MINE: Towards Continuous Depth MPI with NeRF for Novel View Synthesis

Jiaxin Li[1*], Zijian Feng[1*], Qi She[1], Henghui Ding[1], Changhu Wang[1], Gim Hee Lee[2]

[1]ByteDance, [2]National University of Singapore

Abstract

*In this paper, we propose **MINE** to perform novel view synthesis and depth estimation via dense 3D reconstruction from a single image. Our approach is a continuous depth generalization of the **M**ultiplane **I**mages (MPI) by introducing the **NE**ural radiance fields (NeRF). Given a single image as input, MINE predicts a 4-channel image (RGB and volume density) at arbitrary depth values to jointly reconstruct the camera frustum and fill in occluded contents. The reconstructed and inpainted frustum can then be easily rendered into novel RGB or depth views using differentiable rendering. Extensive experiments on RealEstate10K, KITTI and Flowers Light Fields show that our MINE outperforms state-of-the-art by a large margin in novel view synthesis. We also achieve competitive results in depth estimation on iBims-1 and NYU-v2 without annotated depth supervision. Our source code is available at* https://github.com/vincentfung13/MINE.

1. Introduction

Interactive 3D scene is a fascinating way to achieve immersive user experience similar to augmented/virtual reality. To automate or simplify the creation of 3D scenes, increasing efforts are invested on novel view synthesis from a single or multiple image(s) that enables rendering at arbitrary camera poses according to user's interaction. Despite its usefulness, the novel view synthesis problem is challenging because it requires precise geometry understanding, and inpainting of the occluded geometry and textures.

To tackle the problem of view synthesis, most existing methods focus on the design of 3D or 2.5D representations of the scene, and the rendering techniques of novel views. A straightforward idea is to perform Structure-from-Motion (SfM) [39, 38] or monocular/multiview depth estimations [12, 11, 61, 53] to recover the 3D scene. Unfortunately, this naive approach is insufficient to acquire accurate dense 3D geometry and fill in the occluded contents of the scene. Consequently, this results to distorsion and artifacts in the rendered novel views. To alleviate this problem, more sophisticated representations including Layed Depth Image (LDI) [52, 41], Multiplane Images (MPI) [51] are used with deep networks to recovered 2.5D information from single / multiple images. However, 2.5D approaches usually suffer from limited resolution to represent the full 3D scene.

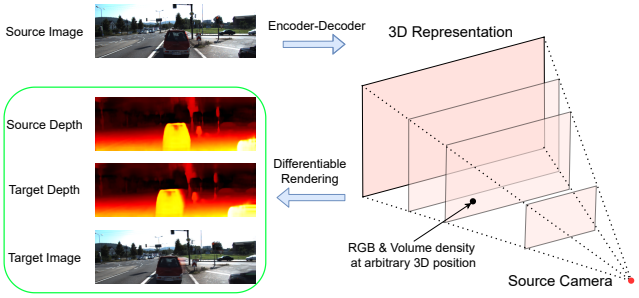

Figure 1. Overview of our proposed method.

Recently, the MPI [51] representation attracts a lot of attentions. Specifically, it is a deep network supervised with other image views of the same scene to lift a RGB image into multiple planes of RGB and alpha values. Novel views are then rendered by performing homography warping and integral over the planes. Despite its success, the MPI method fails to represent continuous 3D space effectively. Its depth-wise resolution is limited by the number of discrete planes, and thus the MPIs cannot be converted to other 3D representations such as mesh, point cloud, etc. In contrast, the Neural Radiance Fields (NeRF) [29] is concurrently proposed to recover 3D information from images using a Multi-layer Perceptron (MLP). The MLP takes a 3D position and a 2D viewing direction as input to predict the RGB and volume occupancy density at that query position. Although NeRF produces high quality 3D structures and novel views, it has to be trained per scene, i.e. one MLP represents only one scene.

In view that MPI [51] is unable to represent the full 3D space, we propose MINE that generalizes MPI to a continuous 3D representation similar to NeRF [29]. Specifically, an input image is first fed into an encoder network to obtain the image features. A decoder network then takes these

*Equal contribution.

image features and an arbitrary depth as inputs to produce a 4-channel, i.e. RGB and volume density values, plane fronto-parallel to the input camera. As shown in Sec. 3.1, our MINE can effectively reconstructs the camera frustum in full 3D space since the plane depth is arbitrary. We prove in Sec. 3.5 that the MPI [51] representation is a limited special case of our approach. Our main contributions are:

- Performs continuous and occlusion-inpainted 3D reconstruction from a single image.
- Our MINE is a continuous depth generalization of the MPI by introducing the NeRF idea.
- Significantly outperforms existing state-of-the-art methods in indoor and outdoor view synthesis and depth estimation.

2. Related Work

Explicit 3D representations for view synthesis. Early works on light fields [23, 5, 13] achieve view synthesis by interplotating nearby views given a set of input images. A recent work [47], predicts the entire light field from a single image. Volumetric representations [22, 40, 48, 19, 16, 44] and view synthesis with predicted depth maps [32, 56] have also been studied intensively. Recently, layered representations, specifically layered depth image (LDI) [41, 52, 42] and multiplane image (MPI) [48, 62, 28, 46, 51], become popular due to their appealing properties of explicitly modeling occluded contents. An MPI consists of multiple planes of RGB-α images at fixed depths, its performance is limited by sparse depths discretization. LDI stores multiple RGBD pixels at every pixel lattice, and thus naturally handles arbitrary number of layers. Rendering of LDI in [52] leads to problems like cracks. More sophisticated LDI methods like [42] contain iterative depth edges inpainting steps, which is prohibitively slow for real-world applications.

Another line of related works is self-supervised depth estimation. These works aim to train depth estimation models using image reconstruction error as the main supervision signal without ground truth depths. Exploiting epipolar geometry, [7, 57, 11] predict per-pixel disparities that recover one image from the other in the pair. [11] additionally adds a left-right consistency term to improve the quality of the disparity maps. [61, 53, 25, 54, 12] have been proposed to use monocular videos for self-supervision. Although depth estimation and view synthesis are closely related, good depth estimation results do not guarantee good view synthesis results, and vice versa. We show that our method achieves state-of-the-art performance in both view synthesis and depth estimation.

Implicit 3D representations for view synthesis. Recent works show that neural network can be used as an implicit representation for 3D shapes. To encode 3D shapes into the network weights, [3, 17, 10, 1, 27, 33, 9, 35] map continuous 3D coordinates to signed distance functions or occupancy. However, they require supervision from ground truth 3D geometry. Others [45, 31, 58] alleviate this requirements with differentiable rendering, which enables RGB-only supervision. Nonetheless, these methods do not deliver photo-realistic rendering results in scenes with complex structure. Recently, NeRF [29] shows astonishing results for novel view synthesis. NeRF works by mapping a continuous 3D coordinate and 2D viewing direction to a 4D output of RGB values and volume density. Works have been proposed to improve NeRF to images in the wild [26] and non-rigid scenes [34]. However, NeRF needs to be optimized per scene. PixelNerf [59] is proposed to solve the generalization problem while is does not solve the single-image scale ambiguity problem. GRF [50] is another improvement that works for multiple-view input. Neither of [59, 50] presents experiments on large scale real world datasets.

We take the best of the two worlds of NeRF and MPI and propose a new 3D representation, which we call MINE. Our method predicts planes of RGB-σ images at any given arbitrary depths, thus allows continuous / dense 3D reconstruction of the scene. Unlike NeRF which encodes the scene geometry in the network weights, our network conditions on the input image, and thus can generalize to unseen scenes.

3. Our Approach

The input to our method is a single image, and the output is our 3D representation illustrated in Sec. 3.1. Our network design and training pipeline are introduced in Sec. 3.2 and 3.3. Furthermore, we discuss how our MINE relates to NeRF and MPI in Sec. 3.4 and 3.5.

3.1. 3D Representation

3.1.1 Planar Neural Radiance Field

We utilize the perspective geometry to represent the camera frustum. Let us denote a pixel coordinate on the image plane as $[x, y]^\top \in \mathbb{R}^2$, and the pinhole camera intrinsic as $\mathrm{K} \in \mathbb{R}^{3\times3}$. A 3D point in the camera frustum is represented as $[x, y, z]^\top$, where z is the depth of that point with respect to the camera. We define the conversion $\mathfrak{C}(\cdot)$ from perspective 3D coordinate $[x, y, z]^\top$ to Cartesian coordinate $[X, Y, Z]^\top$ as:

$$\mathfrak{C}(\begin{bmatrix} x \\ y \\ z \end{bmatrix}) = \mathrm{K}^{-1} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} z = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} zx \\ zy \\ z \end{bmatrix}. \quad (1)$$

As shown in Fig. 2, we can sample arbitrary number of planes within the camera frustum with different depth values $z \in [z_n, z_f]$. Each plane consists of the RGB values $c_z : [x, y]^\top \to \mathbb{R}^3$ and volume densities $\sigma_z : [x, y]^\top \to \mathbb{R}^+$

of every point $[x, y]^\top \in \mathbb{R}^2$ on that plane. The volume density $\sigma(x, y, z)$ represents the differential probability of a ray terminating at an infinitesimal particle at location $[x, y, z]^\top$. The camera frustum within depth range $[z_n, z_f]$ is reconstructed continuously because the RGB $c(x, y, z)$ and $\sigma(x, y, z)$ of any position $[x, y, z]^\top$ are given by sampling a plane at depth z, and querying $c_z(x, y)$ and $\sigma_z(x, y)$. We call this the *planar neural radiance field* because it represents the frustum using planes instead of rays in [29].

