Supplementary Material – 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

A. MINE vs. pixelNeRF and GRF

There are two recent works: pixelNeRF [10] and GRF [8] that condition NeRF [6] on input image(s). pixelNeRF [10] 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 [8] 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:

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

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 [12] and edge-aware smoothness loss [3, 2, 9]. We argue that these dense supervision signals are helpful for generalizing

*Equal contribution.

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 [4], 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 1. The decoder is the same as the depth decoder in [3], 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 [3] 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 [7], 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 [9] and make it constant. Specifically, we set the camera intrinsics as follows:

$$\begin{aligned} f_x &= 0.868056, \quad f_y = 1.250000, \\ c_{x_{ij}} &= 0.5 + 0.002667 * i, \\ c_{y_{ij}} &= 0.5 + 0.002667 * j, \end{aligned} \tag{2}$$

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 [9], 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 1), RealEstate10K [11] (Figure 2) and Flowers Light Fields (Figure 3). 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 [5] 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 [1]
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 1. 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.

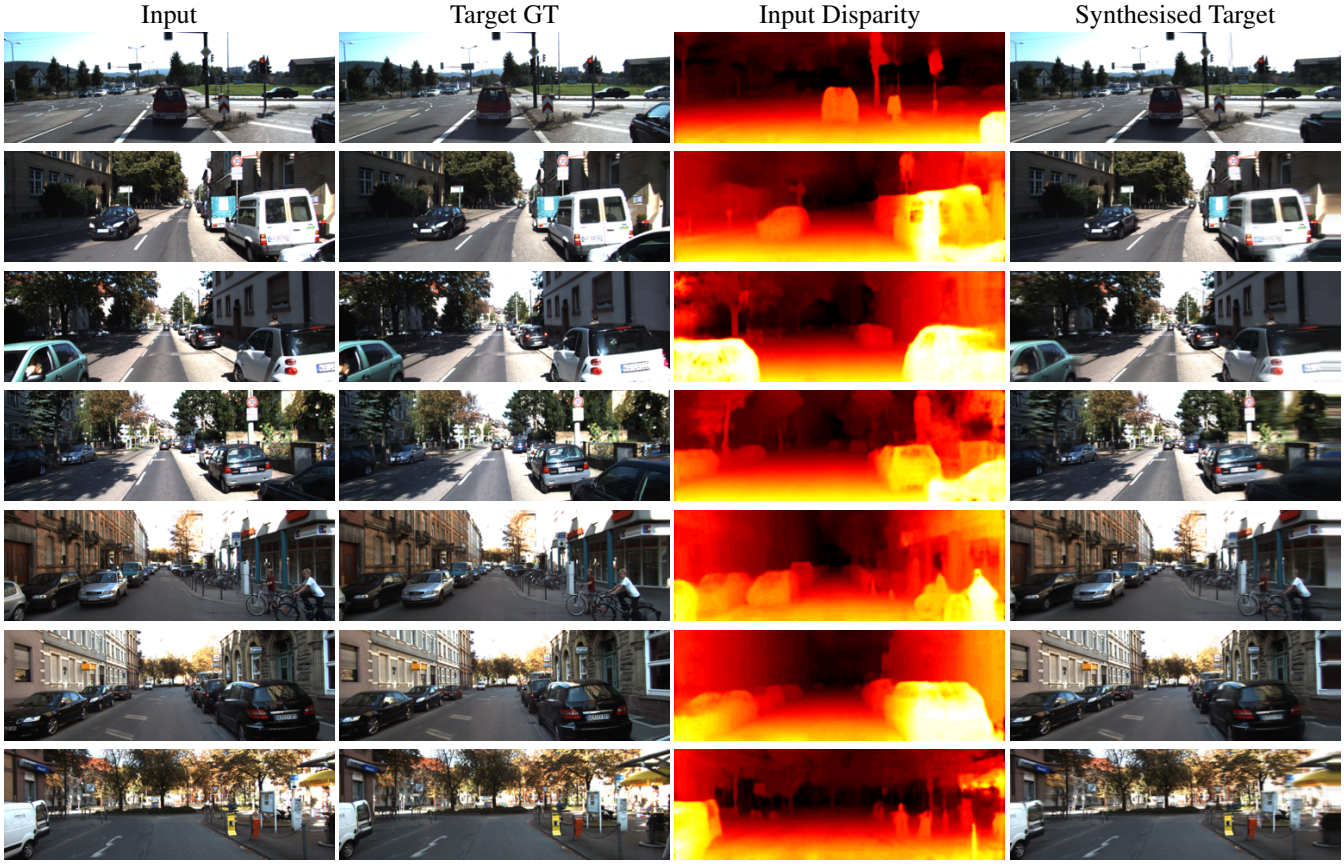

Figure 1. Qualitative results for KITTI.

