Supplementary Material for Efficient Depth-Guided Urban View Synthesis

Sheng Miao[1*], Jiaxin Huang[1*], Dongfeng Bai[2], Weichao Qiu[2], Bingbing Liu[2], Andreas Geiger[3,4], and Yiyi Liao[1✉]

[1]Zhejiang University [2]Huawei Noah's Ark Lab [3]University of Tübingen
[4]Tübingen AI Center

Abstract. In this **supplementary document**, we initially present an in-depth overview of our network architecture in Sec. A. Subsequently, we elaborate on our implementation details in Sec. B, encompassing the random masking strategy, the sampling strategy, and the point cloud accumulation configurations. Following this, we introduce the baselines for comparison in Sec. C, and discuss the datasets utilized in our experiments in Sec. D. Finally, we present additional experimental results in Sec. E. Besides, the **supplementary video** showcases our feed-forward and fine-tuned results across various scenes, along with novel view renderings compared against multiple baselines.

A Network Architecture

A.1 Modulation-based CNN

Our 3D Spatially-Adaptive Normalization convolutional neural network (SPADE CNN) modulates the feature volume generation at multiple resolutions. It takes a single voxelized point cloud with dimension $64 \times 128 \times 256$ as input. The appearance information of the input point cloud is injected into the feature volume at each resolution. Fig. 1 presents the detailed network architecture.

A.2 Foreground Network Architecture

The feature volume generated by the 3D SPADE CNN provides both geometric priors and appearance information for foreground contents. Based on a feature vector $\mathbf{f}_{\text{fg}}^{\text{3D}} \in \mathbb{R}^{16}$ extracted from this feature volume, our density decoder g_θ predicts the density value. In order to capture high-frequency appearance during feedforward inference, the color decoder h_θ combines the 3D feature $\mathbf{f}_{\text{fg}}^{\text{3D}} \in \mathbb{R}^{16}$ and 2D features queried from nearest views $\mathbf{f}_{\text{fg}}^{\text{2D}} \in \mathbb{R}^{9}$ for color prediction. In addition, the color decoder further takes the positional encoding of $\mathbf{x}$ as input, setting the number of frequencies to 10 following [5]. Furthermore, we inject viewing direction $\mathbf{d} \in \mathbb{R}^3$ and a per-frame appearance embedding $\mathbf{w} \in \mathbb{R}^{32}$. We present the detailed network architecture in Tab. 1.

* Equally contributed. ✉ Corresponding author.

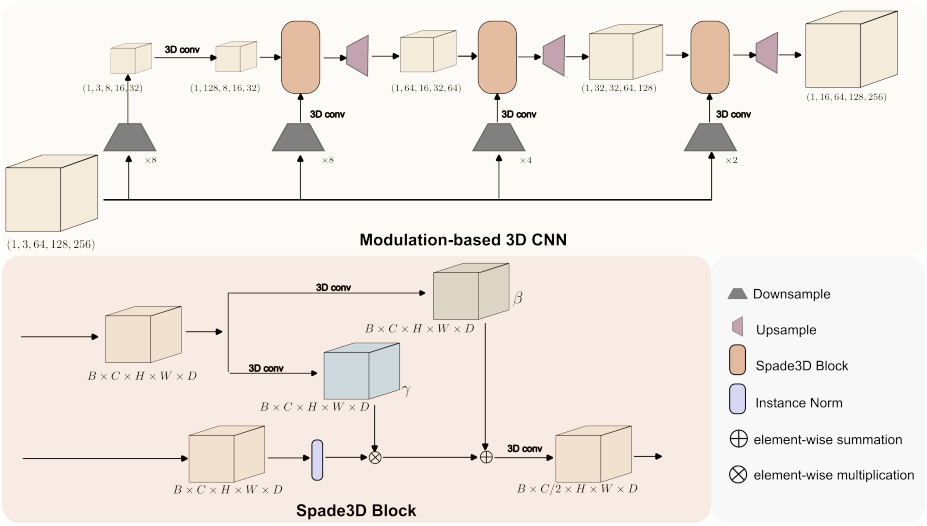


Fig. 1: Architecture of our 3D SPADE CNN.