3.1.2 Volume Rendering

c and σ defined above are continuous two-dimensional functions that represent every possible position in the frustum. In practice, we discretize the planar radiance field in two aspects: a) The frustum consists of N planes $\{c_{z_i}, \sigma_{z_i} \mid i = 1, \cdots, N\}$. b) Each plane (c_{z_i}, σ_{z_i}) is simplified into a 4-channel image plane at depth z_i. Note the discretization is only for the convenience of rendering. The discretized representation is still able to acquire the RGBσ values at any 3D position because: a) Each plane can be at arbitrary depth $z_i \in [z_n, z_f]$ and b) sub-pixel sampling is trivial at each 4-channel plane.

Rendering the input image $\hat{\mathbf{I}}_{src}$. We first illustrate the rendering mechanism with the naive setting of rendering $\hat{\mathbf{I}}_{src}$. A novel view can then be rendered in a similar way with an additional homography warping. Rendering $\hat{\mathbf{I}}_{src}$ is straightforward using the principles from classical volume rendering [18, 29], i.e.:

$$\hat{\mathbf{I}} = \sum_{i=1}^{N} T_i \big(1 - \exp(-\sigma_{z_i}\delta_{z_i})\big) c_{z_i}, \tag{2}$$

where $$T_i = \exp\Big(-\sum_{j=1}^{i-1} \sigma_{z_j}\delta_{z_j}\Big) : \mathbb{R}^2 \to \mathbb{R}^+$$

is the map of accumulated transmittance from the first plane to plane i. Specifically, $T_i(x, y)$ denotes the probability of a ray travels from (x, y, z_1) to (x, y, z_i) without hitting any object. Furthermore,

$$\delta_{z_i}(x, y) = \|\mathfrak{C}([x, y, z_{i+1}]^\top) - \mathfrak{C}([x, y, z_i]^\top)\|_2 : \mathbb{R}^2 \to \mathbb{R}^+$$

is the distance map between plane $i + 1$ and i.

According to Eq. 2, the collection of $\{(c_{z_i}, \sigma_{z_i}, z_i) \mid i = 1, \cdots, N\}$ is required to render the input image. As shown in Sec. 3.2, (c_{z_i}, σ_{z_i}) is the output of our network, which takes $\mathbf{I}_{src}$ and $d_i = 1/z_i$ as inputs. Following the stratified sampling strategy of [29], $\{z_i \mid i = 1, \cdots, N\}$ is sampled within $[z_n, z_f]$. In fact, we sample disparity $\{d_i = 1/z_i\}$ in the perspective geometry. Specifically, $[d_n, d_f]$ is partitioned into N evenly spaced bins, and a sample is drawn uniformly from each bin, i.e.:

$$d_i \sim \mathcal{U}\Big[d_n + \frac{i}{N}(d_f - d_n), d_n + \frac{i-1}{N}(d_f - d_n)\Big]. \tag{3}$$

The sampling strategy ensures that our network is exposed to every depth value in the frustum during training, and thus learning a continuous (c_{z_i}, σ_{z_i}).

In addition, the depth map of the input image can be rendered in a similar way as Eq. 2, i.e.:

$$\hat{\mathcal{Z}} = \sum_{i=1}^{N} T_i \big(1 - \exp(-\sigma_{z_i}\delta_{z_i})\big) z_i. \tag{4}$$

Rendering a novel view $\hat{\mathbf{I}}_{tgt}$. As illustrated in Fig. 2, the rendering into a novel view with camera rotation $\mathrm{R} \in \mathbb{R}^{3\times3}$ and translation $\mathbf{t} \in \mathbb{R}^3$ is achieved in three steps:

1) Apply a homography warping $\mathcal{W}(.)$ to establish the correspondence between the source pixel coordinates $[x_s, y_s]^\top$ and the target pixel coordinates $[x_t, y_t]^\top$. We follow the standard inverse homography [14, 62, 51] to define $\mathcal{W}(.)$. The correspondence between a pixel coordinate $[x_t, y_t]^\top$ in the target plane and and a pixel coordinate $[x_s, y_s]^\top$ in the source plane is given by:

$$[x_s, y_s, 1]^\top \sim \mathrm{K}\Big(\mathrm{R} - \frac{\mathbf{t}\mathbf{n}^\top}{z_i}\Big)\mathrm{K}^{-1}[x_t, y_t, 1]^\top, \tag{5}$$

where $\mathbf{n} = [0, 0, 1]^\top$ is the normal vector of the fronto-parallel plane (c_{z_i}, σ_{z_i}) with respect to the source camera. For brevity, we now denote the above warping as $[x_s, y_s]^\top = \mathcal{W}_{z_i}(x_t, y_t)$ for the plane at depth z_i with respective to the source camera. We then compute the plane projections $(c'_{z_i}, \sigma'_{z_i})$ at the target frame as: $c'_{z_i}(x_t, y_t) = c_{z_i}(x_s, y_s)$, and $\sigma'_{z_i}(x_t, y_t) = \sigma_{z_i}(x_s, y_s)$. Note that the N planes are fronto-parallel to the source camera, and therefore $(c'_{z_i}, \sigma'_{z_i})$ is just the projection into the target camera.

2) The volume rendering relies on the density σ at each location, and the distances between each point along the ray. Consequently, we can compute:

$$\begin{aligned}\delta'_{z_i}(x_t, y_t) = \|&\mathfrak{C}([\mathcal{W}_{z_{i+1}}(x_t, y_t), z_{i+1}]^\top) \\ &- \mathfrak{C}([\mathcal{W}_{z_i}(x_t, y_t), z_i]^\top)\|_2.\end{aligned} \tag{6}$$

As illustrated in Fig. 2, let us imagine a ray that starts from target camera origin and intersects the target image at pixel coordinate $[x_t, y_t]^\top$ to better understand Eq. 6. This ray intersects the (c_{z_i}, σ_{z_i}) plane at pixel coordinate $\mathcal{W}_{z_i}(x_t, y_t)$ with respect to the source camera. Similarly, the ray intersects the $(c_{z_{i+1}}, \sigma_{z_{i+1}})$ plane at source camera pixel coordinate $\mathcal{W}_{z_{i+1}}(x_t, y_t)$. $\delta'_{z_i}(x_t, y_t)$ represents the Euclidean distance between the two intersections.

3) Finally, the rendering into a novel view can be achieved by applying Eq. 2 after replacing c, σ, δ with c', σ', δ'.

3.2. Network and Training Design

As shown above, a discretized planar radiance field requires a set of depth samples $\{z_i \mid i = 1, \cdots, N\}$, and 4-channel images $\{(c_{z_i}, \sigma_{z_i}) \mid i = 1, \cdots, N\}$. Depth samples $\{z_i\}$ or disparity samples $\{d_i = 1/z_i\}$ are randomly sampled according to Eq. 3.

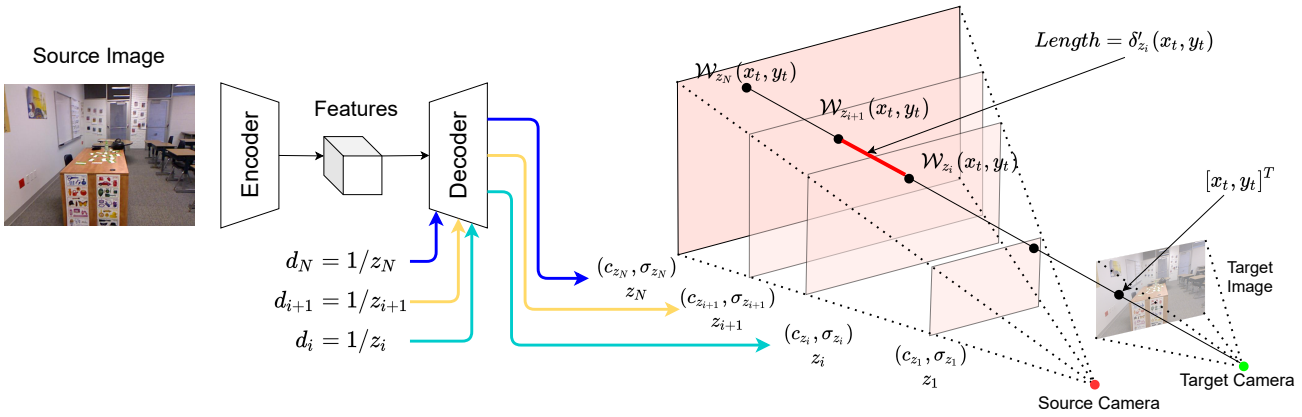

Figure 2. Our network is an encoder-decoder architecture (c.f. Sec. 3.2) that takes an input image and outputs the reconstructed source camera frustum. We then render the reconstructed source camera frustum into a novel view (c.f. Sec. 3.1.2).

