Depth-Guided Robust and Fast Point Cloud Fusion NeRF for Sparse Input Views

Shuai Guo\textsuperscript{1}, Qiuwen Wang\textsuperscript{1}, Yijie Gao\textsuperscript{1}, Rong Xie\textsuperscript{1}, Li Song\textsuperscript{1,2}\textsuperscript{*}

\textsuperscript{1}Institute of Image Communication and Network Engineering, Shanghai Jiao Tong University, Shanghai, China
\textsuperscript{2}Cooperative Medianet Innovation Center, Shanghai Jiao Tong University, Shanghai, China
\{shuaiguo, wangqiuwen, gaoyijie, xierong, song_li\}@sjtu.edu.cn

Abstract

Novel-view synthesis with sparse input views is important for real-world applications like AR/VR and autonomous driving. Recent methods have integrated depth information into NeRFs for sparse input synthesis, leveraging depth prior for geometric and spatial understanding. However, most existing works tend to overlook inaccuracies within depth maps and have low time efficiency. To address these issues, we propose a depth-guided robust and fast point cloud fusion NeRF for sparse inputs. We perceive radiance fields as an explicit voxel grid of features. A point cloud is constructed for each input view, characterized within the voxel grid using matrices and vectors. We accumulate the point cloud of each input view to construct the fused point cloud of the entire scene. Each voxel determines its density and appearance by referring to the point cloud of the entire scene. Through point cloud fusion and voxel grid fine-tuning, inaccuracies in depth values are refined or substituted by those from other views. Moreover, our method can achieve faster reconstruction and greater compactness through effective vector-matrix decomposition. Experimental results underline the superior performance and time efficiency of our approach compared to state-of-the-art baselines.

Introduction

Novel-view synthesis (NVS) serves as a fundamental objective within the realm of computer vision. The recent surge in NVS popularity is largely attributable to the success of Neural Radiance Fields (NeRFs)(Mildenhall et al. 2021). However, NeRFs generally demand numerous images taken from a variety of views for efficient training. In real-world applications such as AR/VR and autonomous driving, where input views are typically sparse (Niemeyer et al. 2022), NeRF risks overfitting. This may lead to inconsistencies in reconstructions or failure in generating any useful solution.

Various strategies have significantly enhanced the performance of NeRF for sparse inputs by 1) optimizing training data utilization (Yu et al. 2021; Niemeyer et al. 2022; Truong et al. 2023), 2) incorporating prior information like depth and flow (Deng et al. 2022; Roessle et al. 2022), or 3) exploring new constraints and regularizations (Kim, Seo, and Han 2022; Yang, Pavone, and Wang 2023). Among these solutions, depth prior has garnered substantial interest due to its ease of access, its capacity to offer object positions, and its aid in handling occlusions and geometry understanding. Numerous methods have been proposed to integrate depth information into NeRFs for sparse input views.

Nevertheless, most existing depth-aware NeRFs for sparse input views disregard the holes, artifacts, and inaccurate values of depth maps. For example, DSNeRF (Deng et al. 2022) introduced depth supervision to leverage depth information, but it failed to account for inaccurate depth values. Certain practices might introduce additional unreliability. DDP-NeRF (Roessle et al. 2022), for instance, used a depth completion network to transform sparse depth into dense depth maps and uncertainty estimates, potentially leading to more inaccurate depth values. The issues inherent to the network itself could subsequently impact sparse view synthesis effects. SparseNeRF (Guangcong et al. 2023) employed the rough point cloud geometry provided by sparse RGB-D inputs to render more images and depict the approx-
Table 1: Comparison of NeRF variants. The training time (min) and model size (MB) are summarized for various methods. The first row lists the methods, and the subsequent rows correspond to different datasets.

