts, it seems to lack some high frequency details. We defer
this limitation to future work.

<table>
<thead>
<tr>
<th>Method</th>
<th>PSNR↑</th>
<th>SSIM↑</th>
<th>LPIPS↓</th>
</tr>
</thead>
<tbody>
<tr>
<td>PN[7]</td>
<td>18.74</td>
<td>21.02</td>
<td>0.618</td>
</tr>
<tr>
<td>MN[5]</td>
<td>16.33</td>
<td>18.26</td>
<td>0.602</td>
</tr>
<tr>
<td>Ours</td>
<td>19.86</td>
<td>21.36</td>
<td>0.657</td>
</tr>
</tbody>
</table>

Table 1: Quantitative comparison of reconstructed images in the DTU [8] dataset without test time optimization.

4.3. Per-scene fine-tune results

Table 2 shows a quantitative comparison of our method with the recent few-shot novel view synthesis state-of-the-art with test time optimization. We outperform all methods in the PSNR and SSIM metrics, including conditional baseline PixelNeRF(PN)[7] and MVSNeRF(MN)[5], and unconditional baselines DietNeRF(DN)[15] and RegNeRF(RN)[23]. Figure 4 shows a qualitative comparison to MVSNeRF and PixelNeRF with 6 input views after finetuning. We obtain overall comparable performances with generalizable methods [5, 7]. We recall again that competition methods here require renderings that are orders of magnitude slower than ours.

<table>
<thead>
<tr>
<th>Method</th>
<th>PSNR↑</th>
<th>SSIM↑</th>
<th>LPIPS↓</th>
</tr>
</thead>
<tbody>
<tr>
<td>PN[7]</td>
<td>17.33</td>
<td>21.52</td>
<td>0.548</td>
</tr>
<tr>
<td>MN[5]</td>
<td>16.26</td>
<td>18.22</td>
<td>0.601</td>
</tr>
<tr>
<td>DN[15]</td>
<td>10.01</td>
<td>18.70</td>
<td>0.354</td>
</tr>
<tr>
<td>RN[23]</td>
<td>15.33</td>
<td>19.10</td>
<td>0.621</td>
</tr>
<tr>
<td>Ours</td>
<td>20.72</td>
<td>22.60</td>
<td>0.677</td>
</tr>
</tbody>
</table>

Table 2: Quantitative comparison of reconstructed images in the DTU [8] dataset with test time optimization.

4.4. Rendering time comparison

As shown in table 3, compared with PixelNeRF[7] and MVS-
NeRF [5], our method requires less inference time on DTU dataset with 3 input views.

<table>
<thead>
<tr>
<th>Method</th>
<th>PSNR↑</th>
<th>SSIM↑</th>
<th>LPIPS↓</th>
</tr>
</thead>
<tbody>
<tr>
<td>PN[7]</td>
<td>17.33</td>
<td>21.52</td>
<td>0.548</td>
</tr>
<tr>
<td>MN[5]</td>
<td>16.26</td>
<td>18.22</td>
<td>0.601</td>
</tr>
<tr>
<td>Ours</td>
<td>19.86</td>
<td>21.36</td>
<td>0.657</td>
</tr>
</tbody>
</table>

Table 3: Comparison of rendering complexity.

4.5. Ablation

We propose an ablative analysis showing the importance of the light field stage in our method. Specifically, we disable the latter (ours w/o lf), and we render the final image directly from the target view aligned convolutional feature volume. Table 4 shows numerical comparisons for 3 and 6 input views on DTU [8], and figure 5 shows qualitative comparisons for 6 input views.

<table>
<thead>
<tr>
<th>Method</th>
<th>PSNR↑</th>
<th>SSIM↑</th>
<th>LPIPS↓</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ours w/o lf</td>
<td>18.21</td>
<td>19.55</td>
<td>0.582</td>
</tr>
<tr>
<td>Ours</td>
<td>19.86</td>
<td>21.36</td>
<td>0.657</td>
</tr>
</tbody>
</table>

Table 4: Quantitative ablation on DTU dataset[8].

5. CONCLUSIONS

We proposed a method for generating novel views from few input calibrated images with a single forward pass prediction deep neural network. We learn an implicit neural light field function that models ray colors directly. In comparison to [4], we proposed a more efficient local ray conditioning, and an optimization free inference. Our method outperforms the baselines and provides competitive performances compared to locally conditioned radiance fields (e.g. [5, 7]), while being roughly 50 times faster at rendering.
References