Density Decoder			
Layer	**in channels**	**out channels**	**description**
LinearRelu_0	16	64	in: $\mathbf{f}_{\text{fg}}^{\text{3D}}$
LinearRelu_1	64	1	out: σ_{fg}
Color Decoder			
Layer	**in channels**	**out channels**	**description**
LinearRelu_0	16+9+63	128	in: $\mathbf{f}_{\text{fg}}^{\text{3D}}, \mathbf{f}_{\text{fg}}^{\text{2D}}, \gamma(\mathbf{x})$
$\text{LinearRelu}_{i=1,2}$	128	128	/
LinearRelu_3	128+32+3	128	in: $+\ \mathbf{d}, \mathbf{w}$
LinearRelu_4	128	64	/
LinearRelu_5	64	3	out: $\mathbf{c}_{\text{fg}}$

Table 1: Density and Color Decoder Architectures. From top to bottom: two-layer MLP for density prediction and six-layer MLP for color prediction.

A.3 Background and Sky Network Architectures

We also build the generalizable model for the background and sky region via the image-based rendering paradigm as we mentioned in the main paper. We now show the details of the background and sky network details in Tab. 2.

B Implementation Details

B.1 Input Volume Random Masking

Our target is to perform novel view synthesis from sparse urban images. When provided with sparse images, the accumulated point clouds are often incomplete

Background network architecture			
Layer	**in channels**	**out channels**	**description**
LinearRelu_0	63+9	128	in: $\mathbf{f}_{\text{fg}}^{\text{2D}}, \gamma(\mathbf{x})$
$\text{LinearRelu}_{i=1,2,3}$	128	128	/
LinearRelu_4	128	16	out: σ_{bg}, $\mathbf{f_e}$
LinearRelu_5	15+3	64	/
LinearRelu_6	64	64	/
LinearRelu_7	64	3	out: $\mathbf{c}_{\text{bg}}$
Sky network architecture			
Layer	**in channels**	**out channels**	**description**
LinearRelu_0	9+3	3	in: $\mathbf{f}_{\text{fg}}^{\text{2D}}, \mathbf{d}$; out: $\mathbf{c}_{\text{sky}}$

Table 2: Background and Sky Network Architectures. The $\mathbf{f_e} \in \mathbb{R}^{\mathbf{15}}$ represents the geometric embedding vector and serves as the input for background color prediction.

and contain holes due to occlusions or insufficient overlap across the input frames. To equip EDUS with the capability to handle sparse and incomplete point clouds by filling in the missing parts, we randomly mask portions of the point cloud during the training stage. Specifically, we randomly select an 8m $\times$ 8m $\times$ 12m cuboid in each iteration, corresponding to a volume of size $[40 \times 40 \times 60]$, and eliminate the point cloud within this area. Note that this is not applied in feed-forward inference nor per-scene fine-tuning. This simple yet effective strategy enhances the generalizable ability on novel scenes, as shown in our ablation study in Sec. E.2.

B.2 Hierarchical Sampling

The total number of samples located at the ray is 160 with 128 sampled from the foreground and 32 from the background. Specifically, we first select 80 points along the ray within the foreground volume, distributing one-half of these samples uniformly, while the other half is allocated linearly based on disparity spacing, following Nerfacto [7]. Next, we iteratively perform importance sampling to sample valid regions containing solid contents based on the coarse density prediction following NeuS [9], with three iterations and 16 samples for each iteration. For the background, we additionally uniformly sample 32 points out of the foreground volume.