<table>
<thead>
<tr>
<th>Method</th>
<th>Training Time (min)</th>
<th>Model Size (MB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSNeRF</td>
<td>10^5</td>
<td>2000</td>
</tr>
<tr>
<td>PlenOctrees</td>
<td>10^5</td>
<td>1750</td>
</tr>
<tr>
<td>SNeRG</td>
<td>10^5</td>
<td>1500</td>
</tr>
<tr>
<td>Plenovels</td>
<td>10^5</td>
<td>1250</td>
</tr>
<tr>
<td>DVGO</td>
<td>10^5</td>
<td>1000</td>
</tr>
<tr>
<td>ViP-NeRF</td>
<td>10^5</td>
<td>750</td>
</tr>
<tr>
<td>Ours</td>
<td>10^5</td>
<td>500</td>
</tr>
<tr>
<td>RegNeRF</td>
<td>10^5</td>
<td>250</td>
</tr>
</tbody>
</table>

In this section, we provide a comprehensive review of the relevant literature in the areas of novel-view synthesis and sparse input NeRF.

### Related Work

#### Novel-View Synthesis

The body of work about novel-view synthesis can generally be divided into two main categories: explicit representation based synthesis and implicit representation based synthesis.

**Explicit Representations**

Explicit representation methods commonly employ point clouds (Ran, Liu, and Wang 2022; Huang et al. 2023), voxels (Sitzmann et al. 2019; Zhou et al. 2018), or animations (Zhao et al. 2019; Yu et al. 2022). Models that rely on explicit representations are limited to the specific data points used for training and may struggle with certain types of scenes or lighting conditions.

**Implicit Representations**

Implicit methods directly model the appearance of a 3D scene, thus eliminating the need for explicit geometric representation. A prime example of this approach is NeRF (Mildenhall et al. 2021). NeRF has been widely adopted in recent studies for numerous extensions, including but not limited to, video synthesis (Li et al. 2022a,b), relighting (Yu et al. 2022; Rudnev et al. 2022), and scene editing (Yuan et al. 2022; Kobayashi, Matsumoto, and Sitzmann 2022).

#### Sparse Input NeRF

A number of works have been proposed to address the data-hungry problem of NeRF by exploiting training data, incorporating prior information like depth and flow, or introducing new constraints and regularizations.
**Exploiting Training Data**  
PixelNeRF (Yu et al. 2021) enhances scene comprehension by conditioning a NeRF on image inputs using a fully convolutional approach. RegNeRF (Niemeyer et al. 2022) applies regularization to the geometry and appearance of rendered patches from unseen views, helping to rectify inaccurately optimized scene geometry and divergent behavior at the optimization outset. SPARF (Truong et al. 2023) employs pixel matches between input views and depth consistency to generate realistic novel-view renderings with sparse inputs.

**Leveraging Prior Information**  
DSNeRF (Deng et al. 2022), for example, uses the sparse depth information created by COLMAP (Schoenberger and Frahm 2016) as explicit supervision for sparse view synthesis. SparseNeRF (Guangcong et al. 2023), meanwhile, relies on depth ranking prior and spatial continuity distillation on NeRF’s, enabling the synthesis of novel views with sparse view inputs. DDPNeRF (Roßle et al. 2022) synthesizes novel views of entire rooms from significantly fewer images by employing dense depth priors to constrain the NeRF optimization, thereby enabling data-efficient novel-view synthesis on challenging indoor scenes.

**Introducing New Regularizations**  
InfoNeRF (Kim, Seo, and Han 2022) enhances the compactness of reconstructed scenes along individual rays while ensuring consistency across neighboring rays, an approach particularly beneficial for few-shot novel-view synthesis. Drawing on the importance of frequency, FreeNeRF (Yang, Pavone, and Wang 2023) regulates the frequency range of NeRF’s inputs. It also imposes penalties on the density fields near the camera. Both methods demonstrate innovative approaches to manipulating input parameters to optimize output with sparse views.

---

**Method**

We first revisit the feature grids and radiance field, followed by an analysis of factorizing radiance fields and point cloud representation. Subsequently, we illustrate the process of continuous field representation, point cloud fusion, and rendering. The presentation concludes with a discussion on the optimization and loss function. Figure 3 is an overview of our method.

**Feature Grids and Radiance Field Revisited**

We construct a model of a radiance field which establishes a relationship between a 3D location \( x \) and a viewing direction \( d \), with its volume density \( \sigma \), and a view-dependent color \( c \). Taking inspiration from TensoRF (Chen et al. 2022), we employ a standard 3D grid \( G \in \mathbb{R}^{I \times J \times K} \). Each voxel within this grid is equipped with multi-channel features, thus allowing us to simulate this function. Here, \( I, J, \) and \( K \) represent the resolutions of the feature grid along the \( X, Y, \) and \( Z \) axes, respectively. We segregate these feature channels into two distinct grids, one for geometry, represented by \( G_\sigma \in \mathbb{R}^{I \times J \times K \times P} \), and another for appearance, represented by \( G_c \in \mathbb{R}^{I \times J \times K \times P} \). In this context, \( P \) indicates the number of appearance feature channels. These individual grids are designed to separately model the volume density \( \sigma \), and the view-dependent color \( c \).

