LoLep: Single-View View Synthesis with Locally-Learned Planes and Self-Attention Occlusion Inference

Cong Wang[1], Yu-Ping Wang[2], Dinesh Manocha[3]

[1]Tsinghua University, [2]The Beijing Institute of Technology, [3]University of Maryland

Abstract

We propose a novel method, **LoLep***, which regresses* **Lo***cally-***Le***arned* **p***lanes from* ***a single RGB image*** *to represent scenes accurately, thus generating better novel views. Without the depth information, regressing appropriate plane locations is a challenging problem. To solve this issue, we pre-partition the disparity space into bins and design a disparity sampler to regress local offsets for multiple planes in each bin. However, only using such a sampler makes the network not convergent; we further propose two optimizing strategies that combine with different disparity distributions of datasets and propose an occlusion-aware reprojection loss as a simple yet effective geometric supervision technique. We also introduce a self-attention mechanism to improve occlusion inference and present a Block-Sampling Self-Attention (BS-SA) module to address the problem of applying self-attention to large feature maps. We demonstrate the effectiveness of our approach and generate state-of-the-art results on different datasets. Compared to MINE, our approach has an LPIPS reduction of 4.8%∼9.0% and an RV reduction of 74.9%∼83.5%. We also evaluate the performance on real-world images and demonstrate the benefits.*

1. Introduction

Single-view view synthesis allows a camera to roam around a scene from a given photograph. It has been used to generate compelling views for different applications including image editing and augmented or virtual reality. The underlying techniques require understanding the geometry of scenes, reasoning about occlusions, and rendering high-quality images of novel views in real time.

Many approaches have been proposed to solve this problem [41, 35, 22, 26, 40]. They synthesize novel views by predicting a naive representation (e.g., depth maps, voxels, or point clouds) from a single image and generating images for novel views using appropriate rendering techniques. While these methods generate some positive results, they limit the performance of single-view view synthesis due to their inability to represent occluded regions well [37]. In this context, layered representations [37, 38, 45, 21, 20, 13] are more suitable for single-view view synthesis.

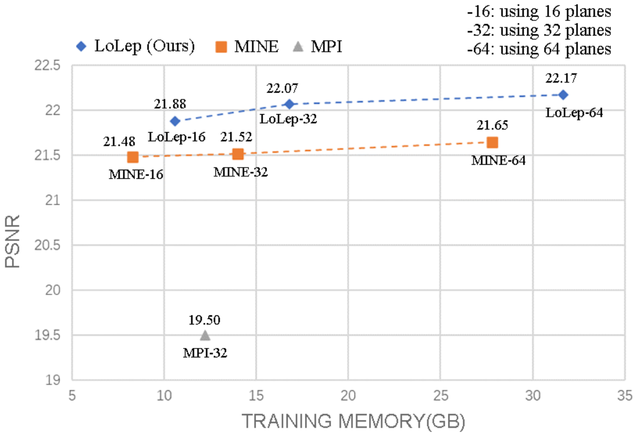


Figure 1. **Comparisons on the KITTI dataset.** LoLep generates state-of-the-art results and even LoLep with fewer planes uses less memory and generates better novel views than previous methods with more planes (LoLep-16 vs. MINE-32, MINE-64 and MPI-32, LoLep-32 vs. MINE-64), which benefits from locally-learned planes and self-attention occlusion inference. The batch size is 4.

Recently, Multiplane Image (MPI) [45] has gained popularity as a layered representation and has been used for single-view view synthesis [37]. Specifically, it is an encoder-decoder structure supervised by multiple images from different views of a given scene and is used to predict multiple planes of RGB and alpha values from a single image. MINE [20] combines MPI with NeRF [25] and generalizes MPI into a continuous depth MPI by considering multiple plane location inputs. This can improve the performance of single-view view synthesis to better infer geometric primitives in a scene. However, these methods sample plane locations randomly, which makes it hard for the planes to learn optimal scene representations. As a result,

Yu-Ping Wang is the corresponding author (wyp_cs@bit.edu.cn).

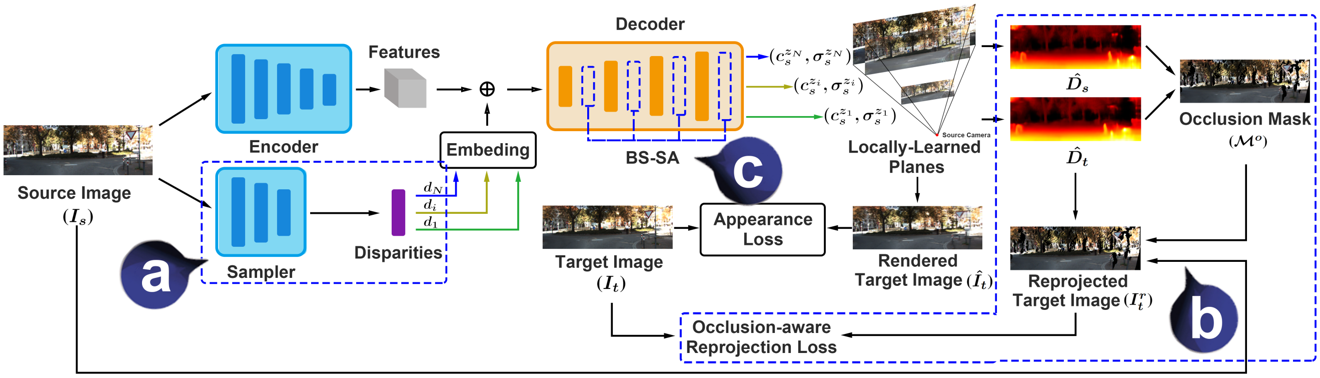

Decoder
Features
Encoder
Embeding
BS-SA
$(c_s^{z_N}, \sigma_s^{z_N})$
$(c_s^{z_i}, \sigma_s^{z_i})$
$(c_s^{z_1}, \sigma_s^{z_1})$
Source Camera
Locally-Learned Planes
$\hat{D}_s$
$\hat{D}_t$
Occlusion Mask ($\mathcal{M}^o$)
Source Image (I_s)
Sampler
d_N
d_i
d_1
Disparities
Target Image (I_t)
Appearance Loss
Rendered Target Image ($\hat{I}_t$)
Reprojected Target Image (I_t^r)
Occlusion-aware Reprojection Loss
a
b
c

Figure 2. **Overview.** LoLep regresses locally-learned planes to represent scenes accurately without a depth map input mainly relying on three novel components. **(a) Disparity Sampler:** regressing accurate locations for multiple planes from only the RGB image; **(b) Occlusion-aware Reprojection Loss:** a simple yet effective geometric supervision technique for single-view view synthesis to learn better geometry; **(c) Block-Sampling Self-Attention:** supporting self-attention applied to large feature maps for higher performance. '$\oplus$' concatenates two tensors.

these methods usually require more planes to obtain satisfactory novel views, requiring huge computing power. To alleviate this requirement, a key issue is *how to fully utilize the limited planes to obtain the most accurate scene representation as possible*.

Previous works [21, 13] solve this issue by regressing more accurate locations for multiple planes. Due to the lack of supervision and using globally-learned planes, however, their networks take an RGB image and an additional depth map as input. The depth map is provided by a pretrained depth prediction network, which introduces a heavy dependence on other networks.