B.3 Accumulated Point Cloud

We design our accumulated point cloud to encourage the model to learn completion. Considering that the sparsity level in real-world driving data is unknown in advance, we aim to train a *single* generalizable model and evaluate its performance at different sparsity levels. In our main experiments, all generalizable methods are trained at the sparsity level of the drop50 setting, i.e., 50% images are available for supervision. Instead of directly using the 50% reference images

to create the accumulated point cloud, we create a more sparse point cloud for training to encourage the model to learn scene completion. Specifically, the accumulated point cloud we use for training is from every fifth of the stereo pairs. Note that during the feed-forward inference or per-scene optimization, we always use the given reference images of different sparsity levels (50%, 80%, or 90%) to generate the corresponding accumulated point cloud. This ensures fair comparison to the baselines and further verifies our method's robustness to the density of the input point cloud.

C Baselines

Generalizable NeRFs We utilize the official implementations of IBR-Net, MVSNeRF, Neo360, and MuRF. For each method, we pretrain the model using 80 scenes from the KITTI360 dataset [4] under a 50% drop rate. In the case of IBR-Net, MVSNeRF, and MuRF, we select the nearest three training frames of the target view as reference frames. As MVSNeRF constructs local cost volumes defined at reference views, we adjust the fine-tuning strategy of MVSNeRF to dynamically use the nearest reference frames to construct the cost volume, similar to the pre-training phase, rather than employing a fixed set of frames as in their original implementation.

Test-Time Optimization Methods The test-time optimization approaches are individually trained for each scene under three sparsity levels. We employ the official codebase of MixNeRF [6] and implement SparseNeRF [8] and DS-NeRF [2] by integrating stereo depth supervision onto Nerfstudio's Mip-NeRF model [1, 7]. Additionally, we utilize the splatfacto model of Nerfstudio to train our 3D GS [3]. To ensure a fair comparison, we provide the depth maps used in our approach to train the depth-supervision methods [2, 8]. When it comes to point cloud-based methods [3], we find that the missing of sky may lead to a decline of performance, so we keep the sky in the point cloud and initialize the model with it.

D Datasets

D.1 KITTI-360

We follow the 50% drop rate (Drop50) and 90% drop rate (Drop90) settings of the KITTI-360 dataset and expand them into 80% drop rate (Drop80). To be more specific, we take a pair of left and right eye image pairs as the unit. In the Drop50 setting, we use odd-numbered image pairs as the training set. Similarly, in the Drop80 and Drop90 settings, we use every fifth or every tenth pair of images for training, respectively. Note that all test frames are the same to allow comparison across different sparsity levels. For example, considering ten consecutive image pairs, Drop50 refers to using pairs of id $0, 2, 4, 6, 8$ for

training, whereas Drop80 and Drop90 refer to using $0, 5$ and 0, respectively. The test frames are selected as the left-eye image of $1, 3, 7, 9$ pairs to avoid overlapping with the training frames.

D.2 Waymo

We use the same drop rate setting and point cloud accumulation scheme as the KITTI-360 dataset for Waymo. Note that we only use the front camera of the Waymo dataset. Therefore, every odd-numbered image is used for training for the Drop50 setting, and every fifth or tenth image is used for Drop80 and Drop90, respectively. The test views are configured to encompass frames at the 3rd and 7th positions, as well as frames with id equivalent to 3 or 7 plus the multiples of 10, which has no overlapping with the training set.