Our model accommodates a range of appearance features within the appearance grid \( G_c \), which depend on a predefined function \( S \). This function transforms an appearance feature vector in combination with a viewing direction \( d \), into a color \( c \). Here \( S \) is a small MLP where the appearance grid \( G_c \) comprises neural features and spherical harmonics (SH) coefficients, respectively. On the other hand, we introduce a single-channel grid \( G_\sigma \), where the values directly rep-
resent volume density, thus eliminating the need for an extra conversion function. This results in a continuous grid-based radiance field which can be expressed by the equation:

$$\sigma_c = G_\sigma(x), S(G_\sigma(x), d).$$ (1)

In this equation, $G_\sigma(x)$ and $G_c(x)$ represent the features from the two grids at the location $x$, interpolated trilinearly.

**Factorizing Radiance Fields**

We model the geometry grid $G_\sigma$ and the appearance grid $G_c$ as factorized tensors. Utilizing the Vector-Matrix (VM) decomposition, we factorize the 3D geometry tensor $G_\sigma$ as:

$$G_\sigma = \sum_{r=1}^{n} A_{c,r}^X \circ b_{3r-2} + A_{c,r}^Y \circ b_{3r-1} + A_{c,r}^Z \circ b_{3r},$$ (2)

where $v_{c,r}^X, v_{c,r}^Y, v_{c,r}^Z \in \mathbb{R}^I, v_{c,r}^X, v_{c,r}^Y, v_{c,r}^Z \in \mathbb{R}^J$, $M_{c,r}^XY \in \mathbb{R}^{I \times J}$, $M_{c,r}^YZ \in \mathbb{R}^{J \times K}$, and $M_{c,r}^XZ \in \mathbb{R}^{K \times I}$.

Our approach differs from TensoRF in that the number of components of the 3D geometry tensor in our method is consistently fixed as the number of input views, denoted as $n$. Each component corresponds to an input view and is used to represent the volume density of the point cloud, constructed by this view. Similarly, the appearance tensor $G_c$ is modeled using comparable vector-matrix spatial factors and additional feature basis vectors $b_r$, which express a multi-channel voxel feature grid:

$$G_c = \sum_{r=1}^{n} A_{c,r}^X \circ b_{3r-2} + A_{c,r}^Y \circ b_{3r-1} + A_{c,r}^Z \circ b_{3r},$$

where $v_{c,r}^X, v_{c,r}^Y, v_{c,r}^Z \in \mathbb{R}^I, v_{c,r}^X, v_{c,r}^Y, v_{c,r}^Z \in \mathbb{R}^J$, $M_{c,r}^XY \in \mathbb{R}^{I \times J}$, $M_{c,r}^YZ \in \mathbb{R}^{J \times K}$, and $M_{c,r}^XZ \in \mathbb{R}^{K \times I}$.

In this context, $v_{c,r}^X, v_{c,r}^Y, v_{c,r}^Z, m_{c,r}^X, m_{c,r}^Y, m_{c,r}^Z \in \mathbb{R}^n$, and $M_{c,r}^XY, M_{c,r}^YZ, M_{c,r}^XZ \in \mathbb{R}^{n \times n}$.

**Point Cloud Representation**

We construct a point cloud for each input view, which is represented within the voxel grid using the corresponding feature vectors and matrices. Initially, we map depth values to the normalized device coordinate (NDC) space to ensure that all visible locations are normalized and represented within a predetermined cubic space. Subsequently, we map all pixels of the input view to the 3D space to generate the point cloud. For a 2D pixel location, $x_p$, in the $r$-th input image, we map it to a 3D world location $x_w = (x, y, z)$, using the equation:

$$x_w = R^{-1}(K^{-1}x_pD - t),$$ (6)

where $K, R$, and $t$ denote the intrinsic parameters, rotation matrix, and translation matrix of the input view corresponding to $x_p$.