Encoder-Decoder Structure. The 4-channel images $\{(c_{z_i}, \sigma_{z_i})\}$ are predictions from our network, which takes a single image and $\{z_i\}$ as input. Our network architecture is shown in Fig. 2. The encoder takes the image as input and produces a series of feature maps. We utilize the Resnet-50 [15] as the encoder. The decoder takes the feature maps and a single disparity value $d_i = 1/z_i$ as input, and produces the 4-channel image (c_{z_i}, σ_{z_i}). The decoder design is similar to Monodepth2 [12]. In training and inference, the encoder runs only once per image (or per mini-batch of images), while the decoder runs N-times to generate the discrete set of planes $\{(c_{z_i}, \sigma_{z_i}) \mid i = 1, \cdots, N\}$.

Disparity Encoding. We find that directly feeding d_i into the decoder gives poor performance, which is consistent with [29, 36, 49]. To circumvent this problem, we apply an encoding function $\gamma : \mathbb{R} \to \mathbb{R}^L$ to d_i before feeding into the decoder, i.e.:

$$\gamma(d_i) = [\sin(2^0 \pi d_i), \cos(2^0 \pi d_i), \cdots, \sin(2^{L-1} \pi d_i), \cos(2^{L-1} \pi d_i)]. \quad (7)$$

3.3. Supervision with RGB Videos

Multi-view images or RGB videos are used to train the network similar to [51]. During training, an input image I_{src} is fed into the network and then rendered into $(\hat{\mathrm{I}}_{tgt}, \hat{\mathcal{Z}}_{tgt})$, according to the novel view camera rotation R and scale-calibrated camera translation $\mathbf{t}'$. The core supervision is by comparing $\hat{\mathrm{I}}_{tgt}$ with the ground truth target image I_{tgt}.

3.3.1 Scale Calibration

The depth scale is ambiguous up to a scale factor $s \in \mathbb{R}^+$ since the input to our system is a single image. The range of the frustum reconstruction $[z_n, z_f]$ is pre-defined as a hyper-parameter, which we set as $z_n = 1$, $z_f = 1000$. Instead of scaling our 3D representation, we scale the camera translation $\mathbf{t}$ into $\mathbf{t}'$ at both training and inference.

To solve the scale factor s, we perform a scale calibration between the sparse 3D points from video Structure-from-Motion (SfM) and our synthesized depth map of Eq. 4. Specifically, we run the SfM using COLMAP [38, 39] on each video to get a sparse point set $\mathbf{P_s} = \{(x_j, y_j, z_j)\}$ for each image. The coordinates here follow the same perspective geometry, i.e. $[x_j, y_j]^\top$ is the pixel coordinate on the image, and z_j is the depth of the corresponding 3D point. After feeding the source image to our network and rendering the predicted depth map $\hat{\mathcal{Z}}_{src}$ using Eq. 4, similar to [51], the scale is estimated by:

$$s = \exp\left[\frac{1}{|\mathbf{P_s}|} \sum_{(x,y,z) \in \mathbf{P_s}} \left(\ln(\hat{\mathcal{Z}}_{src}(x, y) - \ln z\right)\right]. \quad (8)$$

Finally, the calibrated translation is given by $\mathbf{t}' = \mathbf{t} \cdot s$.

3.3.2 Loss Functions

There are four terms in the loss function: RGB L1 loss $\mathcal{L}_{\text{L1}}$, RGB SSIM loss $\mathcal{L}_{\text{ssim}}$, edge-aware disparity map smoothness loss $\mathcal{L}_{\text{smooth}}$, and the optional sparse disparity loss $\mathcal{L}_d$. The total loss is given by:

$$\mathcal{L} = \lambda_{\text{L1}} \mathcal{L}_{\text{L1}} + \lambda_{\text{ssim}} \mathcal{L}_{\text{ssim}} + \lambda_{\text{smooth}} \mathcal{L}_{\text{smooth}} + \lambda_d \mathcal{L}_d, \quad (9)$$

where λ_{L1}, λ_{ssim}, λ_{smooth} and λ_d are hyperparameters to weigh the respective loss term.

RGB L1 and SSIM loss. The L1 and SSIM [55] losses:

$$\mathcal{L}_{\text{L1}} = \frac{1}{3HW} \sum |\hat{\mathrm{I}}_{tgt} - \mathrm{I}_{tgt}|, \quad \mathcal{L}_{\text{ssim}} = 1 - \text{SSIM}(\hat{\mathrm{I}}_{tgt}, \mathrm{I}_{tgt}) \quad (10)$$

are to encourage the synthesized target image $\hat{\mathrm{I}}_{tgt}$ to match the ground truth I_{tgt}. Both $\hat{\mathrm{I}}_{tgt}$ and I_{tgt} are 3-channel RGB images of size $H \times W$.

Edge-aware disparity map smoothness loss. We impose an edge-aware smoothness loss on the synthesized disparity map to penalize drastic changes in disparities at locations

where the original image is smooth, and to align edges in the disparity maps and original images correctly. Note that there are many forms of such a loss [11, 12, 54, 51], we adopt the one in [11, 12], which is defined as:

$$\mathcal{L}_{smooth} = |\partial_x \hat{\mathcal{D}}^*| \exp^{-|\partial_x \mathrm{I}|} + |\partial_y \hat{\mathcal{D}}^*| \exp^{-|\partial_y \mathrm{I}|}, \quad (11)$$

where ∂_x and ∂_y are the image gradients, and $\hat{\mathcal{D}}^* = \hat{\mathcal{D}}/\bar{\mathcal{D}}$ is the mean-normalized disparity, where $\hat{\mathcal{D}} = 1/\hat{\mathcal{Z}}$.

Sparse disparity loss. In the case that SfM is adopted to pre-process the input images/videos to solve the scale ambiguity, we apply the sparse disparity loss to facilitate the depth/disparity predictions. Nonetheless, note that this is optional. In particular, SfM is not necessary and the sparse disparity loss is not applicable in datasets such as KITTI, where the scale is fixed to $s = 1$. We follow the log disparity style as [51, 6].

$$\begin{aligned} &\mathcal{L}_d = 0.5\mathcal{L}_d^{src} + 0.5\mathcal{L}_d^{tgt}, \quad \text{where} \\ &\mathcal{L}_d^{src} = \frac{1}{|\mathbf{P_s}|} \sum_{(x,y,z)\in\mathbf{P_s}} \left(\ln \frac{\hat{\mathcal{D}}_{src}(x,y)}{s} - \ln \frac{1}{z} \right), \\ &\mathcal{L}_d^{tgt} = \frac{1}{|\mathbf{P_t}|} \sum_{(x,y,z)\in\mathbf{P_t}} \left(\ln \frac{\hat{\mathcal{D}}_{tgt}(x,y)}{s} - \ln \frac{1}{z} \right). \end{aligned} \quad (12)$$

Note that we need to scale the disparity maps because the translation $\mathbf{t}$ is calibrated with s as shown in Sec. 3.3.1. The translation and depth should be calibrated together.

3.4. Our Relation to NeRF

Our MINE shares similar underlying representation, i.e., RGB and volume density at arbitrary position in the space. **Advantages**: 1) Our MINE generalizes to unseen scenes, while NeRF has to be optimized per scene. 2) To render a novel view, our MINE requires lesser network inferences (e.g. 32 network inferences), while NeRF requires millions of network inferences. **Limitations**: 1) Our MINE takes only one image as input and therefore it is impossible to reconstruct the whole object from all 360°. 2) MINE does not take viewing direction as input, therefore, it is not able to model complex view-dependent effects.

3.5. Our Relation to MPI

The MPI representation in [51] is a special case of our representation described in Sec. 3.1.

Proof. Instead of sampling with Eq. 3, we can simply set d_i as the bin edges given by:

$$d_i = d_n + (i-1)/N \cdot (d_f - d_n). \quad (13)$$

Additionally, we can define the alpha map at depth z_i as $\alpha_{z_i} : \mathbb{R}^2 \to \mathbb{R}^+$ shown in Eq. 15. Now, Eq. 2 can be rewritten into:

$$\hat{\mathrm{I}} = \sum_{i=1}^{N} T_i \alpha_{z_i} c_{z_i}, \text{ where } \quad T_i = \prod_{j=1}^{i-1} (1 - \alpha_{z_j}), \quad (14)$$

which is identical to the MPI compositing operation in [51] (c.f. Eq. 5 in [51]), where they directly predict:

$$\alpha_{z_i} = 1 - \exp(-\sigma_{z_i} \delta_{z_i}) \quad (15)$$

instead of σ_{z_i}. Note that δ_{z_i} is now a constant since d_i is set as the bin edges without random sampling. □

3.6. Our Relation to pixelNeRF and GRF

Our MINE is different from pixelNeRF [59] and GRF [50] by: **(a)** MINE directly models the frustum of the source camera, while both pixelNeRF and GRF model the entire 3D space. **(b)** MINE reconstructs the frustum of the source camera per plane, while pixelNeRF and GRF reconstruct the entire 3D space per ray. **(c)** Neither pixelNeRF nor GRF presents experiments on large scale real world datasets, while MINE presents results on large scale indoor / outdoor datasets, namely RealEstate10K, NYUv2 and KITTI.