Main Results: We present a novel single-view view synthesis method based on Multiplane Image, LoLep. LoLep aims to make full use of locally-learned planes to represent scenes accurately, thus generating better novel views from a single RGB image with less memory (Figure 1). In order to achieve that, we pre-partition the disparity space into bins and design a disparity sampler to condition local offsets of planes on a single RGB image. However, due to the lack of depth information, applying the sampler directly makes the network not convergent. We further propose two optimizing strategies that combine with different disparity distributions of datasets and an occlusion-aware reprojection loss to solve it (described in Section 4.1). To improve the ability for occlusion inference, we introduce a self-attention mechanism to our decoder and present a Block-Sampling Self-Attention (BS-SA) module to work for large feature maps (described in Section 4.2). Overall, the novel components of our approach include:

- We propose a novel single-view view synthesis method based on Multiplane Image, LoLep, that regresses accurate scene representations and generates better novel views on scene geometry and occluded regions.
- We introduce a self-attention mechanism to improve occlusion inference and present a BS-SA module to address the problem of applying self-attention on large feature maps.
- We compare with prior methods and show that LoLep outperforms MINE on different datasets with an LPIPS reduction of 4.8%∼9.0% and an RV reduction of 74.9%∼83.5%. Moreover, LoLep with fewer planes uses less memory and generates better results than prior methods with more planes.

2. Related Works

Multi-view View Synthesis. Multi-view view synthesis is a well-studied problem, and methods in this area generate images for novel views given a set of images from different views of the same scene. Some earlier methods are based on interpolating nearby views [19, 12, 4]. However, synthesizing novel views using interpolation techniques without a 3D representation causes inconsistency between different generated views. To alleviate this problem, many approaches based on depth maps [5, 28] and multi-view geometry [7, 8, 47, 18] have been proposed. Moreover, deep learning methods have also been applied to novel view synthesis [45, 15, 34, 6, 24, 1]. Some of these methods use deep neural networks to improve on traditional approaches, so they can be applied to more challenging scenarios. Recently, NeRF [25] techniques have been used to generate improved results for view synthesis. However, these techniques can only generate novel views of specific static scenes and involve intensive computation. Many techniques have also been proposed to improve the performance [36, 43, 23, 27, 2, 42]. Unlike these methods, we deal with the problem of single-view view synthesis, which only has one input image at test time.

Table 1. Symbols and Notation

I	image	d_f	far disparity	$(\cdot)_s$	an entity related to the source view or the source camera
D	depth map	d_n	near disparity	$(\cdot)_t$	an entity related to the target view or the target camera
d	disparity value	N	number of planes	$(\cdot)_i$	an entity related to the i-th plane
z	depth value	K	intrinsic matrix	$(\cdot)^{z_i}$	an entity related to the plane, whose depth is z_i
H	height	T	transformation matrix	θ_{ED}	parameters of the encoder-decoder
W	width	R	rotation matrix	θ_S	parameters of the disparity sampler
c	RGB values	t	translation vector	$\pi(\cdot)$	projecting a 3D camera coordinate to 2D pixel coordinate
σ	volume density values	$\hat{(\cdot)}$	predicted results	$\pi^{-1}(\cdot)$	the inverse operation of $\pi(\cdot)$

Single-view View Synthesis. Compared to multi-view view synthesis, single-view view synthesis is a more challenging task and has wider applications. Deep3D [41] predicts a probabilistic disparity map from the left eye's view and renders a novel image for the right eye. The missing regions are inpainted implicitly using neural networks. [26] renders novel views using a predicted depth map. They utilize context-aware inpainting to fill in missing regions, thereby generating better results. To make single-view view synthesis more general, SynSin [40] takes a single image as input and can synthesize an image at any given pose. Recently, many approaches based on layered representations have been proposed that are better at handling occluded regions. [38] uses layered depth images (LDI) for single-view view synthesis, successfully inferring not only the depth of visible pixels but also the texture and depth of content that is occluded. [37] performs single-view view synthesis using MPI, which results in better performance. [21] extends MPI representation and proposes Variable MPI, which allows the locations of multiple planes to be inferred from the input image and the depth map. To further explore the potential of MPI for view synthesis, [20] combines MPI with NeRF and propose a novel layered representation called MINE, which results in considerable performance improvement. AdaMPI [13] was recently proposed to synthesize novel views for in-the-wild photographs, but it still requires a depth map input. In general, these prior MPI-based methods either randomly sample plane locations, which requires more planes and incurs huge computing overhead, or learn plane locations globally, which requires an additional depth map input.

3. Background

The symbols and notation in this paper are defined in Table 1. Our approach employs the same scene representation as MINE [20], which is referred to as *MINE planes* in this paper. MINE planes are a set of 4-channel (i.e., RGB and volume density) planes parallel to the current camera at different disparities and can be used to render the image and depth map in the current view using volume rendering. Specifically, given MINE planes (i.e., $\{c^{z_i}, \sigma^{z_i} | i = 1 \cdots N\}$), the current view can be rendered as:

$$\hat{I} = \sum_{i=1}^{N} w_i c^{z_i}, w_i = \mathcal{T}_i(1 - \exp(-\sigma^{z_i} \delta^{z_i})), \quad (1)$$

where $\mathcal{T}_i = \exp(-\sum_{j=1}^{i-1} \sigma^{z_j} \delta^{z_j}) : \mathbb{R}^2 \rightarrow \mathbb{R}^+$ is the map of accumulated transmittance from the first plane to the i-th plane, and $\mathcal{T}_i(x, y)$ denotes the probability of a ray traveling from (x, y, z_1) to (x, y, z_i) without hitting any object. Furthermore, $\delta^{z_i}(x, y) = ||\pi^{-1}([x, y, z_{i+1}]^T) - \pi^{-1}([x, y, z_i]^T)||_2 : \mathbb{R}^2 \rightarrow \mathbb{R}^+$ is the distance map between planes $i + 1$ and i. The depth map of the current view can be rendered similar to Eq. (1), i.e.:

$$\hat{D} = \sum_{i=1}^{N} w_i z_i. \quad (2)$$

Given MINE planes in the source view, a target view can be generated using a homography warping and volume rendering [16, 25]. Following the standard inverse homography [37, 14, 45], the correspondence between a pixel coordinate $[x_s, y_s]^T$ in a source plane and a pixel coordinate $[x_t, y_t]^T$ in a target plane is given by:

$$[x_s, y_s, 1]^T \sim K_s(R - \frac{tn^T}{z_i})K_t^{-1}[x_t, y_t, 1]^T, \quad (3)$$

where $n = [0, 0, 1]^T$ is the normal vector of MINE planes in the target view. For brevity, we denote Eq. (3) as $[x_s, y_s]^T = \mathcal{W}(x_t, y_t)$. MINE planes in the source view can then be projected to the target view as: $c_t^{z_i}(x_t, y_t) = c_s^{z_i}(\mathcal{W}(x_t, y_t))$, $\sigma_t^{z_i}(x_t, y_t) = \sigma_s^{z_i}(\mathcal{W}(x_t, y_t))$. Generated MINE planes in the target view are used to render the target image and the target depth map using Eqs. (1) and (2).