Next, we represent the point cloud by constructing its geometry grid $G_\sigma$ and appearance grid $G_c$. As the 3D space is represented by a standard 3D grid $G \in \mathbb{R}^{I \times J \times K}$, the world location $x_w = (x, y, z)$ is projected onto the matrix $M_{c,r}^XY, M_{c,r}^YZ, M_{c,r}^XZ$ and vectors $v_{c,r}^X, v_{c,r}^Y, v_{c,r}^Z$. Elements of the matrices and vectors that are projections of the point cloud are assigned a value of 1, while all other elements are assigned a value of 0. This assignment method effectively indicates the presence of points in the geometry grid.

For the appearance grid, elements of the matrix $M_{c,r}^XY, M_{c,r}^YZ, M_{c,r}^XZ$ are assigned the average of R, G, B color values of the points projected onto this element. Meanwhile, vectors $v_{c,r}^X, v_{c,r}^Y, v_{c,r}^Z$ are assigned random values. This provides a rough representation of the point clouds. After the training step, the representation of point clouds will be refined to overcome inaccurate depth values and will be fused together to characterize the entire scene.

**Continuous Field Representation**

We employ trilinear interpolation to represent a continuous field. For instance, consider a component tensor represented as $A_r^X = v_{r}^X \circ M_{c,r}^YZ$. Each tensor element within this can be described as $A_r^X = v_{r}^X \circ M_{c,r}^YZ$. The interpolated values can then be calculated using:

$$A_r^X(x) = v_{r}^X(x)M_{c,r}^YZ(y, z).$$ (7)

In the equation above, $v_{r}^X(x)$ signifies the trilinearly interpolated value at the 3D location $x = (x, y, z)$ of $A_r$. The term $v_{r}^X(x)$ represents the linear interpolation at position $x$ along the X-axis. Meanwhile, $M_{c,r}^YZ(y, z)$ denotes the bilinear interpolation at $(y, z)$ of $M_{c,r}^YZ$ in the YZ plane. Similarly, the following relations hold:

$$A_r^Y(x) = v_{r}^Y(y)M_{c,r}^XZ(x, z),$$

$$A_r^Z(x) = v_{r}^Z(z)M_{c,r}^XY(x, y).$$ (8)
By trilinearly interpolating both grids and merging the point cloud, we obtain:

\[
G_\sigma(x) = \sum_r \sum_m A_{\sigma,r}^m(x), \quad (9)
\]

\[
G_c(x) = B \oplus ([A_{c,r}^m(x)]_{m,r}). \quad (10)
\]

**Point Cloud Fusion and Rendering**

Integrating equations (1), (9) and (10), the factorized tensorial radiance field for the fused point cloud in our model is articulated as:

\[
\sigma, c = \sum_r \sum_m A_{\sigma,r}^m(x), S(B \oplus ([A_{c,r}^m(x)]_{m,r}), d). \quad (11)
\]

To render images, we march along a ray, and \( Q \) shading points are sampled along each ray. The color of the pixel is then determined using:

\[
C = \sum_{q=1}^Q \tau_q (1 - \exp(-\sigma_q \Delta q)) c_q,
\]

\[
\tau_q = \exp(-\sum_{p=1}^{q-1} \sigma_p \Delta p).
\]

Here, \( \sigma_q \) and \( c_q \) denote the density and color respectively, determined by our model at their specific sampled locations \( s_q \). Meanwhile, \( \Delta q \) is defined as the step size of the ray and \( \tau_q \) stands for transmittance.

To further improve quality and avoid local minima, we apply coarse-to-fine reconstruction. Similar to TensoRF (Chen et al. 2022), our coarse-to-fine reconstruction is simply achieved by linearly and bilinearly upsampling our XYZ-mode vector and matrix factors.