A direct consequent of (a) (b) is that our MINE is significantly more efficient at inference. Both pixelNeRF and GRF render the output image pixel by pixel, and therefore the number of forward passes required is proportional to the spatial resolution of the output, the number of points along each ray, and the number of target views to render. On the contrary, since our MINE reconstructs the entire frustum of the source camera per plane, we only require N_{planes} forward passes of the fully-convolutional decoder to obtain the representation. The rendering for any novel view only requires an additional homography warping step. Detailed analysis is presented in the supplementary materials.

4. Experiments

For novel view synthesis, we perform both quantitative and qualitative comparisons with state-of-the-art methods on the RealEstate10K [62], flowers light field [47], and KITTI [8] datasets. To measure the quality of the generated images, we compute the Structural Similarity Index (SSIM) [63], PSNR, and the recently proposed LPIPS perceptual similarity [60]. We use an ImageNet-trained VGG16 [43] model when computing the LPIPS score. For depth estimation from single image, we perform evaluations on the iBims-1 [21] dataset and the NYU-Depth V2 [30] dataset.

4.1. View Synthesis on KITTI

Following the settings of [52, 51], we train our models on the 20 city sequences from the KITTI Raw dataset [8], and evaluates on another 4 city sequences. We fix the scale factor to 1 because only stereo pairs with a constant scale are utilized. During training, the left or right image is randomly taken as the source image and the target image. Following [51], we crop 5% from all sides of all images before computing the scores in testing. Quantitative comparisons with [51, 52] are presented in Tab. 1. Both our 32- and 64-plane models outperform these existing methods by a large

	Train Res.	N	Pre-trained	Depth Smoothess	LPIPS↓	SSIM↑	PSNR↑
MINE	384x128	32	N	Y	0.129	0.812	21.4
MINE	384x128	32	Y	N	0.123	0.816	21.6
MINE	384x128	32	Y	Y	0.122	0.815	21.6
MINE	384x128	64	Y	Y	0.117	0.818	21.6
MINE	384x128	256	Y	Y	**0.112**	**0.828**	**21.9**
Tulsiani et. al. [52]	768x256	NA	NA	NA	-	0.572	16.5
MPI [51]	768x256	32	NA	NA	-	0.733	19.5
MINE	768x256	32	Y	Y	0.112	**0.822**	21.4
MINE	768x256	64	Y	Y	**0.108**	0.820	21.3

Table 1. View synthesis on KITTI dataset. Note that [52] trains the model at 768×256 and tests at 384×128 to avoid cracks in the output, and [51] adopts this setting for comparison. We follow this setting and all our models are tested with resolution of 384×128.

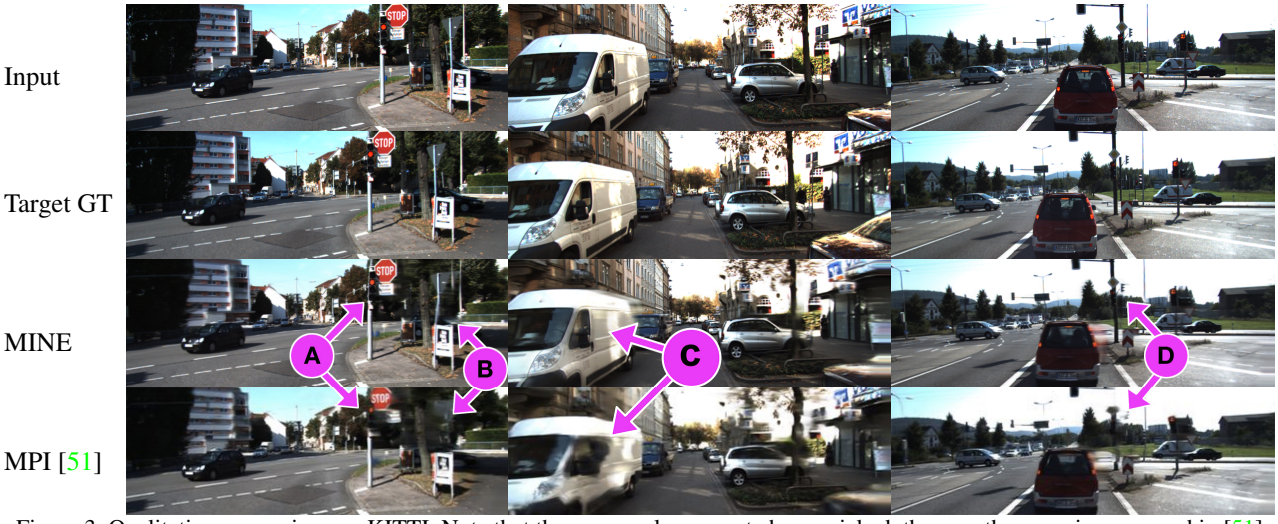

Figure 3. Qualitative comparison on KITTI. Note that these examples are not cherry-picked, they are the same images used in [51].

margin. Notably, we significantly improve the SSIM from 0.733 to 0.822 when compared to [51]. We also qualitatively demonstrate our superior view synthesis performance in Fig. 3. Compared to [51], we generate more realistic images with lesser artefacts and shape distortions. The visualization verifies our ability to model the geometry and texture of complex scenes.

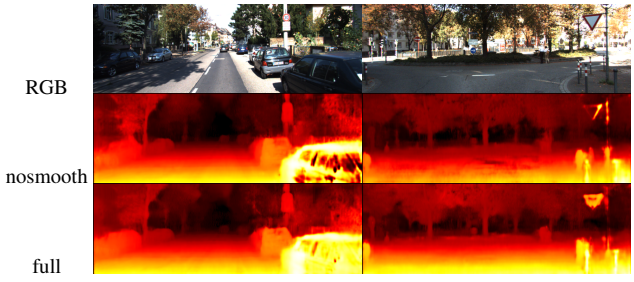

Figure 4. Effects of the edge-aware smoothness loss on KITTI.

Ablation Studies. As shown in Tab. 1, ablation studies are conducted on the KITTI dataset to validate some design choices. We observe that ImageNet pre-training for the encoder brings moderate improvements on all metrics. The edge-aware depth smoothness loss only brings marginal improvements quantitatively, but we qualitatively show in Fig. 4 that it enables the model to synthesize better disparity maps. More importantly, we see consistent improvements with increasing N. Since the model capacity remains the same as we vary N, the improvements can be attributed to the ability of the model to learn more complex scene geometry when sampling the depths more densely while training.

4.2. View Synthesis on RealEstate10K

RealEstate10K [62] is a large-scale dataset of walk-through videos that contains both indoor and outdoor scenes. The dataset consists of $> 70,000$ video sequences, which are pre-split into a training set and a test set. Each sequence contains the video frames and their corresponding camera intrinsic and extrinsic. To obtain the sparse 3D point clouds for scale-invariant learning and sparse depth supervision, we use COLMAP [38, 39] to perform SfM on each video sequence. For testing, we randomly sample 600 sequences from the official test split, and then we draw 5 frames from each sequence as the source frames. This gives us 3,000 source frames in total. Following [51], we choose the target frames to be 5 or 10 frames apart for each reference frame. Additionally, we randomly sample another

Method	LPIPS↓			SSIM↑			PSNR↑		
	$n=5$	$n=10$	$n=random$	$n=5$	$n=10$	$n=random$	$n=5$	$n=10$	$n=random$
SynSin [56]	-	-	-	-	-	0.74	-	-	22.31
MPI [51]	0.0967	0.1420	0.1761	0.8699	0.8124	0.7851	27.05	24.43	23.52
MINE ($N=32$)	0.0934	0.1346	0.1674	0.8970	0.8464	0.8172	**28.51**	**25.73**	**24.56**
MINE ($N=64$)	**0.0896**	**0.1280**	**0.1562**	**0.8974**	**0.8500**	**0.8219**	28.39	25.71	24.50

Table 2. Results on RealEstate10K [62]. ↑ denotes higher is better and ↓ means otherwise. n is the number of frames between the source and target frames. The results of SynSin are from the original paper, where they use the same test setting (target frames are chosen randomly from within 30 frames of the source frames) as ours, but a different set of test pairs. They also use a lower resolution of 256x256.

Method	Supervision	Dataset	NYU-Depth V2 [30]						iBims-1 [21]					
			rel↓	log10↓	RMS↓	$\sigma 1$↑	$\sigma 2$↑	$\sigma 3$↑	rel↓	log10↓	RMS↓	$\sigma 1$↑	$\sigma 2$↑	$\sigma 3$↑
DIW [2]	Depth	DIW	0.25	0.1	0.76	0.62	0.88	0.96	0.25	0.1	1	0.61	0.86	0.95
DIW [2]	Depth	DIW+NYU	0.19	0.08	0.6	0.73	0.93	0.98	0.19	0.08	0.8	0.72	0.91	0.97
MegaDepth [24]	Depth	Mega	0.24	0.09	0.72	0.63	0.88	0.96	0.23	0.09	0.83	0.67	0.89	0.96
MegaDepth [24]	Depth	Mega+DIW	0.21	0.08	0.65	0.68	0.91	0.97	0.2	0.08	0.78	0.7	0.91	0.97
3DKenBurns [32]	Depth	Mega+NYU+3DKenBurn	**0.08**	**0.03**	**0.3**	**0.94**	**0.99**	**1**	**0.1**	**0.04**	**0.47**	**0.9**	**0.97**	**0.99**
MiDaS v2.1 [37]	Depth	MiDaS 10 datasets	0.16	0.06	0.50	0.80	0.95	0.99	0.14	0.06	0.57	0.84	**0.97**	**0.99**
MPI [51]	RGB	RealEstate10K	0.15	0.06	0.49	0.81	0.96	0.99	0.21	0.08	0.85	0.7	0.91	0.97
MINE ($N=64$)	RGB	RealEstate10K	0.11	0.05	0.40	0.88	0.98	0.99	0.11	0.05	0.53	0.87	**0.97**	**0.99**

Table 3. Depth estimation results on iBims-1 and NYU-Depth V2. We significantly outperform MPI [51] which is also supervised on only RGB images and sparse depth, and achieves comparable performance with state-of-the-art methods that use dense depth supervision.