4. Our Method

In this section, we present our novel approach for single-view view synthesis. Figure 2 gives an overview of our approach. The source image is first fed into the encoder and the disparity sampler. The encoder is used to extract image features, and the disparity sampler is used to regress locally-learned plane locations (described in Section 4.1.2). The regressed locations are embedded in the same manner as NeRF [25] and concatenated with extracted features through channels. The decoder takes the concatenated features as input and predicts locally-learned planes in the source view, which can be used to render images in novel views. The appearance loss is computed mainly using the difference between the ground truth and predicted novel views. To build our occlusion-aware reprojection loss, an occlusion mask is first obtained using our detection method, and the reprojection loss is computed as the masked difference between the projected image and the ground truth (described in Section 4.1.3). Our BS-SA module can be applied after any layer of the decoder to handle occlusions without worrying about large feature maps (described in Section 4.2).

4.1. Locally-Learned Planes

4.1.1 The Point for Locally-Learned Planes

Compared to fixed planes. The insight on using MINE planes is to approximate the integral of volume rendering using the rectangular approximation method. Imagine that there are two rays that intersect on a pixel of a given plane. If the plane location is fixed, the network can only set this pixel to the average of densities of sampled points in two views, aiming to obtain a good approximation of the integral in both views. However, if the plane location is flexible (i.e., learned), the network can find a better plane location at which the densities of sampled points in two views are more similar or exactly the same, which provides a more accurate approximation of the integral of volume rendering for all views.

Compared to globally-learned planes. Since networks tend to produce low-frequency outputs, if there is not enough supervision or regularization, globally-learned planes would cluster around a certain disparity, and one of those planes would cluster all rendering weights (shown in our supplementary materials). Therefore, previous methods with globally-learned planes usually require a depth map as an additional input [21, 13]. Locally-learned planes itself as a regularization can avoid this issue.

For the reasons above, we first propose a disparity sampler to regress plane locations, which is then fed into a decoder to obtain locally-learned planes. However, a direct application of such a pipeline still makes the network not convergent due to the lack of depth information. We further propose two optimizing strategies that combine different disparity distributions of datasets to solve this issue. In addition, an occlusion-aware reprojection loss is also explored as a novel geometric supervision technique for single-view view synthesis.

4.1.2 Disparity Sampler

We design the disparity sampler as an encoder, taking a single image as input and regressing several offsets $\{v_i | 0 < v_i < 1, i = 1 \cdots N\}$. For locally learning, we pre-partition the disparity space $[d_f, d_n]$ into N bins uniformly, and the locations of locally-learned planes are computed as:

$$d_i = d_n + (v_i + i - 1)\tfrac{d_f - d_n}{N}. \tag{4}$$

Our formulation naturally restricts each locally-learned plane into the corresponding bin, thereby preventing planes from clustering as globally-learned planes do.

We observe that different datasets may have different disparity distributions, which impacts the convergence of our network. We divide these disparity distributions into two cases. The first is *the uniform disparity distribution*, which has approximately the same number of pixels at each disparity, while the second is *the aggregated disparity distribution*, which has most pixels concentrated at some disparities that are far apart and only a few pixels at the rest of the values. Detailed descriptions and visualizations of the distributions can be found in our supplementary materials. To make our disparity sampler work well with both disparity distributions, we propose the following parameter optimizing strategies.

Parameter optimizing strategy for uniform disparity distribution (U-opt): For images with uniform disparity distributions (e.g., the KITTI and RealEstate10K datasets), there are enough pixels in each bin to optimize the network parameters. Therefore, we propose U-opt to simultaneously optimize θ_{ED} and θ_S to fit $(\boldsymbol{c}, \boldsymbol{\sigma}) = \mathcal{F}_{\theta_{ED}, \theta_S}(I_s, \mathcal{S}(I_s))$.

Parameter optimizing strategy for aggregated disparity distribution (A-opt): For images with aggregated disparity distributions (e.g., the Flowers Light Field dataset), there could be only a few pixels in some bins for optimization, which cannot provide enough supervision to learn θ_{ED} and θ_S jointly. Therefore, we design A-opt, which uses a two-stage procedure. In the first stage, we optimize θ_{ED} without the disparity sampler. In the second stage, we employ the full pipeline in Figure 2, learning θ_{ED} with a small learning rate and θ_S with a big one. The first stage aims to provide a better initialization for the encode-decoder, based on which the sampler can be updated in the right direction during the second stage even with a few pixels.

Our proposed sampler is somewhat similar to Adabins [3]. Adabins and our method both attempt to get a depth distribution prior for each image, thus obtaining better prediction. However, due to different tasks and conditions, we have different designs: (1) With the ground truth depth, Adabins learns depth distributions globally, which is not feasible for our task, as explained in Section 4.1.1. (2) The task of Adabins is the monocular depth estimation, so Adabins employ a heavy network (an encoder-decoder and an mViT) to obtain a depth distribution prior (compared to our disparity sampler). This will cause high computing requirements, which is not expected in our work.

4.1.3 Occlusion-Aware Reprojection Loss

The occlusion-aware reprojection loss supervises only rendered depth maps, making up for the lack of depth supervision and helping to obtain better scene geometry. According to multi-view geometry [14], a pixel coordinate in the target image $[x_t, y_t]^T$ can be projected to a camera coordinate in

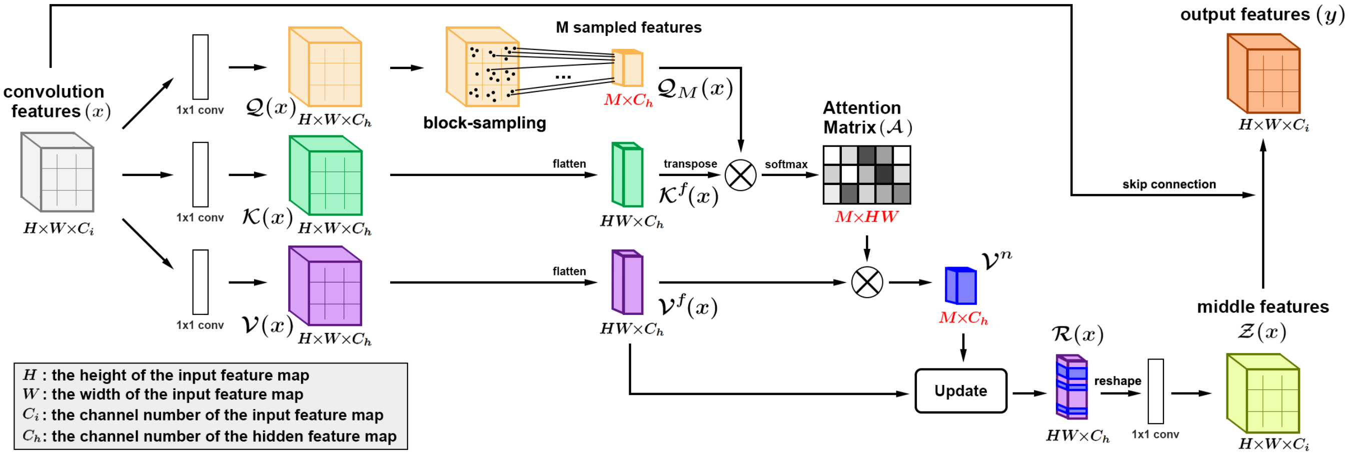


Figure 3. **Block-Sampling Self-Attention Module.** The block-sampling self-attention module reduces the size of the attention matrix from $HW \times HW$ to $M \times HW$ and solves the issue that the original self-attention mechanism cannot be applied to large feature maps. M is a hyper-parameter. "$\otimes$" denotes matrix multiplication. The softmax operation is performed on each row.