**Optimization**

The network parameters are optimized using a collection of RGB-D frames, each containing color, depth, and camera pose data. Our loss function consists of two primary components. The first component is an RGB loss function. This component involves an L2 rendering loss combined with additional regularization terms to optimize our tensor factors for radiance field reconstruction. It can be represented as:

\[
\mathcal{L}_{\text{RGB}} = ||C - \tilde{C}||_2^2 + \omega_{\text{reg}} \mathcal{L}_{\text{reg}}. \quad (13)
\]

Here, \( \tilde{C} \) denotes the ground truth color, \( C \) represents the predicted color, \( \mathcal{L}_{\text{reg}} \) is an L1 regularization term, and \( \omega_{\text{reg}} \) is the weight assigned to this regularization. To promote sparsity in our tensor factors’ parameters, we use the standard L1 regularization. This technique has proven effective in enhancing the extrapolation of views and eliminating anomalies like floaters or outliers in the final renderings.

The second component is a depth loss function, in which we employ an L2 rendering loss and can be expressed as:

\[
\mathcal{L}_{\text{depth}} = ||D - \tilde{D}||_2^2. \quad (14)
\]

In this equation, \( \tilde{D} \) is the ground truth depth, while \( D \) stands for the predicted depth. Contrary to the approach in

In summary, the composite loss function for our model, denoted by \( \mathcal{L} \), is articulated as:

\[
\mathcal{L} = \mathcal{L}_{\text{RGB}} + \lambda_{\text{depth}} \mathcal{L}_{\text{depth}}, \quad (15)
\]

where \( \lambda_{\text{depth}} \) serves as a hyperparameter to strike a balance between color and depth supervision. Figure 4 illustrates how the inaccuracies are addressed through point cloud fusion and radiance field optimization.

**Datasets**

Our results are derived from the real-world multi-view datasets LLFF (Mildenhall et al. 2019) and DTU (Jensen et al. 2014). The LLFF dataset includes 8 forward-facing scenes, each having a varied count of frames, all presented at a spatial resolution of 4032×3024. The DTU dataset features various objects captured from multiple perspectives in a controlled indoor setting.

For every scene, we selected subsets containing 2, 3, or 4 training views for our evaluation. Depth maps were generated using COLMAP (Schonberger and Frahm 2016), adhering to the dataset-provided camera parameters.

**Baselines**

We compare our method against several state-of-the-art models, including InfoNeRF (Kim, Seo, and Han 2022), DietNeRF (Jain, Tancik, and Abbeel 2021), RegNeRF (Niemeyer et al. 2022), DSNeRF (Deng et al. 2022), DDP-NeRF (Roessle et al. 2022), and ViP-NeRF (Somraj and Soundararajan 2023). It is noteworthy that all results of these methods are obtained through publicly accessible codes or papers.
Table 1: Quantitative Comparisons on The LLFF Dataset.

<table>
<thead>
<tr>
<th>Method</th>
<th>2 views</th>
<th>3 views</th>
<th>4 views</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PSNR↑</td>
<td>SSIM↑</td>
<td>LPIPS↓</td>
</tr>
<tr>
<td>InfoNeRF</td>
<td>9.23</td>
<td>0.2095</td>
<td>0.7761</td>
</tr>
<tr>
<td>DietNeRF</td>
<td>11.89</td>
<td>0.3209</td>
<td>0.7265</td>
</tr>
<tr>
<td>RegNeRF</td>
<td>16.90</td>
<td>0.4872</td>
<td>0.4402</td>
</tr>
<tr>
<td>DSNeRF</td>
<td>17.06</td>
<td>0.5068</td>
<td>0.4548</td>
</tr>
<tr>
<td>DDP-NeRF</td>
<td>17.21</td>
<td>0.5377</td>
<td>0.4223</td>
</tr>
<tr>
<td>ViP-NeRF</td>
<td>16.76</td>
<td>0.5222</td>
<td>0.4017</td>
</tr>
<tr>
<td>Our Method</td>
<td>17.83</td>
<td>0.5512</td>
<td>0.3832</td>
</tr>
</tbody>
</table>

Figure 5: Qualitative comparisons on the LLFF dataset with two input views. Notably, the predictions from DSNeRF and DDP-NeRF exhibit noticeable floater artifacts. RegNeRF struggles to capture the finer details in bone structures. In contrast, our method significantly reduces these imperfections. In the second and fourth examples, we highlight the color changes predicted by DDP-NeRF. Our model’s predictions are free from the aforementioned artifacts.