E Additional Experimental Results

E.1 Qualitative Results of Ablation Study

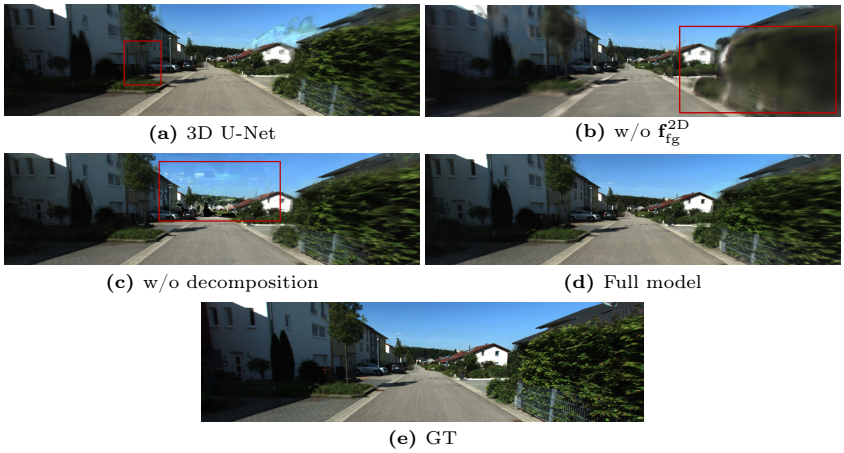
Fig. 2 demonstrates the qualitative outcomes of our ablation study. These results are all derived from feed-forward inference images under the Drop50 sparsity level, which align seamlessly with the corresponding quantitative results in Table 3 of the main paper. The substitution of the modulated SPADE CNN with a conventional 3D U-Net leads to less detailed prediction and yields blurriness in some regions. The elimination of the 2D image-based feature results in the high-frequency contents losing their sharpness and becoming noticeably blurry. Moreover, if we solely represent the scene as foreground, it results in artifacts in the distant region, due to the wrong modeling of the scene geometry. Our fully integrated model exhibits the highest quality of reconstruction, effectively validating our design choices.

E.2 Additional Ablation of Input Volume Random Masking

As shown in Tab. 3, the random mask strategy helps improve our feed-forward performance, which means our model is equipped with filling capacity to process the incomplete point cloud by adding random masking.

E.3 Additional Comparison on Waymo Dataset

We conduct zero-shot NVS under a drop rate of 50% using the model trained on the KITTI-360 dataset, and the rendered images are shown in Fig. 3. To further verify our method's performance given different training scenes, we train all generalizable methods on the Waymo dataset and evaluate them on Waymo and KITTI-360, see Tab. 4. Specifically, all models are trained with the Drop50 setting on the Waymo dataset. Our model has better performance under 80% drop rate thanks to the depth-guided global volume and achieves the best metrics



(a) 3D U-Net **(b)** w/o $\mathbf{f}_{\text{fg}}^{\text{2D}}$ **(c)** w/o decomposition **(d)** Full model **(e)** GT

Fig. 2: Qualitative Results of Ablation Study. The experiments are conducted on KITTI-360 under Drop50 sparsity level.

Ablation	KITTI360$_{drop50}$			KITTI360$_{drop80}$		
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
w/o random masking	20.62	0.694	0.228	18.54	0.641	0.288
Full model	21.93	0.745	0.178	19.63	0.668	0.244

Table 3: Ablation Study of Input Volume Random Masking. The results show that the random masking strategy in the training stage improves the feed-forward inference performance using different sparse settings.

after fine-tuning. Note that our depth guidance of the Waymo dataset is obtained from a monocular depth estimation method [11], demonstrating that we are able to learn an effective generalizable model guided by noisy monocular depth priors. Fig. 4 shows the feed-forward and fine-tuned qualitative results of our method trained and evaluated on the Waymo dataset.

E.4 Additional Qualitative Analysis of Geometry Prediction

Fig. 5 shows the novel view synthesis and depth comparison of our method to other optimization-based methods under Drop90 setting. The quantitative comparisons are reported in Table 2 of the main paper. Note that all methods except for Mix-NeRF use the same depth maps as supervision (DS-NeRF, SparseNeRF) or as input (3DGS, Ours). Although 3DGS [3] renders relatively good depth maps, it shows obvious flaws in appearance under sparse settings. Our model shows the best geometry thanks to our generalizable prior. As shown in Fig. 6, we illustrate the geometry extracted from the feed-forward inference,

Fig. 3: Zero-shot Inference on Waymo dataset. Our model is trained on the KITTI-360 dataset using Drop50 sparsity level.

