Supplementary Material for MV-Map: Offboard HD Map Generation with Multi-view Consistency

Ziyang Xie*[1,2] Ziqi Pang*[1] Yu-Xiong Wang[1]
[1]University of Illinois Urbana-Champaign [2]Fudan University
{ziyang, ziqip2, yxw}@illinois.edu

Our supplementary material contains the following contents:

(A) **Demo video.** We provide a demo for offboard HD map generation in Sec. A.
(B) **Voxel-NeRF details.** We explain the formulation of training and using Voxel-NeRFs in Sec. B.
(C) **Generalizability of MV-Map with LiDAR.** In addition to the vision-oriented experimentation in the main paper, we show the generalizability of MV-Map and incorporate it with the LiDAR modality in Sec. C.
(D) **Applications of Auto-labeling.** We validate the effectiveness of MV-Map for auto-labeling, by using it to generate pseudo HD map labels in Sec. D.
(E) **Additional quantitative results.** We supplement ablation studies, especially using additional onboard models, in Sec. E.
(F) **Additional qualitative results.** The generated HD maps together with the reconstructed 3D structure via our Voxel-NeRF are visualized in Sec. F.
(G) **Implementation details.** We describe additional implementation details for reproducing our results in Sec. G.

A. Demo Video

We provide a demo video at https://youtu.be/SN14oTyMFrk that showcases how our MV-Map produces high-quality HD maps by fusing frames from diverse viewpoints. Notably, the video highlights the effectiveness of MV-Map in *iteratively refining* complex road topologies and long road elements while dealing with frequent occlusions in urban traffic.

B. Voxel-NeRF Details

In this section, we introduce the details of optimizing our Voxel-NeRF and augmenting our MV-Map with the encoded 3D structure (Sec. 4.3 of the main paper).

*Equal contribution.

B.1. NeRF optimization

We supervise our Voxel-NeRF in a way that is identical to standard NeRF models [12, 13, 14, 17], by using a photometric loss between the rendered pixel color and the ground-truth color.

We first describe how NeRF infers the color of every pixel in this process. NeRF renders the color of an arbitrary pixel by accumulating the density and color information along the camera ray. Specifically, we denote the camera ray for the pixel as $\mathbf{r}$, which is unique for each pixel. By denoting the camera origin as $\mathbf{o}$ and the direction of $\mathbf{r}$ as $\mathbf{d}$, every 3D coordinate along the ray can be written as $\{\mathbf{o} + t\mathbf{d} | t \in \mathcal{R}^+\}$. The RGB color of the pixel comes from the integral along the ray $\mathbf{r}$:

$$\hat{\mathbf{C}}(\mathbf{r}) = \int_{t_n}^{t_f} T(t)\sigma(t)\mathbf{c}(t)\mathrm{d}t, \tag{A}$$

where t ranges from the near and far planes t_n and t_f, $T(t) = \exp(-\int_{t_n}^{t} \sigma(\mathbf{o} + s\mathbf{d})\mathrm{d}s)$ models the accumulated transmittance along the ray from t_n to t, and σ and $\mathbf{c}$ denote the density and color encoded in NeRF, respectively.

The photometric loss is a reconstruction loss between the RGB colors predicted by NeRF and from the ground-truth images:

$$\mathcal{L}_{\text{color}} = \mathbb{E}_{\mathbf{r}} \|\hat{\mathbf{C}}(\mathbf{r}) - \mathbf{C}(\mathbf{r})\|_2^2, \tag{B}$$

where $\mathbf{C}(\mathbf{r})$ is the ground-truth RGB values extracted from the images.

As discussed in Sec. 4.3 of the main paper, we further add a total-variance loss $\mathcal{L}_{\text{TV}}$ to guide the optimization of near-ground geometry. The final loss term is:

$$\mathcal{L} = \lambda_1 \mathcal{L}_{\text{color}} + \lambda_2 \mathcal{L}_{\text{TV}}, \tag{C}$$

where λ_1 and λ_2 are trade-off hyper-parameters.

B.2. NeRF ray casting

In Sec. 4.3 of the main paper, we show how we incorporate the multi-view geometry in Voxle-NeRF with our uncertainty network. The *key operator* is to reconstruct the

position of the nearest surface for each voxel by ray-casting through the corresponding image pixel. We achieve this by rendering the *termination depth* through volume rendering.