Figure 5. Qualitative comparison on RealEstate10K with [51]. Our MINE generates much more photo-realestic outputs than [51], it also inpaints the dis-occluded area much better (see hightlighted area in Row 3).

target frame from within 30 frames apart form the reference frame to create a more challenging setting.

In the RealEstate10K experiments, N is set to 32 or 64, $\lambda_{smooth} = 0.01$, and λ_d, λ_{rgb}, λ_{SSIM} are set to 1.0. The disparity range is $[1.0, 0.001]$ for both the 32- and 64-plane models. The input resolution is set to 384×256. We train our models on 48 NVIDIA V100 SXM2 GPUs. We use the Adam Optimizer [20] with an initial learning rate of 0.0002 for the encoder, and 0.001 for the decoder, we train the models for a total of 1,000,000 steps and the learning rate is decayed once at step 600,000. In training, we randomly sample the source frames, and sample the target frame within 30 frames apart from the source frames. For fair comparisons, we run the MPI [51] open-sourced models on our test set, and obtained similar scores as the original paper. As shown in Tab. 2, it is clear that MINE outperforms both [56, 51] in all 3 criteria by a large margin. We show again that increasing N from 32 to 64 gives better results, which is consistent with the ablation studies in KITTI. In Fig. 5, we show qualitative comparison with [51]. MINE generates sharp and realistic target images, while [51] produces unpleasant artefacts and distortions. In particular, we high-

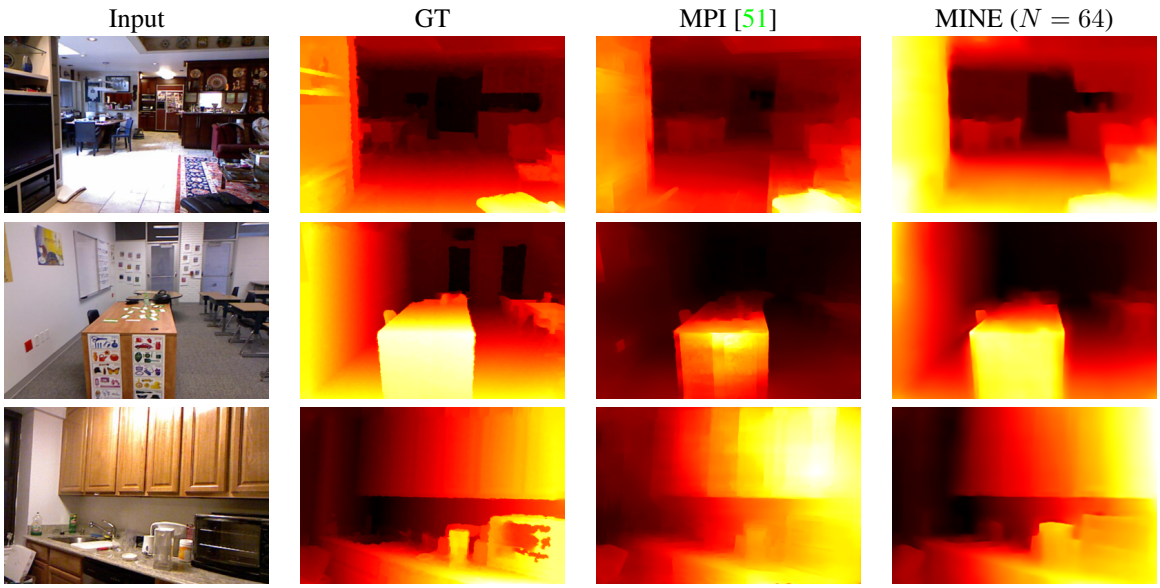

Figure 6. Qualitative comparison for disparity maps.

light our inpainting capacity with a yellow bounding box.

4.3. Depth Estimation on iBims-1 and NYU-V2

We evaluate our depth estimation on the iBims-1 [21] and NYU-Depth V2 [30] benchmarks. Both benchmarks contain indoor scenes with dense ground truth depths. We use the model trained with RealEstate10K to synthesize the disparity maps and measure our depth estimation performance. Following [51, 32], to solve the scale ambiguity of depth from single image, we scale and bias the depth predictions to minimize the $L2$ depth error before evaluation. We compare with Depth in the Wild [2], MegaDepth [24], 3DKenBurns [32] and MiDaS [37], which are state-of-the-art systems trained with ground truth depth supervision. We also compare with [51] which uses the same RGB video supervision as ours. Tab. 3 shows the quantitative results. Notably, even though MINE does not use any ground truth depth supervision in training, we achieve comparable performance as 3DKenBurns [32], and significantly outperforms the other methods by a large margin. We further show qualitative comparison with [51] in Fig. 6. We find that MPI is easily biased towards image textures and thus producing unpleasant artefacts in the disparity maps, while MINE is able to generate smooth and more accurate disparity maps for images with texture-rich surfaces.

4.4. View Synthesis on Flowers Light Fields

The Flowers light fields dataset [47] consists of 3,343 light fields captured with the Lytro Illum camera. Each light field has 14×14 angular samples and 376 spatial samples. Following [47] and [51], we use the central 8×8 grids in our experiments to avoid using the angular samples outside the aperture. In testing, the centre image of the 8×8 grid is the source image and the four corner ones are the target images. For comparison, we obtain the training and testing splits from [51]. Following their experiment settings, we randomly adjust the gamma from $[0.3, 0.7]$ in training and fixing it to 0.5 during testing. Since the scale is constant in this dataset, we set the scale factor to 1 following [51]. As shown in Tab. 4, MINE improves upon [47] and [51]. As expected, increasing N brings consistent improvements.

Method	LPIPS↓	SSIM↑	PSNR↑
Srinivasan et al, full [47]	-	0.822	28.1
Tucker et al. [51]	-	0.851	30.1
MINE ($N = 32$)	0.1603	0.868	30.2
MINE ($N = 64$)	**0.1559**	**0.872**	**30.3**

Table 4. View Synthesis on flowers light fields.

5. Conclusion

We propose MINE that is a continuous depth generalization of MPI by introducing NeRF. Given a single image, we jointly do a dense reconstruction of the camera frustum and inpainting of the occluded contents. We render our reconstructed frustum into novel view RGB images and depth maps with differentiable rendering. Extensive experiments show that our method significantly outperforms existing state-of-the-art single-image view synthesis methods, and achieves near state-of-the-art performance on depth estimation without dense ground truth depth supervision.

Acknowledgment This work is supported in part by the Singapore MOE Tier 2 grant MOE-T2EP20120-0011.

References

[1] Matan Atzmon, Niv Haim, Lior Yariv, Ofer Israelov, Haggai Maron, and Yaron Lipman. Controlling neural level sets. In *Advances in Neural Information Processing Systems*, pages 2032–2041, 2019. 2

[2] Weifeng Chen, Zhao Fu, Dawei Yang, and Jia Deng. Single-image depth perception in the wild. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016. 7, 8

[3] Zhiqin Chen and Hao Zhang. Learning implicit fields for generative shape modeling. *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 2

[4] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). In Yoshua Bengio and Yann LeCun, editors, *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016. 14

[5] Michael Cohen, Steven J. Gortler, Richard Szeliski, Radek Grzeszczuk, and Rick Szeliski. The lumigraph. Association for Computing Machinery, Inc., August 1996. 2

[6] David Eigen, Christian Puhrsch, and Rob Fergus. Depth map prediction from a single image using a multi-scale deep network. *arXiv preprint arXiv:1406.2283*, 2014. 5

[7] Ravi Garg, Vijay Kumar B.G., Gustavo Carneiro, and Ian Reid. Unsupervised cnn for single view depth estimation: Geometry to the rescue. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *Computer Vision – ECCV 2016*, pages 740–756, Cham, 2016. Springer International Publishing. 2

[8] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *International Journal of Robotics Research (IJRR)*, 2013. 5

[9] Kyle Genova, Forrester Cole, Avneesh Sud, Aaron Sarna, and Thomas Funkhouser. Local deep implicit functions for 3d shape. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 2

[10] Kyle Genova, Forrester Cole, Daniel Vlasic, Aaron Sarna, William T. Freeman, and Thomas Funkhouser. Learning shape templates with structured implicit functions. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019. 2

[11] Clément Godard, Oisin Mac Aodha, and Gabriel J. Brostow. Unsupervised monocular depth estimation with left-right consistency. In *CVPR*, 2017. 1, 2, 5, 12

[12] Clément Godard, Oisin Mac Aodha, Michael Firman, and Gabriel J. Brostow. Digging into self-supervised monocular depth prediction. October 2019. 1, 2, 4, 5, 12, 13