Implementation Details

Our implementation was carried out in PyTorch (Paszke et al. 2019), excluding any customized CUDA kernels. This model was optimized over $T$ iterations, with a batch size of 4096 pixel rays, executed on a single NVIDIA RTX 4090 GPU (24GB). We introduced a feature decoding MLP and set $P = 27$. To facilitate a stepwise transition from coarse-to-fine reconstruction, we initiated with a grid of $N_0^3$, where $N_0 = 128$. This grid was upsampled at intervals of 2000, 3000, 4000, 5500, and 7000 steps, with voxel counts transitioning linearly in logarithmic space from $N_0^3$ to $N^3$.

Comparisons

Comparisons on The LLFF Dataset

The qualitative and quantitative comparisons with competing models are respectively showcased in the referenced Table 1 and Figure 5. Here, we note that predictions from other models frequently exhibit blurriness, especially for views substantially distanced from the input. In contrast, our method performs well in geometry predictions, generating more realistic novel views. Our method consistently delivers sharp outputs accompanied by precise scene geometry across all tested scenarios. A standout feature of our approach is its ability to accurately represent and recover intricate details. Additionally, our model is good at eliminating visual noise, ensuring clearer visuals around objects in comparison to the baseline models.

Comparisons on The DTU Dataset

The qualitative comparisons between the competing models can be found in Figure 6. Our approach consistently surpasses other models, especially in the perceptual metric domain. Additionally, these models often present inconsistent appearances for novel views, particularly when the camera perspectives deviate significantly from the input views. We can see that our model aligns more closely with the ground truth and avoids many of the artifacts evident in the predictions of other models.
Table 2: Influence of depth quality on our method. In this table, we add 5% and 10% white noise to depth maps respectively to observe the performance of our method. The original depth maps are obtained with COLMAP.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>0% Noise</th>
<th>5% Noise</th>
<th>10% Noise</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PSNR↑</td>
<td>SSIM↑</td>
<td>LPIPS↓</td>
</tr>
<tr>
<td>LLFF</td>
<td>17.83</td>
<td>0.5512</td>
<td>0.3832</td>
</tr>
<tr>
<td>DTU</td>
<td>19.34</td>
<td>0.7431</td>
<td>0.2576</td>
</tr>
</tbody>
</table>

Ablation Study

Ablation of Point Cloud Construction and Fusion

In Figure 7, we highlight the significance of our point cloud construction and fusion, which effectively addresses the challenges posed by inaccurate depth values. When the point cloud construction is not applied, all matrices and vectors default to random values. The adoption of our point cloud construction and fusion approach yields superior quantitative and qualitative outcomes.

Depth Quality

To delve deeper into the effects of varied depth quality on our method, we present synthesis outcomes as depth quality fluctuates in Table 2. For reference, the highest quality depth maps are sourced from COLMAP (Schonberger and Frahm 2016). We intentionally degrade the quality of these depth maps by converting 5% and 10% of the best depth values into white noise. Even with diminishing depth map quality, our method maintains commendable performance.

Discussion

With the depth supervision and our streamlined tensorial radiance field structure, our method boasts better performance and faster reconstruction.

To reduce the influence of inaccurate depth values, we use depth information to optimize the neural radiance field. We project points from each input view into 3D space, creating a unique point cloud for every view, represented by matrices and vectors. Although the initial representation of the point clouds is rudimentary and imprecise, the fusion and training processes refine inaccuracies in depth values or replace them with values from alternate views.

Conclusion

As shown in Figure 2, our method has less model size and less reconstruction time, as we effectively present the point cloud constructed for each input view with a few vectors and matrices. For example, for a 300 × 300 × 300 feature grid with P = 27 channels (plus one density channel), the total number of parameters in a dense grid is 756 M, while the number of parameters used for our method is only about 0.36 M (four views input). We can achieve a compression rate of about 0.05%.

Limitations and FutureWork

We believe that depth-guided radiance fields based on matrix and vector representations hold significant promise in time efficiency enhancement. Moving forward, we will aim to further leverage depth information and tensorial structures to improve the performance and efficiency of NeRF.
Acknowledgments

This work was supported by the Fundamental Research Funds for the Central Universities, National Key R&D Project of China (2019YFB1802701), MoE-China Mobile Research Fund Project (MCM20180702), STCSM under Grant 22DZ2229005, 111 project BP0719010.

References