the source view $[X_s, Y_s, Z_s]^T$ as

$$[X_s, Y_s, Z_s]^T = T_{t\rightarrow s}\pi^{-1}([x_t, y_t, \hat{D}_t(x_t, y_t)]^T). \quad (5)$$

$[X_s, Y_s, Z_s]^T$ can be further projected to a pixel coordinate $[x_s, y_s]^T$ in the source image as

$$[x_s, y_s]^T = \pi([X_s, Y_s, Z_s]^T). \quad (6)$$

Then a pixel $[x_t, y_t]^T$ in the target image is considered occluded if $Z_s - \hat{D}_s(x_s, y_s) >= c \cdot s$, where c is a constant that equals 0.2 in our experiments and s is the scale of learned planes. The generated occlusion mask is denoted as $\mathcal{M}^o$, with 1 for occluded pixels and 0 for others.

Based on $\mathcal{M}^o$, the occlusion-aware reprojection loss can be computed as:

$$L_{rep} = \frac{1}{HW}\sum |I_t - I_t^r| \cdot (\mathbf{1} - \mathcal{M}^o), \quad (7)$$

where I_t is the ground truth in the target view. I_t^r is the image in the target view projected from the ground truth in the source view using Eqs. (5) and (6).

Combined with the reprojection loss, our overall loss is:

$$L_{total} = L_{app} + \lambda L_{rep}, \quad (8)$$

where L_{app} is the appearance loss, built on the edge-aware smoothness loss [10, 11] and the L1 loss between $\hat{I}_t$ and I_t. λ is set to 1 after searching in a manual range.

4.2. Self-Attention Occlusion Inference

The self-attention mechanism [39] improves the performance of neural networks by considering the correlation between features. Intuitively, it can be employed to infer occluded pixels using dis-occluded regions. However, due to the huge size of the attention matrix, the self-attention mechanism has prohibitive computational cost and vast video memory occupation [46] and is hard to be used on large feature maps for higher performance. To alleviate this problem, we propose a BS-SA module.

As shown in Figure 3, image features from the previous layer $\boldsymbol{x} \in \mathbb{R}^{H\times W\times C_i}$ are first transformed into different feature spaces $\mathcal{Q}(x) \in \mathbb{R}^{H\times W\times C_h}$, $\mathcal{K}(x) \in \mathbb{R}^{H\times W\times C_h}$, and $\mathcal{V}(x) \in \mathbb{R}^{H\times W\times C_h}$ using 1×1 convolutions. Unlike the original self-attention mechanism that causes an attention matrix of size $HW \times HW$, our BS-SA module reduces the size to $M \times HW$ with slight accuracy sacrifice by block-sampling M query points during each training step. Specifically, during each training step, we block-sample M locations in feature maps and take features of $\mathcal{Q}(x)$ at these locations as the query vector instead of all the features of $\mathcal{Q}(x)$. The query vector is then multiplied with flattened features of $\mathcal{K}(x)$ to obtain a smaller attention matrix $\mathcal{A}$. The resulting features of query points can be computed by multiplying $\mathcal{A}$ with the flattened features of $\mathcal{V}(x)$, while those of other points are set to the same as $\mathcal{V}(x)$. We summarize our BS-SA module in Algorithm 1.

5. Implementation and Results

In this section, we describe the implementation and evaluate the performance on different datasets. We perform both quantitative and qualitative comparisons on the KITTI [9], Flowers Light Fields [35], and RealEstate10K [45] datasets. We use the same metrics (SSIM, PSNR, and LPIPS [1]) as previous works [37, 20] to measure the quality of synthesized images and propose a new metric, Rendering Variance (RV), to measure the dispersion of weights in the volume rendering. We conduct many ablation studies on the KITTI

[1] The reported results on LPIPS in MINE [20] is wrong. They input images in the range [0, 1] to the LPIPS function, instead of [-1, 1]. (`https://github.com/vincentfung13/MINE/issues/4`).

```
Algorithm 1: Block-Sampling Self-Attention.
Input: The features from the previous layer $\boldsymbol{x}$; The number of sample points $M$.
Output: Output features $\boldsymbol{y}$.
Taking $\boldsymbol{x}$ as input, compute $\mathcal{Q}(\boldsymbol{x}), \mathcal{K}(\boldsymbol{x})$, and $\mathcal{V}(\boldsymbol{x})$ using 1x1 convolutions.
Randomly block-sample $M$ features in $\mathcal{Q}(\boldsymbol{x})$, and take sampled features as the query vector $\mathcal{Q}_M(\boldsymbol{x})$.
$\mathcal{K}^f(\boldsymbol{x}) \leftarrow \text{flatten}(\mathcal{K}(\boldsymbol{x}))$;
$\mathcal{V}^f(\boldsymbol{x}) \leftarrow \text{flatten}(\mathcal{V}(\boldsymbol{x}))$;
$\mathcal{A} \leftarrow \text{softmax}(\mathcal{Q}_M(\boldsymbol{x}) \times \mathcal{K}^f(\boldsymbol{x})^T)$;
$\mathcal{V}^{new} \leftarrow \mathcal{A} \times \mathcal{V}^f(\boldsymbol{x})$;
Update $\mathcal{V}^f(\boldsymbol{x})$ using $\mathcal{V}^{new}$ to obtain the resulting features $\mathcal{R}(\boldsymbol{x})$;
Taking $\mathcal{R}(\boldsymbol{x})$ as input, compute middle features $\mathcal{Z}(\boldsymbol{x})$ using a 1x1 convolution;
$\boldsymbol{y} \leftarrow \mathcal{Z}(\boldsymbol{x}) + \boldsymbol{x}$.
```

dataset to demonstrate the functionality of each proposed component, and a depth evaluation is also performed on the NYU-Depth V2 [33] and iBims-1 [17] datasets. Moreover, we compare our model with MINE on real-world images in our supplementary materials (SMs).

5.1. Rendering Variance

Given a ray ϕ with N sample points, rendering variance (RV) is formulated as:

$$RV(\phi) = \sum_i w_i (s \cdot z_i - z)^2. \tag{9}$$

w_i is defined in Eq. (1), and z_i is the depth of the i-th sample. z is the ground truth depth. s is a relative scale to solve the scale ambiguity of depth from a single image [37]. RV computes a weighted variance of depths of the sample points, and the weights are obtained using volume rendering. Intuitively, with smaller RVs, the rendering will concentrate on fewer sample points around the real depth, which may not do much for the current view but will generate sharper images with fewer artifacts for novel views. Since the RealEstate10K and Flowers Light Field dataset do not provide the ground truth depth, we only compare RV on the KITTI dataset in our experiments (using their public LiDAR data).