[13] Steven Gortler, Chris Buehler, Michael Bosse, Leonard Mcmillan, and Michael Cohen. Unstructured lumigraph rendering. *Proceedings of SIGGRAPH 2001*, 01 2001. 2

[14] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second edition, 2004. 3

[15] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016. 4, 13

[16] Philipp Henzler, Niloy J. Mitra, and Tobias Ritschel. Learning a neural 3d texture space from 2d exemplars. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 2

[17] Chiyu "Max" Jiang, Avneesh Sud, Ameesh Makadia, Jingwei Huang, Matthias Niessner, and Thomas Funkhouser. Local implicit grid representations for 3d scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 2

[18] James T Kajiya and Brian P Von Herzen. Ray tracing volume densities. *ACM SIGGRAPH computer graphics*, 18(3):165–174, 1984. 3

[19] Abhishek Kar, Christian Häne, and Jitendra Malik. Learning a multi-view stereo machine. 2017. 2

[20] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. 7

[21] 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, *European Conference on Computer Vision Workshop (ECCV-WS)*, pages 331–348. Springer International Publishing, 2018. 5, 7, 8, 13

[22] Kiriakos N. Kutulakos and Steven M. Seitz. A theory of shape by space carving. Technical report, USA, 1998. 2

[23] Marc Levoy and Pat Hanrahan. Light field rendering. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '96, page 31–42, New York, NY, USA, 1996. Association for Computing Machinery. 2

[24] Zhengqi Li and Noah Snavely. Megadepth: Learning single-view depth prediction from internet photos. In *Computer Vision and Pattern Recognition (CVPR)*, 2018. 7, 8

[25] Reza Mahjourian, Martin Wicke, and Anelia Angelova. Unsupervised learning of depth and ego-motion from monocular video using 3d geometric constraints. In *CVPR*, 2018. 2

[26] 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 *CVPR*, 2021. 2

[27] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019. 2

[28] Ben Mildenhall, Pratul P. Srinivasan, Rodrigo Ortiz-Cayon, Nima Khademi Kalantari, Ravi Ramamoorthi, Ren Ng, and Abhishek Kar. Local light field fusion: Practical view synthesis with prescriptive sampling guidelines. *ACM Transactions on Graphics (TOG)*, 2019. 2

[29] 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 *ECCV*, 2020. 1, 2, 3, 4, 12

[30] Pushmeet Kohli Nathan Silberman, Derek Hoiem and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *ECCV*, 2012. 5, 7, 8

[31] Michael Niemeyer, Lars Mescheder, Michael Oechsle, and Andreas Geiger. Differentiable volumetric rendering: Learning implicit 3d representations without 3d supervision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 2

[32] Simon Niklaus, Long Mai, Jimei Yang, and Feng Liu. 3d ken burns effect from a single image. *ACM Transactions on Graphics*, 38(6):184:1–184:15, 2019. 2, 7, 8

[33] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 2

[34] Keunhong Park, Utkarsh Sinha, Jonathan T. Barron, Sofien Bouaziz, Dan B Goldman, Steven M. Seitz, and Ricardo Martin-Brualla. Deformable neural radiance fields, 2020. 2

[35] Songyou Peng, Michael Niemeyer, Lars Mescheder, Marc Pollefeys, and Andreas Geiger. Convolutional occupancy networks. In *European Conference on Computer Vision (ECCV)*, Cham, Aug. 2020. Springer International Publishing. 2

[36] Nasim Rahaman, Aristide Baratin, Devansh Arpit, Felix Draxler, Min Lin, Fred Hamprecht, Yoshua Bengio, and Aaron Courville. On the spectral bias of neural networks. In *International Conference on Machine Learning*, pages 5301–5310. PMLR, 2019. 4

[37] René Ranftl, Katrin Lasinger, David Hafner, Konrad Schindler, and Vladlen Koltun. Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2020. 7, 8

[38] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 1, 4, 6

[39] Johannes Lutz Schönberger, Enliang Zheng, Marc Pollefeys, and Jan-Michael Frahm. Pixelwise view selection for unstructured multi-view stereo. In *European Conference on Computer Vision (ECCV)*, 2016. 1, 4, 6

[40] Steven M. Seitz and Charles R. Dyer. Photorealistic scene reconstruction by voxel coloring. In *Proceedings of the 1997 Conference on Computer Vision and Pattern Recognition (CVPR '97)*, CVPR '97, page 1067, USA, 1997. IEEE Computer Society. 2

[41] Jonathan Shade, Steven Gortler, Li-wei He, and Rick Szeliski. Layered depth images. pages 231–242. Association for Computing Machinery, Inc., July 1998. 1, 2

[42] Meng-Li Shih, Shih-Yang Su, Johannes Kopf, and Jia-Bin Huang. 3d photography using context-aware layered depth inpainting. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 2

[43] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015. 5

[44] Vincent Sitzmann, Justus Thies, Felix Heide, Matthias Nießner, Gordon Wetzstein, and Michael Zollhöfer. Deepvoxels: Learning persistent 3d feature embeddings. In *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 2019. 2

[45] Vincent Sitzmann, Michael Zollhoefer, and Gordon Wetzstein. Scene representation networks: Continuous 3d-structure-aware neural scene representations. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. 2

[46] Pratul P. Srinivasan, Richard Tucker, Jonathan T. Barron, Ravi Ramamoorthi, Ren Ng, and Noah Snavely. Pushing the boundaries of view extrapolation with multiplane images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 2

[47] 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 *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct 2017. 2, 5, 8, 13

[48] Rick Szeliski and Polina Golland. Stereo matching with transparency and matting. *International Journal of Computer Vision*, 32(1):45–61, May 1999. 2

[49] Matthew Tancik, Pratul Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 7537–7547. Curran Associates, Inc., 2020. 4

[50] Alex Trevithick and Bo Yang. Grf: Learning a general radiance field for 3d scene representation and rendering. In *arXiv:2010.04595*, 2020. 2, 5, 12

[51] Richard Tucker and Noah Snavely. Single-view view synthesis with multiplane images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 1, 2, 3, 4, 5, 6, 7, 8, 12, 13

[52] Shubham Tulsiani, Richard Tucker, and Noah Snavely. Layer-structured 3d scene inference via view synthesis. In *ECCV*, 2018. 1, 2, 5, 6

[53] Sudheendra Vijayanarasimhan, Susanna Ricco, Cordelia Schmid, Rahul Sukthankar, and Katerina Fragkiadaki. Sfm-net: Learning of structure and motion from video. 04 2017. 1, 2

[54] Chaoyang Wang, José Miguel Buenaposada, Rui Zhu, and Simon Lucey. Learning depth from monocular videos using direct methods. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 2, 5

[55] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004. 4

[56] Olivia Wiles, Georgia Gkioxari, Richard Szeliski, and Justin Johnson. Synsin: End-to-end view synthesis from a single image. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 2, 7

[57] Junyuan Xie, Ross 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*, pages 842–857, Cham, 2016. Springer International Publishing. 2

[58] Lior Yariv, Yoni Kasten, Dror Moran, Meirav Galun, Matan Atzmon, Ronen Basri, and Yaron Lipman. Multiview neural surface reconstruction by disentangling geometry and appearance, 2020. 2

[59] Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. pixelnerf: Neural radiance fields from one or few images, 2020. 2, 5, 12

[60] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018. 5

[61] Tinghui Zhou, Matthew Brown, Noah Snavely, and David G. Lowe. Unsupervised learning of depth and ego-motion from video. In *CVPR*, 2017. 1, 2

[62] Tinghui Zhou, Richard Tucker, John Flynn, Graham Fyffe, and Noah Snavely. Stereo magnification: Learning view synthesis using multiplane images. In *SIGGRAPH*, 2018. 2, 3, 5, 6, 7, 13

[63] Zhou Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004. 5, 12

MINE: Towards Continuous Depth MPI with NeRF for Novel View Synthesis - Supplementary Materials

Jiaxin Li[1*], Zijian Feng[1*], Qi She[1], Henghui Ding[1], Changhu Wang[1], Gim Hee Lee[2]

[1]ByteDance, [2]National University of Singapore

A. MINE vs. pixelNeRF and GRF

There are two recent works: pixelNeRF [59] and GRF [50] that condition NeRF [29] on input image(s). pixelNeRF [59] first extracts a feature map from a given input image. A feature vector at each query position x and viewing direction d is subsequently sampled from the feature map via projection and bilinear interpolation. The sampled feature vector then serves as an additional input to the MLP along with x and d to predict the RGB-σ values. The rendering process is the same as NeRF. GRF [50] follows the same principles, but it assumes multiple views of a scene are available at test time.

Our MINE is different from pixelNeRF and GRF in two aspects:

- MINE directly models the frustum of the source camera, while both pixelNeRF and GRF model the entire 3D space.
- MINE reconstructs the frustum of the source camera per plane, while pixelNeRF and GRF reconstruct the entire 3D space per ray.

A direct consequent of these differences is that our MINE is significantly more efficient. Both pixelNeRF and GRF render the output image pixel by pixel, and therefore the number of forward passes required is proportional to the spatial resolution of the output, the number of points along each ray, and the number of target views to render. On the contrary, since our MINE reconstructs the entire frustum of the source camera per plane, we only require N_{planes} forward passes of the fully-convolutional decoder to obtain the representation. Furthermore, the rendering for each novel view only requires an additional homography warping step.