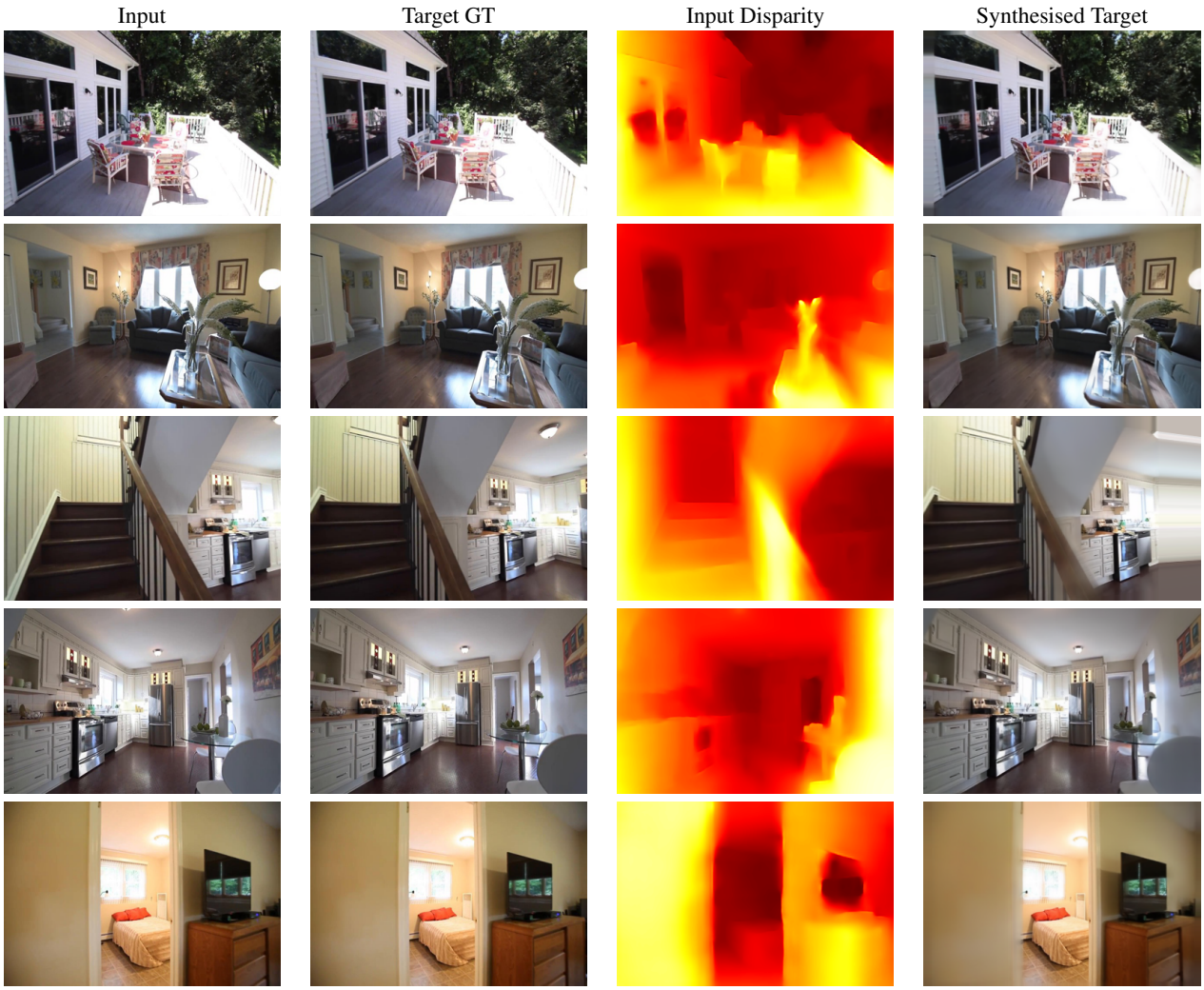

Figure 2. Qualitative results for RealEstate10K.