5.2. View Synthesis on KITTI

Using the same settings as previous works [37, 38, 20], 20 city sequences of the dataset are used to train our models, 4 sequences are used for validation, and the remaining 4 sequences are used for testing. During training, the left or the right image is randomly taken as the source image, and the other is the target image. Following [20], we also crop 5% from all sides of images when testing. The quantitative results have been shown in Table 2. LoLep has better performance than previous methods, and even our models with fewer planes use less memory and generate better results than models of previous methods with more planes (e.g., LoLep-16 vs. MINE-32, MINE-64 and MPI-32, LoLep-32 vs. MINE-64). The massive reduction of RV shows that our regressed locations allow the volume rendering to concentrate on fewer and more accurate planes, thereby generating sharper results and alleviating artifacts for novel views. As shown in Figure 4, LoLep can handle occlusions better (Figure 4(B)) and generate more reasonable geometry and shaper images (Figure 4(A), (C)-(D)).

Table 2. Evaluation results on the KITTI dataset. LoLep obtains the best performance compared to prior methods, and even LoLep with fewer planes uses less memory and generates better results than prior methods with more planes. The image resolution is 384×128. The best is in **bold**, and the second best is underlined.

Methods	LPIPS[1]↓	SSIM↑	PSNR↑	RV↓	Memory(MB)↓ (train/inference)	Converge↓ Iterations
LDI [38]	-	0.572	16.50	-	-	-
MPI-32 [37]	-	0.733	19.50	-	-	-
MINE-16 [20]	0.146	0.806	21.48	839.17	**8495 / 2039**	30k
MINE-32 [20]	0.134	0.813	21.52	492.50	14351 / 3167	30k
MINE-64 [20]	0.127	0.818	21.65	197.65	28457 / 5287	34k
LoLep-16 (full)	0.134	0.820	21.88	138.74	10868 / 2232	**24k**
LoLep-32 (full)	0.122	0.825	22.07	89.61	17208 / 3478	25k
LoLep-64 (full)	**0.117**	**0.828**	**22.17**	**49.53**	32421 / 5639	28k

* The training batch size is 4 and that of the inference is 1.

Table 3. Evaluation results on the RealEstate10K dataset. LoLep generates better results than prior methods. The image resolution is 384×256. The best is in **bold**, and the second best is underlined.

Methods	LPIPS[1]↓	SSIM↑	PSNR↑	Memory(MB)↓ (train/inference)	Converge↓ Iterations
SynSin [40]	-	0.740	22.31	-	-
MPI-32 [37]	-	0.785	23.52	-	-
MINE-16 [20]	0.208	0.804	23.71	**13195 / 2671**	1500k
MINE-32 [20]	0.187	0.813	24.33	19842 / 3955	1580k
MINE-64 [20]	0.176	0.818	24.50	34231 / 6421	1660k
LoLep-16 (full)	0.191	0.816	24.41	14963 / 2932	**1000k**
LoLep-32 (full)	0.174	0.828	25.02	22987 / 4413	1030k
LoLep-64 (full)	**0.161**	**0.832**	**25.14**	38754 / 6845	1100k

* The training batch size is 4 and that of the inference is 1.

5.3. View Synthesis on RealEstate10K

RealEstate10K [45] is a large-scale dataset collected from video clips on YouTube and consists of over 70,000 video sequences. Since different sequences have different scales, we use COLMAP [30, 31] to generate sparse point clouds of each sequence for scale-invariant learning [37]. Due to the huge size of the dataset, we randomly select 10% from the official training sequences to train our model. For testing, we randomly sample 600 sequences from the official test split and draw 5 frames from each sequence as source images. During both training and testing, target images are in the same sequence as source images and are randomly selected within 30 frames of source images. As shown in Table 3, LoLep generates better results than pre-

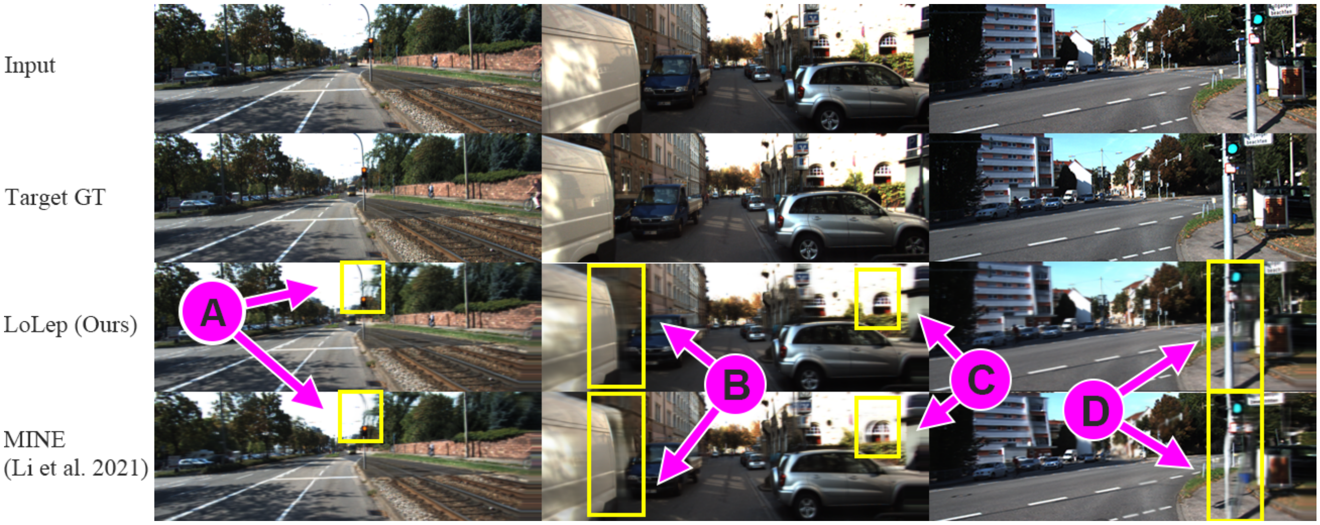


Figure 4. **Qualitative comparison on the KITTI dataset.** All images are from the test dataset and highlight the benefits of LoLep. (A) MINE synthesizes a broken pole. (B) MINE fails to infer occluded regions, thereby causing ghosting. (C) MINE regresses a suboptimal scene representation, thereby generating ghosting. (D) MINE synthesizes a twisted pole due to inconsistent depths of the pole.

Table 4. Evaluation results on the Flowers Light Field dataset. The image resolution is 512×384. The best is in **bold**, and the second best is underlined.

Methods	LPIPS[1]↓	SSIM↑	PSNR↑	Memory(MB)↓ (train/inference)	Converge↓ Iterations
LLFF [35]	-	0.822	28.10	-	-
MPI-32 [37]	-	0.851	30.10	-	-
MINE-16 [20]	0.208	0.862	29.86	**13251** / **4042**	250k
MINE-32 [20]	0.201	0.868	30.17	20340 / 6373	252k
MINE-64 [20]	0.188	0.873	30.31	34503 / 11407	260k
LoLep-16 (full)	0.198	0.868	30.21	15746 / 4832	**200k**
LoLep-32 (full)	0.183	0.876	30.35	23427 / 7231	**200k**
LoLep-64 (full)	**0.181**	**0.880**	**30.41**	39054 / 12384	205k

* The training batch size is 2 and that of the inference is 1.

vious methods on all metrics. Qualitative results in Figure 5 further demonstrate that LoLep synthesizes sharper and more realistic images for novel views (Figure 5(A)-(B)).