More concretely, let us denote the output resolution as $H \times W$. We further denote the number of points along each ray from pixelNeRF and GRF as N_{points}, and the number of planes from our MINE as N_{planes}. The number of network forward passes P required for these methods are:

$$\begin{aligned} P_{\text{pixelNeRF}} &= 1 + N_{\text{targets}} \times N_{\text{points}} \times H \times W, \\ P_{\text{GRF}} &= N_{\text{views}} + N_{\text{targets}} \times N_{\text{points}} \times H \times W, \\ P_{\text{MINE}} &= 1 + N_{\text{planes}}, \end{aligned} \tag{16}$$

where N_{targets} denotes the number of novel views.

All three methods listed above utilize the encoder-decoder structure to condition on the input image(s). pixelNeRF and our MINE takes single image as input, and then the encoder is forwarded only 1 time. In contrast, GRF takes multiple images as input, and thus requires N_{views} encoder inferences.

Note that for pixelNeRF and GRF, $N_{\text{points}} \times H \times W$ times of decoder (MLP) inferences are required for **each** target view. On the other hand, our MINE reconstructs the frustum using N_{planes} decoder (Fully Convolutional Network) inferences. After the reconstruction, only homography warping is required to render into any target view. Consequently, the complexity of our MINE is independent of N_{targets}, while the complexity of pixelNeRF and GRF is proportional to N_{targets}.

Also note that our method does not take viewing direction as inputs, but we argue that the viewing directions can be easily integrated into our framework by concatenating the per-ray viewing directions with the output feature maps at the target view (after warping), and then using a lightweight fully convolutional network to predict the view-dependent radiance. This only adds N_{views} network inferences in total, which is still significantly faster than pixelNeRF and GRF.

The efficiency of our MINE also allows for more flexible training strategies. Since our MINE renders the full target image and disparity map in training time, it is possible to impose dense supervision signals, e.g. SSIM [63] and edge-aware smoothness loss [12, 11, 51]. We argue that these dense supervision signals are helpful for generalizing to real-world large-scale datasets, as verified empirically by our extensive experiments. Due to their inefficiency, it is infeasible for NeRF-like methods to render the full image at training time.

Lastly, neither pixelNeRF nor GRF presents experiments with large scale real-worlds data. Our MINE is verified with well-known datasets like KITTI, NYU-V2, RealEstate10k, etc.

B. Additional Implementation Details

Network architecture. Our encoder is a standard ResNet50 [15], we take the outputs of [conv1, layer1, layer2, layer3, layer4] as the final output of the encoder. We give a complete description of our decoder architecture in Table 5. The decoder is the same as the depth decoder in [12], except that we add two additional downsampling blocks and two upsampling blocks to increase the receptive fields of the network, and the output of the network is a 4-channel RGB-σ image. The RGB output is produced by a Sigmoid layer, and σ is produced by taking the absolute value of the last channel of the output. We adopt the multi-scale training strategy in [12] with the exception that $\mathcal{L}_{L1}$ and $\mathcal{L}_{SSIM}$ are only applied on output1 and $\mathcal{L}_{smooth}$ is applied on all [output1, output2, output3, output4]. Note that our method is not restricted to specific network architecture.

Pre-processing for Flowers Light Fields. For the Flower Light Fields dataset [47], we set the disparity range to be $[3.0, 0.03]$. Since this dataset was taken with the Lytro Illum camera, there is a shift in the principle point between different views. In this dataset, all light fields are taken with the same camera, but the shifts in the principle points could vary across different scenes. Since there is no metadata to indicate the amount of the shifts, we follow [51] and make it constant. Specifically, we set the camera intrinsics as follows:

$$\begin{aligned} f_x &= 0.868056, \quad f_y = 1.250000, \\ c_{x_{ij}} &= 0.5 + 0.002667 * i, \\ c_{y_{ij}} &= 0.5 + 0.002667 * j, \end{aligned} \tag{17}$$

where $[i, j]$ is the index in the extracted 8×8 grid and $[0, 0]$ denotes the top left view. Since this dataset was captured with a light field camera, in addition to the principle point shift, there is a translation between different views and there is no rotation. In training and testing, same as [51], we set the distance between adjacent grids to be 0.00128.

C. Additional Qualitative Results

Supplementary image results. We include additional qualitative results for KITTI (Figure 7), RealEstate10K [62] (Figure 8) and Flowers Light Fields (Figure 9). Our method generalizes well to a wide range of real-world scenes, including outdoor and indoor scenes, and flowers with complex geometry. All scenes are unseen in training.

Supplementary video results. We also include supplementary videos results (uploaded separately) for the RealEstate10K, KITTI and iBims-1 [21] datasets, covering both outdoor scenes and indoor scenes with complex geometries and textures. For each scene, we include both the RGB videos and the videos of disparity maps. Given a single image as input, we generate the video by rendering into multiple novel views. All scenes are unseen during training. We demonstrate that even under large camera motion, our MINE is still able to generate temporally consistent realistic images, and smooth and accurate disparity maps.

layer	k	in-channels	out-channels	input	activation
downconv1	1	2048	512	encoder_layer4	ELU [4]
downconv2	3	512	256	downconv1	ELU
upconv1_extra	3	256	256	downconv2	ELU
upconv2_extra	1	256	2048	upconv1_extra	ELU
upconv5	3	2048 + 21	256	cat(upconv2_extra, disparity_encoding)	ELU
iconv5	3	256 + 1024 + 21	256	cat(upconv5, encoder_layer3, disparity_encoding)	ELU
upconv4	3	256	128	iconv5	ELU
iconv4	3	128 + 512 + 21	128	cat(upconv4, encoder_layer2, disparity_encoding)	ELU
output4	3	128	4	iconv4	Sigmoid (for RGB) and abs (for σ)
upconv3	3	128	64	iconv4	ELU
iconv3	3	64 + 256 + 21	64	cat(upconv3, encoder_layer1, disparity_encoding)	ELU
output3	3	64	4	iconv3	Sigmoid (for RGB) and abs (for σ)
upconv2	3	64	32	iconv3	ELU
iconv2	3	32 + 64 + 21	32	cat(upconv2, encoder_conv1, disparity_encoding)	ELU
output2	3	32	4	iconv2	Sigmoid (for RGB) and abs (for σ)
upconv1	3	32	16	iconv2	ELU
iconv1	3	16	16	upconv1	ELU
output1	3	16	4	iconv1	Sigmoid (for RGB) and abs (for σ)

Table 5. Network architecture for our depth decoder. All upconv blocks consist of a convolution layer, a batch normalization layer and an activation layer as specified in the table, followed by a $2\times$ nearest neighbour upsampling. The downconv blocks consist of a max pooling layer of stride 2, a convolution layer followed by an activation layer.