Fig. 4: Qualitative Results trained and evaluated on Waymo dataset. Our model is trained on Waymo dataset using Drop50 sparsity level. The feed-forward inference and fine-tuning are performed under the Drop80 setting.

demonstrating that our EDUS is able to refine the noisy input point cloud and fill holes with generalizable 3D CNN.

E.5 Number of Reference Views for Generalizable NeRFs

As we have mentioned in the main paper, our method takes input as a point cloud, which accumulates multi-frame information in addition to the reference frames. To verify fairness and our advancement compared with other reference-

Methods	Setting	Waymo_{drop50}			Waymo_{drop80}			KITTI360_{drop50}		
		PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
IBR-Net	No per-scene opt.	22.90	0.739	0.148	17.90	0.619	0.242	18.88	0.606	0.272
MVSNeRF		18.05	0.595	0.378	17.41	0.581	0.404	15.35	0.524	0.389
Neo360		16.28	0.505	0.569	15.83	0.505	0.584	10.99	0.204	0.687
MuRF		**23.66**	0.746	0.256	21.25	0.664	0.327	19.83	**0.669**	0.340
Ours		23.41	**0.769**	**0.147**	**21.64**	**0.703**	**0.204**	**20.13**	0.659	**0.257**
IBR-Net	Per-scene opt.	23.71	**0.861**	0.121	18.58	0.741	0.222	20.78	0.650	0.205
MVSNeRF		22.42	0.699	0.309	20.81	0.657	0.348	18.80	0.611	0.354
Neo360		21.76	0.647	0.517	20.66	0.631	0.533	17.89	0.480	0.571
MuRF		24.06	0.753	0.224	22.85	0.711	0.260	21.24	0.710	0.265
Ours		**26.88**	0.839	**0.109**	**24.33**	**0.776**	**0.164**	**24.35**	**0.787**	**0.145**

Table 4: Quantitative Comparison on five test scenes among generalizable methods. All models are trained on the Waymo dataset using Drop50 sparsity level.