5.4. View Synthesis on Flowers Light Fields

The Flowers Light Fields dataset [35] consists of 3,343 light field photos of flowers. During training, a random image is selected as the source image and another image in the same light field is taken as the target image. In testing, we use a center image as the source image and four corner images as the target images. The training and testing splits are obtained from [20]. Quantitative results are shown in Table 4 and qualitative results are shown in our SMs.

5.5. Depth Evaluation on NYU-V2 and iBims-1

We further perform the depth evaluation on the NYU-Depth V2 [33] and iBims-1 [17] datasets using 64-plane models trained on RealEstate10K; the results are shown in Table 5. Since the models are trained on RealEstate10K but evaluated on other datasets, the generalization of models is the key to obtaining good performance. In our settings, we only using 10% of the dataset for training makes our models not have good generalization ability for new datasets, so the quality of depth maps is not comparable to state-of-the-art methods in depth estimation. However, depth maps generated by our models are significantly better than those of MINE with the same settings, which demonstrates that our method can regresses more accurate scene representation, a major point of improvement. Qualitative comparisons are shown in our SMs.

Table 5. Depth Evaluation on NYU-Depth V2 and iBims-1. The significant improvements show the superiority of LoLep in regressing more accurate scene representation. The best is in **bold**.

Data	Methods	rel↓	log10↓	RMS↓	$\sigma 1 \uparrow$	$\sigma 2 \uparrow$	$\sigma 3 \uparrow$	RV↓
NYU	MINE-64	0.17	0.07	0.58	0.77	0.93	0.98	3.41
	LoLep-64	**0.15**	**0.06**	**0.49**	**0.81**	**0.95**	**0.99**	**2.53**
iBims	MINE-64	0.17	0.08	0.73	0.75	0.91	0.96	3.95
	LoLep-64	**0.15**	**0.06**	**0.62**	**0.81**	**0.94**	**0.99**	**3.02**

5.6. Ablation Study

To further demonstrate the benefits of our proposed methods, we perform some ablation studies on the KITTI dataset; the results are shown in Table 6. (a.1)-(a.4) compare different approaches to obtaining plane locations. (a.1) is our baseline, obtaining plane locations by first dividing the disparity space into N bins and then randomly selecting locations in each bin as [20] does. (a.2) obtains plane locations by equally dividing the disparity space. (a.3) learns plane locations globally. (a.4) learns plane locations locally using our proposed sampler with U-opt, which is the best way verified by our experiments. (b.1)-(b.2) show the functionality of our proposed components, and (b.3) shows that

Table 6. Ablation study on the KITTI dataset. Our proposed components improve the performance for single-view view synthesis. For the BS-SA module, the performance improvement becomes greater as the number of sampling points M increases, but a too small M causes performance degradation. "Locations" indicates the method used to obtain plane locations. "SA-i" applies the original self-attention mechanism after the i-th layer of the decoder. "BS-SA-i(X)" applies our BS-SA module after the i-th layer of the decoder with $M = X$. "Reprojection" indicates whether the occlusion-aware reprojection loss is used. The best is in **bold**, and the second best is underlined.

Label	Methods	Locations	Attention	Reprojection	LPIPS↓	SSIM↑	PSNR↑	RV↓	Video Memory (train)
(a.1)	LoLep-16	Random			0.146	0.806	21.48	839.17	8495MB
(a.2)	LoLep-16	Equally-divided			0.158	0.793	21.28	992.32	8495MB
(a.3)	LoLep-16	Globally-learned			0.207	0.703	19.64	3524.53	8518MB
(a.4)	LoLep-16	U-opt			0.138	0.814	21.72	182.13	8518MB
(b.1)	LoLep-16	Random	BS-SA-3(400)		0.142	0.813	21.62	730.45	10645MB
(b.2)	LoLep-16	Random		√	0.141	0.813	21.70	722.67	8506MB
(b.3)	LoLep-16	Random		w/o $\mathcal{M}^o$	0.151	0.802	21.32	815.46	8502MB
(c)	LoLep-16	U-opt	SA-2	√	×	×	×	×	≫ 24576MB
(d.1)	LoLep-16	U-opt	SA-1	√	0.136	0.816	21.78	140.42	8856MB
(d.2)	LoLep-16	U-opt	BS-SA-1(200)	√	0.136	0.816	21.76	138.20	8596MB
(e.1)	LoLep-16	U-opt		√	0.138	0.814	21.74	145.33	8542MB
(e.2)	LoLep-16	U-opt	BS-SA-3(100)	√	0.142	0.808	21.57	271.56	9264MB
(e.3)	LoLep-16	U-opt	BS-SA-3(400)	√	$\underline{0.134}$	$\underline{0.820}$	$\underline{21.88}$	$\underline{138.74}$	10868MB
(e.4)	LoLep-16	U-opt	BS-SA-3(1500)	√	**0.132**	**0.824**	**21.94**	**115.27**	19346MB

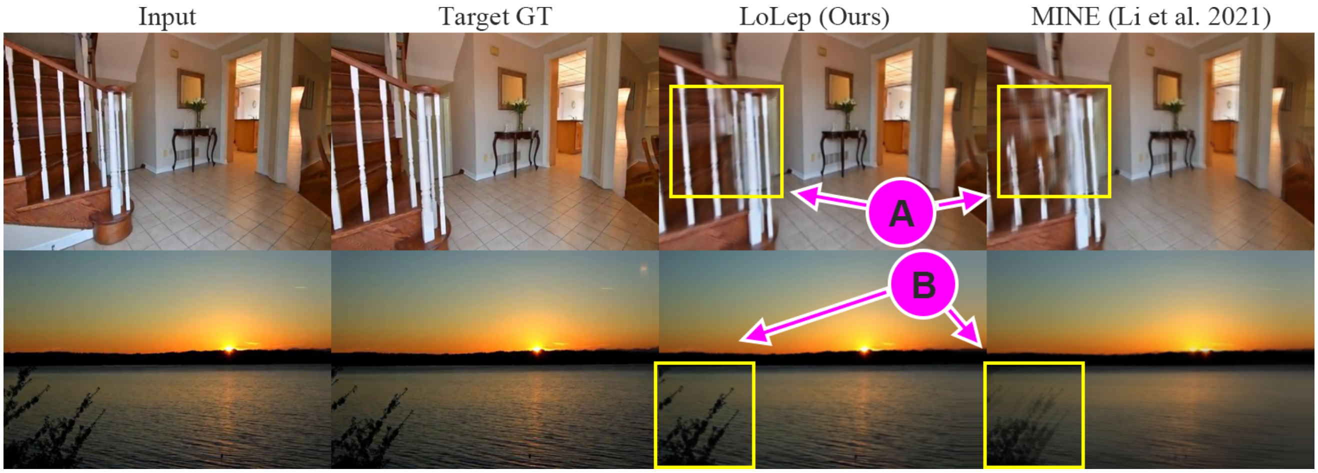


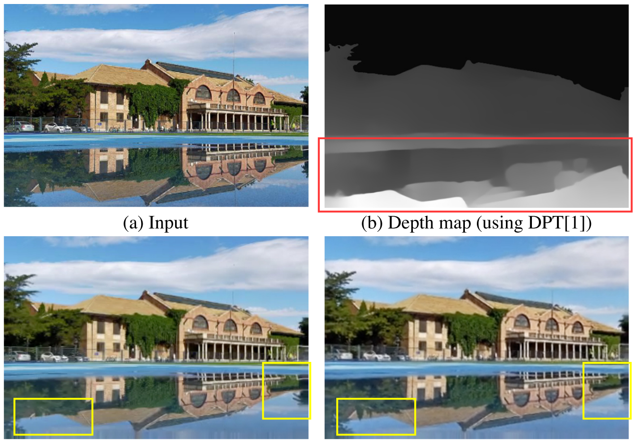
Figure 5. **Qualitative comparison on the RealEstate10K dataset.** (A) MINE fails to infer the geometry of the balustrade in stairs. (B) MINE generates many artifacts and blurry regions. In contrast, LoLep generates improved results.

only using a reprojection loss without an occlusion mask degrades the performance.