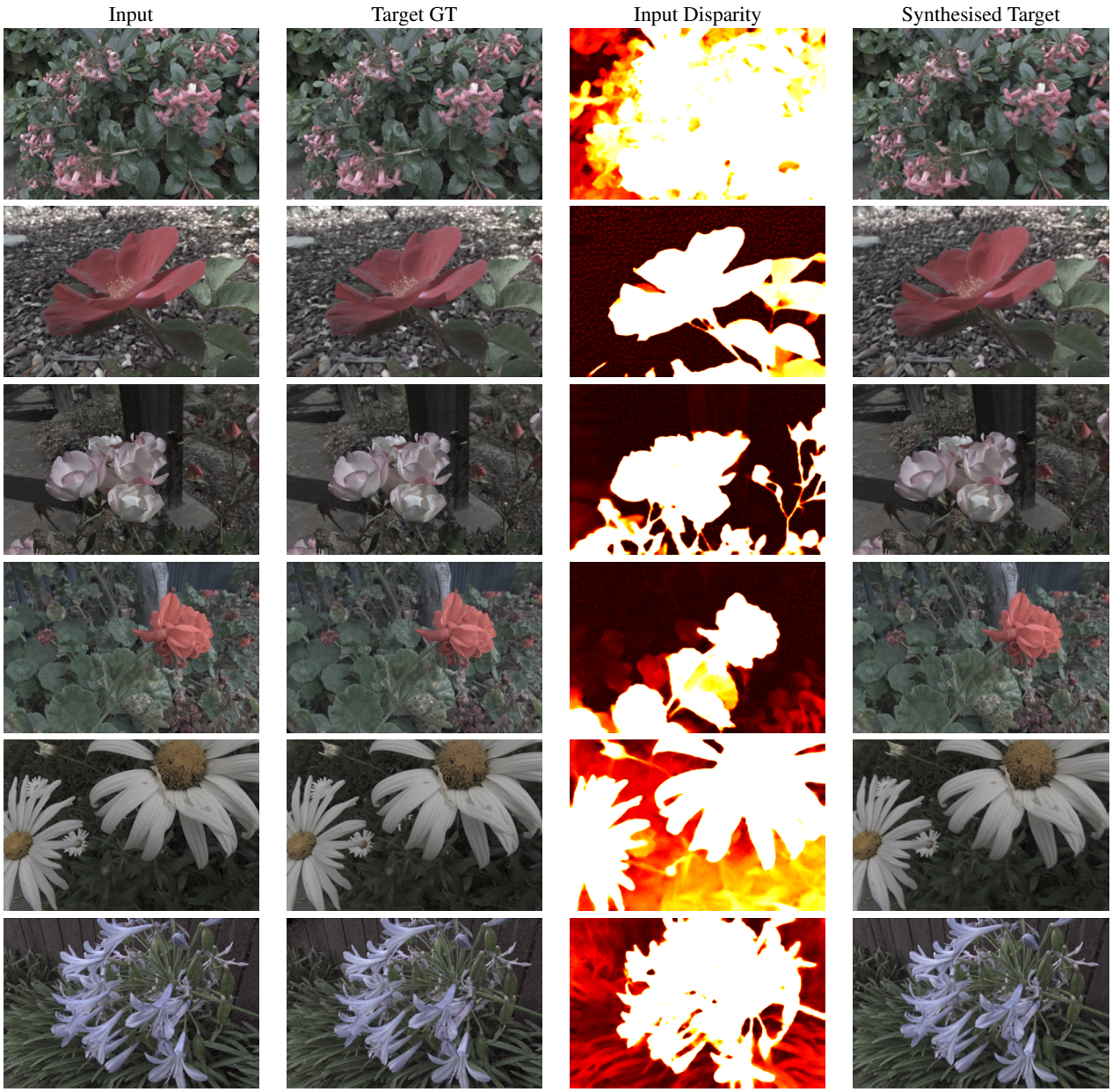

Figure 3. Qualitative results for Flowers Light Fields.

References

[1] 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. 3

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

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

[4] 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. 2

[5] 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. 2

[6] 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

[7] 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

[8] Alex Trevithick and Bo Yang. Grf: Learning a general radiance field for 3d scene representation and rendering. In *arXiv:2010.04595*, 2020. 1

[9] 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

[10] Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. pixelnerf: Neural radiance fields from one or few images, 2020. 1

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

[12] 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. 1