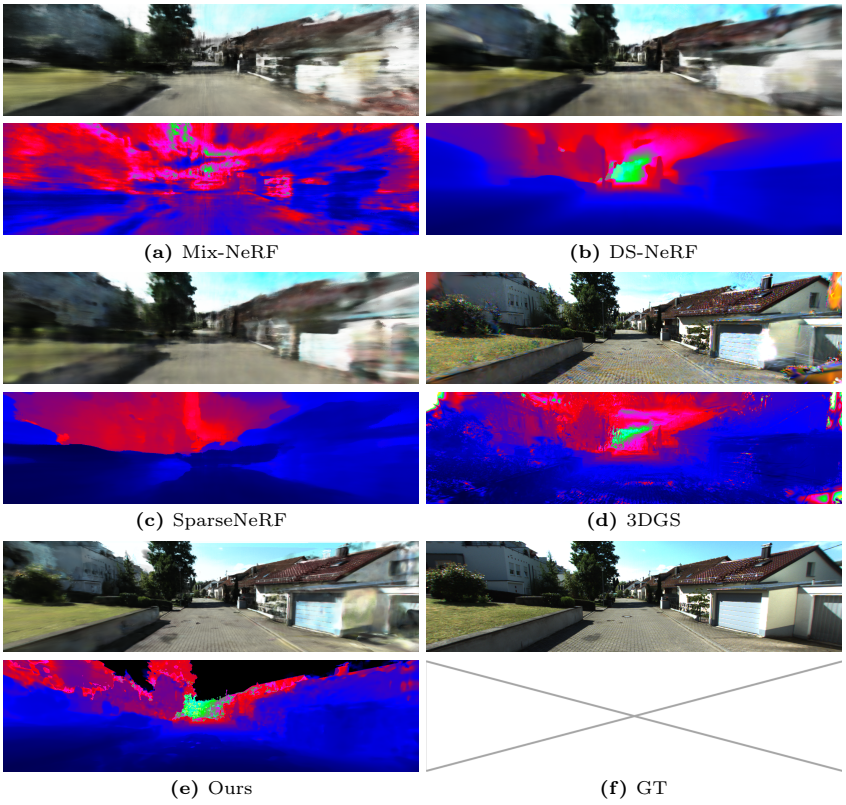
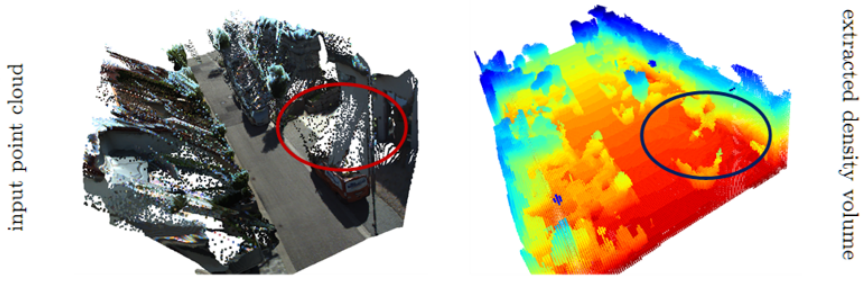
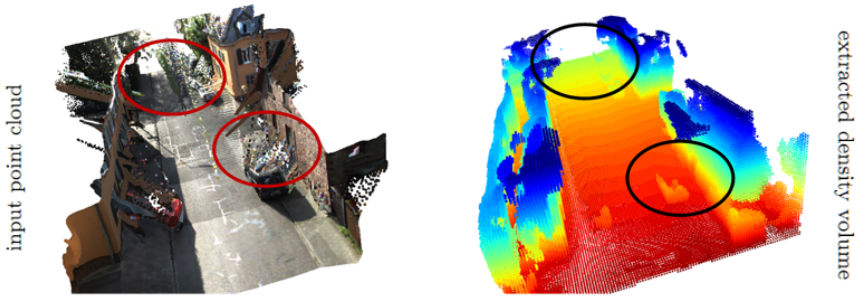


Fig. 5: Qualitative Depth Comparison with test-time optimization methods.



(a) Filling holes in input point clouds.



(b) Removing noise from input point clouds.

Fig. 6: Qualitative Results of density volume extracted from our feed-forward inference.

based methods, we evaluate the state-of-the-art approach MuRF [10] using different numbers of reference images. Specifically, we gradually increase the number of reference frames which is used for feature extraction and conduct feed-forward inference on our test sequence of the KITTI-360 dataset under Drop80 setting. We find that increasing the number of reference frames does not improve the model's performance. This may be due to the fact that further reference frames have stronger occlusions, hence increasing the difficulty of feature matching. On the other hand, the usage of more input frames also increases the burden on computing resources and causes out-of-memory when the number of views achieves 10. On the contrary, our method utilizes a fixed size of point cloud and manages to combine multi-frame guidance to predict accurate scene geometry. The results of MuRF using different numbers of reference views are shown in Tab. 5.

E.6 Limitations

Although our approach achieves superior performance on NVS from sparse urban images, it still suffers degeneration when there is less overlap between the target

Num. of Views	PSNR↑	SSIM↑	LPIPS↓
2	18.06	0.606	0.374
3	18.69	0.639	0.353
4	18.47	0.633	0.361
6	16.61	0.589	0.410
8	15.01	0.544	0.459
10	/	/	/

Table 5: Number of Reference Views for MuRF. We evaluate MuRF using various numbers of reference views. Note that the performance of MuRF does not increase wrt. the number of reference views.