We also perform some experiments to explore the benefits of our BS-SA module. (c) shows that the original self-attention cannot be applied to feature maps of size 32×96 due to the vast memory overhead. However, our BS-SA module even can be applied to feature maps of size 64×192 for higher performance ((d.1) and (e.3)). Compared to the original self-attention, our BS-SA module can obtain comparable accuracy with less memory ((d.1)-(d.2)). In addition, as shown in (e.1)-(e.4), we can trade between the memory overhead and the performance by adjusting the number of sampling points M. As M increases, the improvements of our BS-SA module increase. However, a too small value of M leads to performance degradation because too few samples cannot guide parameters to update in the right direction.

6. Discussion on Methods using Monocular Depth Estimators

In some cases, off-the-shelf monocular depth estimators indeed aid in learning reasonable locations of MPI [32, 21, 13]. However, monocular depth estimation is still a challenging problem and has many unsolved limitations [44] (e.g., reflections and transferability), inevitably introducing these limitations to the single-view view synthesis. For example, Fig. 6 compares our approach to AdaMPI [13] on a real-world scene with mirror reflections. Due to the wrong depth estimation for reflection regions (the red box), AdaMPI produces obvious artifacts (yellow boxes). In contrast, our approach generates more resonable results. A possible explanation is that our sampler is jointly learned with the view synthesis task and solve only a simpler optimization problem (learning locations for different depth levels)



(a) Input

(b) Depth map (using DPT[1])

(c) AdaMPI

(d) LoLep (Ours)

Figure 6. **A failure case of AdaMPI on a scene with mirror reflections.** (b) is generated using DPT [29], consistent with AdaMPI.

than monocular depth estimation (learning per-pixel depth values).

7. Conclusion, Limitations, and Future Work

We present a novel method, LoLep, for single-view view synthesis that regresses locally-learned planes to represent scenes accurately, thus generating better novel views. This includes a novel disparity sampler with different parameter optimizing strategies, exploration of an occlusion-aware reprojection loss, and a novel BS-SA module that can be applied to large feature maps. Results on different datasets and real-world images show that LoLep can generate better results and achieve new state-of-the-art performance.

Limitations. Although locally-learned planes prevent all planes from clustering around a certain disparity and obtain promising results, it is a suboptimal solution. An optimal solution should allow planes to be optimized through the whole disparity space and prevent them from clustering using some new techniques. In the future, we will work on this topic and provide a further solution.

References

[1] Kara-Ali Aliev, Artem Sevastopolsky, Maria Kolos, Dmitry Ulyanov, and Victor S. Lempitsky. Neural point-based graphics. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part XXII*, volume 12367 of *Lecture Notes in Computer Science*, pages 696–712. Springer, 2020.

[2] Relja Arandjelovic and Andrew Zisserman. Nerf in detail: Learning to sample for view synthesis. *CoRR*, abs/2106.05264, 2021.

[3] Shariq Farooq Bhat, Ibraheem Alhashim, and Peter Wonka. Adabins: Depth estimation using adaptive bins. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*, pages 4009–4018. Computer Vision Foundation / IEEE, 2021.

[4] Chris Buehler, Michael Bosse, Leonard McMillan, Steven J. Gortler, and Michael F. Cohen. Unstructured lumigraph rendering. In Lynn Pocock, editor, *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 2001, Los Angeles, California, USA, August 12-17, 2001*, pages 425–432. ACM, 2001.

[5] Gaurav Chaurasia, Sylvain Duchêne, Olga Sorkine-Hornung, and George Drettakis. Depth synthesis and local warps for plausible image-based navigation. *ACM Trans. Graph.*, 32(3):30:1–30:12, 2013.

[6] Inchang Choi, Orazio Gallo, Alejandro J. Troccoli, Min H. Kim, and Jan Kautz. Extreme view synthesis. In *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*, pages 7780–7789. IEEE, 2019.

[7] Paul E. Debevec, Camillo J. Taylor, and Jitendra Malik. Modeling and rendering architecture from photographs: A hybrid geometry- and image-based approach. In John Fujii, editor, *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 1996, New Orleans, LA, USA, August 4-9, 1996*, pages 11–20. ACM, 1996.

[8] Andrew W. Fitzgibbon, Yonatan Wexler, and Andrew Zisserman. Image-based rendering using image-based priors. In *9th IEEE International Conference on Computer Vision (ICCV 2003), 14-17 October 2003, Nice, France*, pages 1176–1183. IEEE Computer Society, 2003.

[9] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The KITTI dataset. *Int. J. Robotics Res.*, 32(11):1231–1237, 2013.

[10] Clément Godard, Oisin Mac Aodha, and Gabriel J. Brostow. Unsupervised monocular depth estimation with left-right consistency. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 6602–6611. IEEE Computer Society, 2017.

[11] Clément Godard, Oisin Mac Aodha, Michael Firman, and Gabriel J. Brostow. Digging into self-supervised monocular depth estimation. In *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*, pages 3827–3837. IEEE, 2019.

[12] Steven J. Gortler, Radek Grzeszczuk, Richard Szeliski, and Michael F. Cohen. The lumigraph. In John Fujii, editor, *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 1996, New Orleans, LA, USA, August 4-9, 1996*, pages 43–54. ACM, 1996.

[13] Yuxuan Han, Ruicheng Wang, and Jiaolong Yang. Single-view view synthesis in the wild with learned adaptive multiplane images. In Munkhtsetseg Nandigjav, Niloy J. Mitra, and Aaron Hertzmann, editors, *SIGGRAPH '22: Special Interest Group on Computer Graphics and Interactive Techniques Conference, Vancouver, BC, Canada, August 7 - 11, 2022*, pages 14:1–14:8. ACM, 2022.

[14] Andrew Harltey and Andrew Zisserman. *Multiple view geometry in computer vision (2. ed.)*. Cambridge University Press, 2006.

[15] Peter Hedman, Julien Philip, True Price, Jan-Michael Frahm, George Drettakis, and Gabriel J. Brostow. Deep blending for free-viewpoint image-based rendering. *ACM Trans. Graph.*, 37(6):257:1–257:15, 2018.

[16] James T. Kajiya and Brian Von Herzen. Ray tracing volume densities. In Hank Christiansen, editor, *Proceedings of the 11th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 1984, Minneapolis, Minnesota, USA, July 23-27, 1984*, pages 165–174. ACM, 1984.