Specifically, for every camera ray represented in the form of $\{\mathbf{o}+t\mathbf{d}|t\in\mathcal{R}^+\}$ (explained in Sec. B.1), the termination depth of the ray $\hat{D}(\mathbf{r})$ is:

$$\hat{D}(\mathbf{r}) = \int_{t_n}^{t_f} T(t)\sigma(t)\mathrm{d}t. \tag{D}$$

Similar to Eqn. A, t ranges from the near and far planes t_n and t_f, $T(t)=\exp(-\int_{t_n}^{t}\sigma(\mathbf{o}+s\mathbf{d})\mathrm{d}s)$ models the accumulated transmittance along the ray from t_n to t, and σ denotes the density in NeRF.

C. Generalizability of MV-Map with LiDAR

In this section, we provide the model design details of incorporating MV-Map with LiDAR modality as described in Sec. 5.6 (main paper).

Onboard model. We modify the original image-based onboard model by adding a branch of LiDAR encoder with PointPillar [6] to generate the BEV feature maps from the point clouds. The BEV feature maps generated by the LiDAR encoder are later stacked with the image-based BEV features to form the final BEV features.

Uncertainty network. The architecture of the uncertainty network remains unchanged when integrating the LiDAR sensor, as our offboard fusion pipeline is *agnostic* to the upstream BEV perception modules.

Voxel-NeRF. In addition to optimizing the Voxel-NeRF with the photometric loss and total-variance loss as in Sec. 4.3 (main paper), we further leverage the point clouds to improve NeRF. Specifically, we follow DS-NeRF [3] and apply an extra depth loss term:

$$\mathcal{L}_{\text{depth}} = \mathbb{E}_{\mathbf{r}}\|\hat{D}(\mathbf{r}) - D(\mathbf{r})\|_2^2, \tag{E}$$

where $\hat{D}(\mathbf{r})$ is the rendered termination depth in Eqn. D, and $D(\mathbf{r})$ is the depth of LiDAR points. Note that point clouds are sparser than image pixels, so we project LiDAR points onto the images and only apply the above loss term to the pixels that correspond to LiDAR points for supervision.

Our final training loss for Voxel-NeRF combines the photometric, total-variance, and depth losses when the LiDAR modality is available:

$$\mathcal{L} = \lambda_1\mathcal{L}_{\text{color}} + \lambda_2\mathcal{L}_{\text{TV}} + \lambda_3\mathcal{L}_{\text{depth}}, \tag{F}$$

where λ_1, λ_2, and λ_3 are trade-off hyper-parameters.

Results. Table 5 (main paper) summarizes the result of MV-Map with LiDAR, which again significantly outperforms the onboard model. In addition, due to leveraging the additional modality, MV-Map with both camera and LiDAR achieves larger improvement, compared with the unimodal model result with only camera shown in Table 1 This result serves as further evidence that our framework is capable of adapting to multi-modality and achieving improved performance.

Table A: Comparison between onboard models trained with either ground-truth labels (GT) or pseudo-labels generated by our MV-Map (PL). The model trained with our pseudo-labels achieves comparable performance. This validates the high quality of HD maps generated by MV-Map and further supports its effectiveness for auto-labeling.

Label	mIoU (Validation set, Long-range)			
	Divider	Ped Crossing	Boundary	All
PL (Ours)	**38.99**	25.15	**38.68**	**34.27**
GT	38.89	**25.40**	38.16	34.15

D. Applications of Auto-labeling

The experimental results in the main paper show that MV-Map generates high-quality HD map labels. This indicates that our method is an effective *auto-labeling* strategy, which can potentially serve as a substitute for human labeling and thus support downstream applications. To further assess the quality of these labels, which we refer to as "*pseudo-labels*," we conduct an experiment by training a new onboard model with pseudo-labels and comparing its efficacy with that trained with ground-truth labels.

To this end, we follow the *semi-supervised learning* experimental setup introduced in offboard 3D detection [15]. We use 50 out of 700 sequences on the training set of nuScenes [1] to train our uncertainty network and deploy it to infer the HD map labels for the remaining 650 sequences on the training set. We then train an onboard model *from scratch* on these 650 sequences with either the ground-truth labels or pseudo-labels from MV-Map.

The result in Table A shows that the model trained with pseudo-labels achieves comparable performance to that trained with ground-truth labels. This suggests that our auto-labeling approach is effective for supporting semi-supervised training. It is worth noting that our pseudo-labeling performs slightly better, likely because it helps to reduce over-fitting as evidenced by that on the *training* set using ground-truth labels results in 2.5% higher mIoU over pseudo-labels. Based on the high quality of the generated HD maps, our auto-labeling pipeline has the potential to be useful for other BEV perception tasks that involve traffic elements, such as BEV segmentation and lane detection. We leave such investigation as interesting future work.