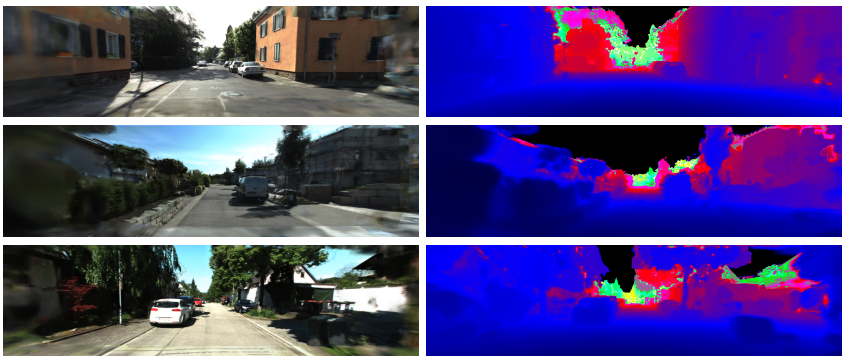


Fig. 7: Limitations in Drop90 Feed-Foward Inference. Our performance degenerates in regions with insufficient overlap between reference and target frames, e.g., the bottom of the images.

image and reference images. In this case, 3D sample points will not query accurate appearance information from nearby images, leading to blurry rendering at the bottom of the test images, as illustrated in Fig. 7. This problem may be mitigated in future work by improving the 3D CNN's capacity to directly predict high-frequency appearance instead of querying color from reference images.

References

1. Barron, J.T., Mildenhall, B., Tancik, M., Hedman, P., Martin-Brualla, R., Srinivasan, P.P.: Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 5855–5864 (2021)
2. Deng, K., Liu, A., Zhu, J.Y., Ramanan, D.: Depth-supervised nerf: Fewer views and faster training for free. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 12882–12891 (2022)

3. Kerbl, B., Kopanas, G., Leimkühler, T., Drettakis, G.: 3d gaussian splatting for real-time radiance field rendering. ACM Transactions on Graphics **42**(4) (2023)
4. Liao, Y., Xie, J., Geiger, A.: Kitti-360: A novel dataset and benchmarks for urban scene understanding in 2d and 3d. IEEE Transactions on Pattern Analysis and Machine Intelligence **45**(3), 3292–3310 (2022)
5. Mildenhall, B., Srinivasan, P.P., Tancik, M., Barron, J.T., Ramamoorthi, R., Ng, R.: Nerf: Representing scenes as neural radiance fields for view synthesis. Communications of the ACM **65**(1), 99–106 (2021)
6. Seo, S., Han, D., Chang, Y., Kwak, N.: Mixnerf: Modeling a ray with mixture density for novel view synthesis from sparse inputs. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 20659–20668 (2023)
7. Tancik, M., Weber, E., Ng, E., Li, R., Yi, B., Wang, T., Kristoffersen, A., Austin, J., Salahi, K., Ahuja, A., et al.: Nerfstudio: A modular framework for neural radiance field development. In: ACM SIGGRAPH 2023 Conference Proceedings. pp. 1–12 (2023)
8. Wang, G., Chen, Z., Loy, C.C., Liu, Z.: Sparsenerf: Distilling depth ranking for few-shot novel view synthesis. arXiv preprint arXiv:2303.16196 (2023)
9. Wang, P., Liu, L., Liu, Y., Theobalt, C., Komura, T., Wang, W.: Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. arXiv preprint arXiv:2106.10689 (2021)
10. Xu, H., Chen, A., Chen, Y., Sakaridis, C., Zhang, Y., Pollefeys, M., Geiger, A., Yu, F.: Murf: Multi-baseline radiance fields. arXiv preprint arXiv:2312.04565 (2023)
11. Yin, W., Zhang, C., Chen, H., Cai, Z., Yu, G., Wang, K., Chen, X., Shen, C.: Metric3d: Towards zero-shot metric 3d prediction from a single image. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 9043–9053 (2023)