[17] Tobias Koch, Lukas Liebel, Friedrich Fraundorfer, and Marco Körner. Evaluation of cnn-based single-image depth estimation methods. In Laura Leal-Taixé and Stefan Roth, editors, *Computer Vision - ECCV 2018 Workshops - Munich, Germany, September 8-14, 2018, Proceedings, Part III*, volume 11131 of *Lecture Notes in Computer Science*, pages 331–348. Springer, 2018.

[18] Johannes Kopf, Fabian Langguth, Daniel Scharstein, Richard Szeliski, and Michael Goesele. Image-based rendering in the gradient domain. *ACM Trans. Graph.*, 32(6):199:1–199:9, 2013.

[19] Marc Levoy and Pat Hanrahan. Light field rendering. In John Fujii, editor, *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 1996, New Orleans, LA, USA, August 4-9, 1996*, pages 31–42. ACM, 1996.

[20] Jiaxin Li, Zijian Feng, Qi She, Henghui Ding, Changhu Wang, and Gim Hee Lee. MINE: towards continuous depth MPI with nerf for novel view synthesis. In *2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021*, pages 12558–12568. IEEE, 2021.

[21] Qinbo Li and Nima Khademi Kalantari. Synthesizing light field from a single image with variable MPI and two network fusion. *ACM Trans. Graph.*, 39(6):229:1–229:10, 2020.

[22] Miaomiao Liu, Xuming He, and Mathieu Salzmann. Geometry-aware deep network for single-image novel view synthesis. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 4616–4624. Computer Vision Foundation / IEEE Computer Society, 2018.

[23] 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 *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*, pages 7210–7219. Computer Vision Foundation / IEEE, 2021.

[24] Moustafa Meshry, Dan B. Goldman, Sameh Khamis, Hugues Hoppe, Rohit Pandey, Noah Snavely, and Ricardo Martin-Brualla. Neural rerendering in the wild. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 6878–6887. Computer Vision Foundation / IEEE, 2019.

[25] 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 Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part I*, volume 12346 of *Lecture Notes in Computer Science*, pages 405–421. Springer, 2020.

[26] Simon Niklaus, Long Mai, Jimei Yang, and Feng Liu. 3d ken burns effect from a single image. *ACM Trans. Graph.*, 38(6):184:1–184:15, 2019.

[27] Keunhong Park, Utkarsh Sinha, Jonathan T. Barron, Sofien Bouaziz, Dan B. Goldman, Steven M. Seitz, and Ricardo Martin-Brualla. Nerfies: Deformable neural radiance fields. In *2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021*, pages 5845–5854. IEEE, 2021.

[28] Eric Penner and Li Zhang. Soft 3d reconstruction for view synthesis. *ACM Trans. Graph.*, 36(6):235:1–235:11, 2017.

[29] René Ranftl, Alexey Bochkovskiy, and Vladlen Koltun. Vision transformers for dense prediction. In *ICCV*, 2021.

[30] Johannes L. Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 4104–4113. IEEE Computer Society, 2016.

[31] Johannes L. Schönberger, Enliang Zheng, Jan-Michael Frahm, and Marc Pollefeys. Pixelwise view selection for unstructured multi-view stereo. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part III*, volume 9907 of *Lecture Notes in Computer Science*, pages 501–518. Springer, 2016.

[32] Meng-Li Shih, Shih-Yang Su, Johannes Kopf, and Jia-Bin Huang. 3d photography using context-aware layered depth inpainting. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pages 8025–8035. Computer Vision Foundation / IEEE, 2020.

[33] Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. Indoor segmentation and support inference from RGBD images. In Andrew W. Fitzgibbon, Svetlana Lazebnik, Pietro Perona, Yoichi Sato, and Cordelia Schmid, editors, *Computer Vision - ECCV 2012 - 12th European Conference on Computer Vision, Florence, Italy, October 7-13, 2012, Proceedings, Part V*, volume 7576 of *Lecture Notes in Computer Science*, pages 746–760. Springer, 2012.

[34] Vincent Sitzmann, Justus Thies, Felix Heide, Matthias Nießner, Gordon Wetzstein, and Michael Zollhöfer. Deepvoxels: Learning persistent 3d feature embeddings. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 2437–2446. Computer Vision Foundation / IEEE, 2019.

[35] Pratul P. Srinivasan, Tongzhou Wang, Ashwin Sreelal, Ravi Ramamoorthi, and Ren Ng. Learning to synthesize a 4d RGBD light field from a single image. In *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, pages 2262–2270. IEEE Computer Society, 2017.

[36] Alex Trevithick and Bo Yang. GRF: learning a general radiance field for 3d scene representation and rendering. *CoRR*, abs/2010.04595, 2020.

[37] Richard Tucker and Noah Snavely. Single-view view synthesis with multiplane images. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pages 548–557. Computer Vision Foundation / IEEE, 2020.

[38] Shubham Tulsiani, Richard Tucker, and Noah Snavely. Layer-structured 3d scene inference via view synthesis. In Vittorio Ferrari, Martial Hebert, Cristian Sminchisescu, and Yair Weiss, editors, *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part VII*, volume 11211 of *Lecture Notes in Computer Science*, pages 311–327. Springer, 2018.

[39] Xiaolong Wang, Ross B. Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 7794–7803. Computer Vision Foundation / IEEE Computer Society, 2018.

[40] Olivia Wiles, Georgia Gkioxari, Richard Szeliski, and Justin Johnson. Synsin: End-to-end view synthesis from a single image. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pages 7465–7475. Computer Vision Foundation / IEEE, 2020.

[41] Junyuan Xie, Ross B. Girshick, and Ali Farhadi. Deep3d: Fully automatic 2d-to-3d video conversion with deep convolutional neural networks. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part IV*, volume 9908 of *Lecture Notes in Computer Science*, pages 842–857. Springer, 2016.

[42] Dejia Xu, Yifan Jiang, Peihao Wang, Zhiwen Fan, Humphrey Shi, and Zhangyang Wang. Sinnerf: Training neural radiance fields on complex scenes from a single image. *CoRR*, abs/2204.00928, 2022.

[43] Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. pixelnerf: Neural radiance fields from one or few images. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*, pages 4578–4587. Computer Vision Foundation / IEEE, 2021.

[44] Chaoqiang Zhao, Qiyu Sun, Chongzhen Zhang, Yang Tang, and Feng Qian. Monocular depth estimation based on deep learning: An overview. *CoRR*, abs/2003.06620, 2020.

[45] Tinghui Zhou, Richard Tucker, John Flynn, Graham Fyffe, and Noah Snavely. Stereo magnification: learning view synthesis using multiplane images. *ACM Trans. Graph.*, 37(4):65:1–65:12, 2018.

[46] Zhen Zhu, Mengdu Xu, Song Bai, Tengteng Huang, and Xiang Bai. Asymmetric non-local neural networks for semantic segmentation. In *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*, pages 593–602. IEEE, 2019.

[47] C. Lawrence Zitnick, Sing Bing Kang, Matthew Uyttendaele, Simon A. J. Winder, and Richard Szeliski. High-quality video view interpolation using a layered representation. *ACM Trans. Graph.*, 23(3):600–608, 2004.