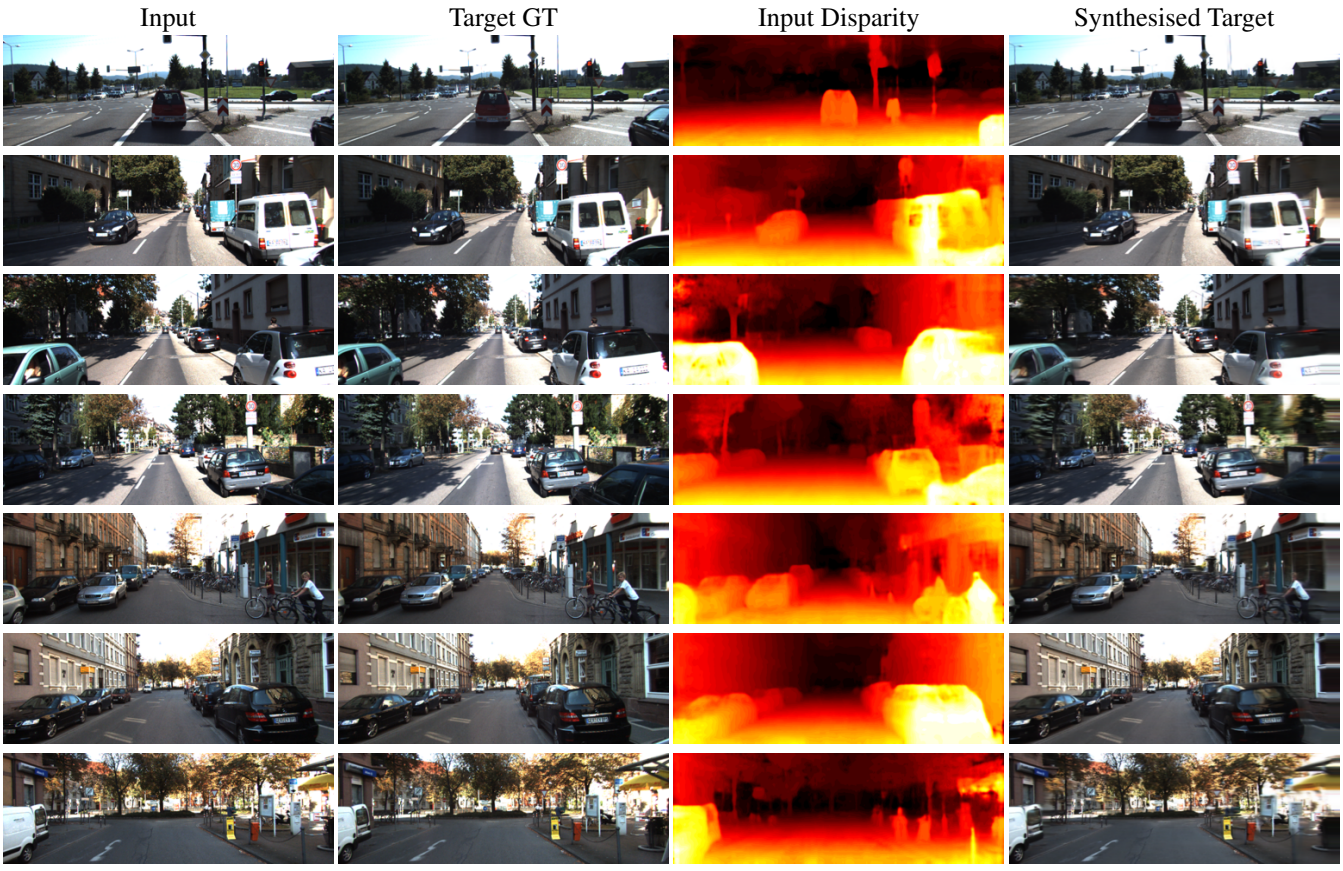

Figure 7. Qualitative results for KITTI.

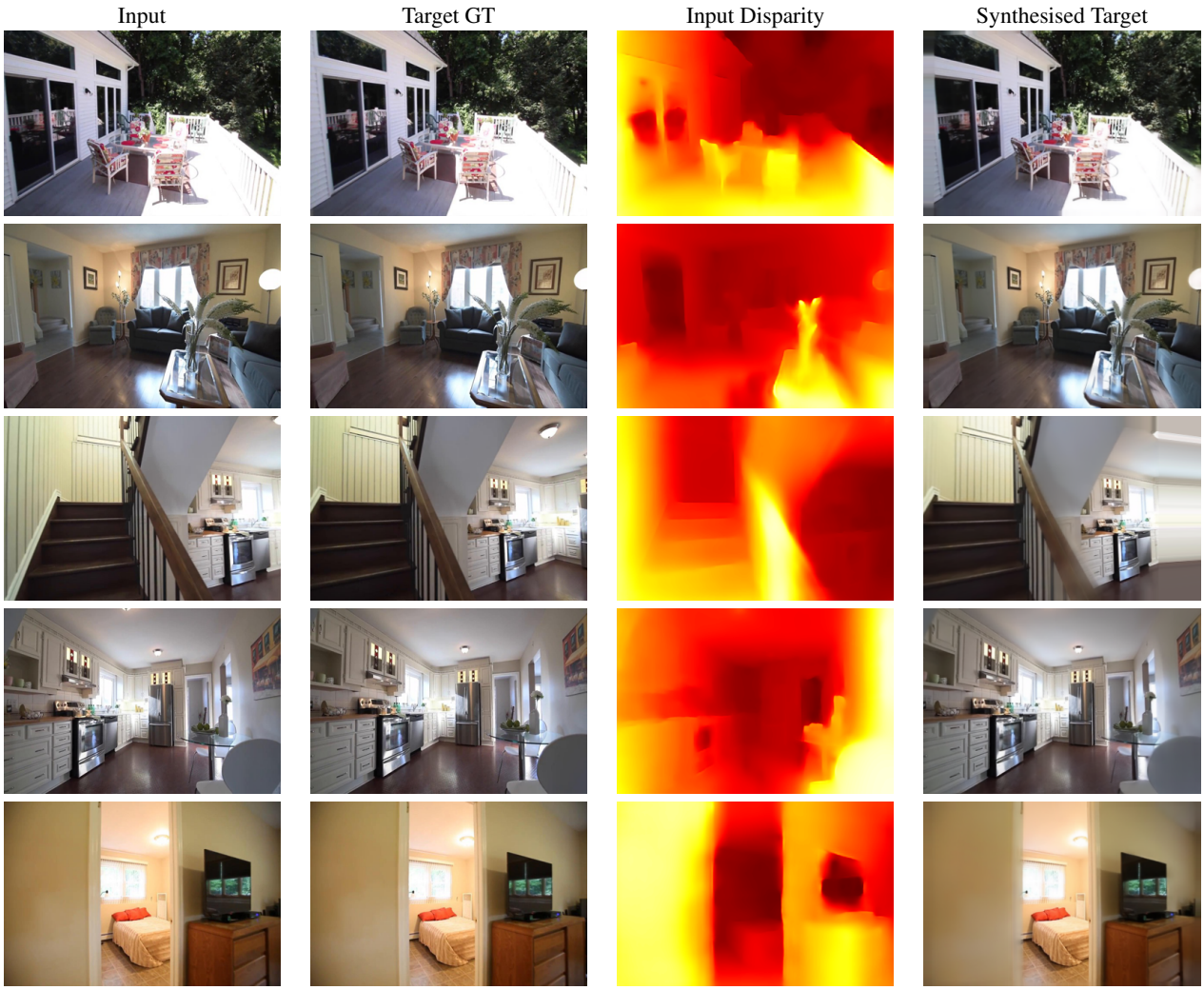

Figure 8. Qualitative results for RealEstate10K.

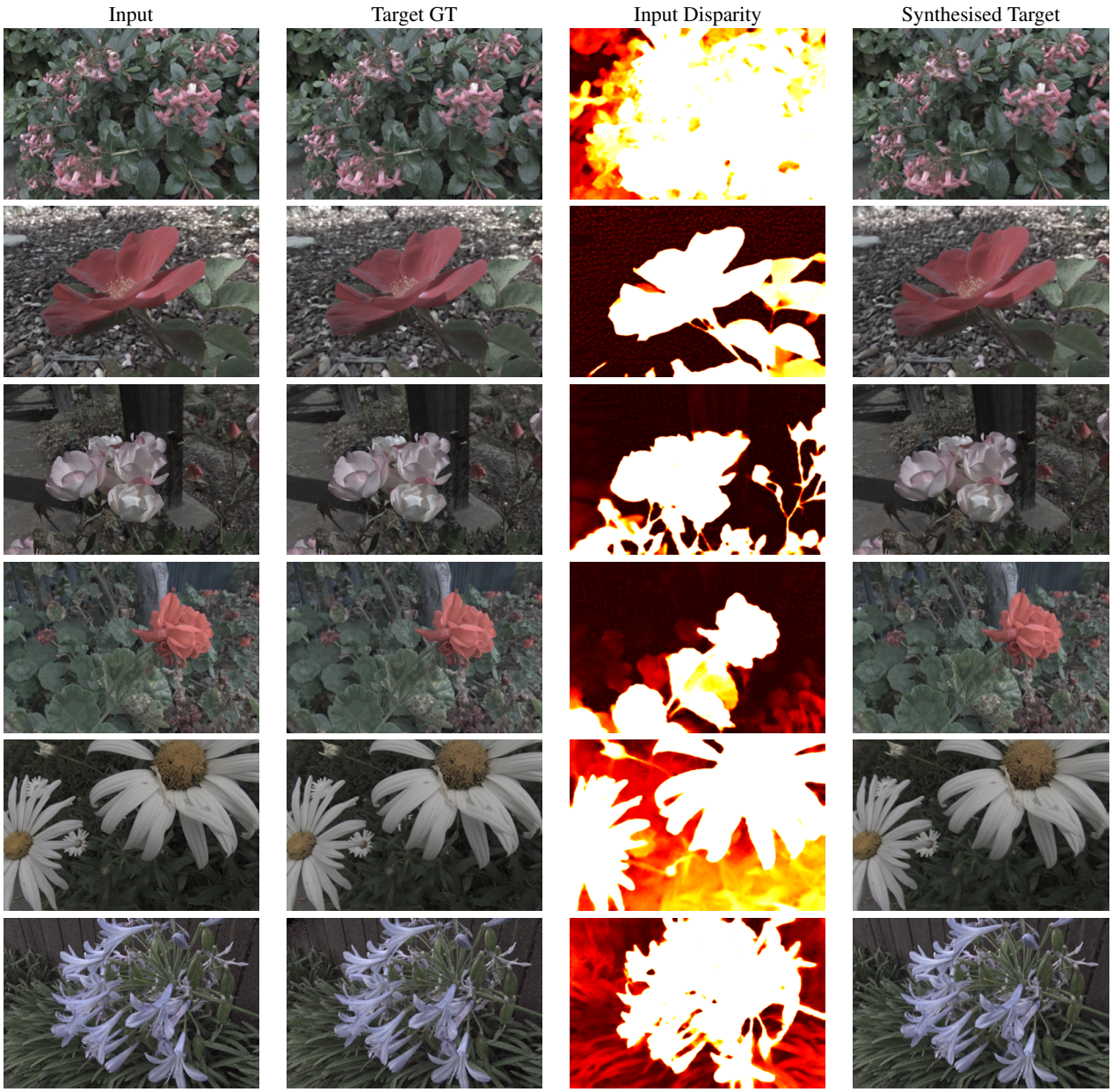

Figure 9. Qualitative results for Flowers Light Fields.