Table B: Performance of MV-Map with varied training-time frame numbers. Given that the performance saturates at 5 frames, we adopt 5 frames for training to best trade-off accuracy and computational cost. Note that during inference, we apply MV-Map to sequences with unbounded lengths.

#Frames	mIoU (Long-range)			
	Divider	Ped Crossing	Boundary	All
3	47.64	32.36	49.67	43.22
5	48.15	33.34	**50.28**	43.92
7	**48.23**	**34.31**	50.11	**44.22**

Table C: MV-Map generalizes to other onboard models. Using HDMapNet [7] as our onboard model, MV-Map is consistently effective and significantly improves the HD map quality.

Methods	mIoU			
	Divider	Ped Crossing	Boundary	All
HDMapNet	46.20	24.38	56.99	42.52
+MV-Map	**49.82**	**29.83**	**58.54**	**46.06**
△ mIoU	+3.62	+5.45	+1.55	+3.54

E. Additional Quantitative Results

E.1. Impact of Training-time Frame Number

As described in Sec. 4.4, we train the uncertainty network on clips with a fixed number of frames due to GPU capacities but later apply it to *sequences with unbounded lengths*. We analyze how the training-time frame number impacts the performance of the uncertainty network. In Table B, we demonstrate that *increasing the number of frames is beneficial to the fusion performance*, as the uncertainty network has access to more diverse viewpoints for fusion during the training time. Moreover, increasing from 3 to 5 frames has a significant gain in performance, while increasing from 5 to 7 frames only has marginal improvement. Our experiments in the main paper use 5-frames for training to balance the performance and computation cost.

E.2. Generalizing to Additional Onboard Models

We demonstrate that the region-centric fusion approach in our MV-Map is *generalizable to other onboard BEV perception models*. In addition to the SimpleBEV [4] and VectorMapNet discussed in Sec. 4 of the main paper, we adopt HDMapNet [7]. Following our experiment in Sec. C, we incorporate the HDMapNet encoder with the LiDAR point clouds. As shown in Table C, MV-Map significantly improves upon both the HDMapNet [7] and VectorMapNet [9] result, thus supporting the generalizability of MV-Map.

E.3. Sensitivity to KL divergence loss

To further investigate the sensitivity of MV-Map's performance to the weight of the KL divergence loss (Sec. 4.2, main paper), we conduct ablation experiments with different settings of ω. Compared to the original setting of ω = 0.1, setting ω = 0.2 results in an increase of 0.25 in mIoU, while setting ω = 0.05 causes a small decrease of 0.04 in mIoU. These minor variations in mIoU with sizable changes in ω indicate that MV-Map's performance is fairly robust to the exact weight of the KL divergence loss.

Table D: Comparison between fusing BEV feature maps F_i and semantic maps. We choose to fuse semantic maps in Sec. 4.2 (main paper) because of its better performance.

Fusion	mIoU			
	Divider	Ped Crossing	Boundary	All
Onboard	39.30	26.44	39.10	34.95
BEV feature	45.88	33.06	46.38	41.77
Semantic	**48.15**	**33.34**	**50.28**	**43.92**

E.4. Fusing semantics versus BEV features.

Our region-centric framework performs weighted averages over the semantic maps S_i instead of the BEV features F_i. In Table D, we justify our design choices, where fusing BEV features is worse than fusing semantic maps. The main reason is the domain shift between training and inference when we have numerous input frames of offboard data. Furthermore, fusing BEV features is also less practical which requires significantly more disk space to store high-dimensional features.

F. Additional Qualitative Results

F.1. HD-Map Visualization

We provide more qualitative results on HD map generation in Fig. A, in addition to our visualizations in Fig. 5 and Fig. 8 of the main paper. Compared with onboard approaches, our offboard MV-Map significantly improves the quality of HD maps for complex structures.

F.2. Voxel-NeRF Visualization

We provide more visualization results of our Voxel-NeRF to indicate its capability of encoding multi-view consistency. In Fig. B, we show the reconstructed 3D structure of the scenes by our Voxel-NeRF model, by converting the diffuse color and opacity of every voxel to a colored point cloud. Qualitative results demonstrate that our Voxel-NeRF successfully optimizes a high-resolution scene representation with multi-view consistency.

G. Implementation Details

We provide the detailed hyper-parameters and procedures to reproduce MV-Map as described in Sec. 4.4 and Sec. 5.1 (main paper), as well as Sec. C.

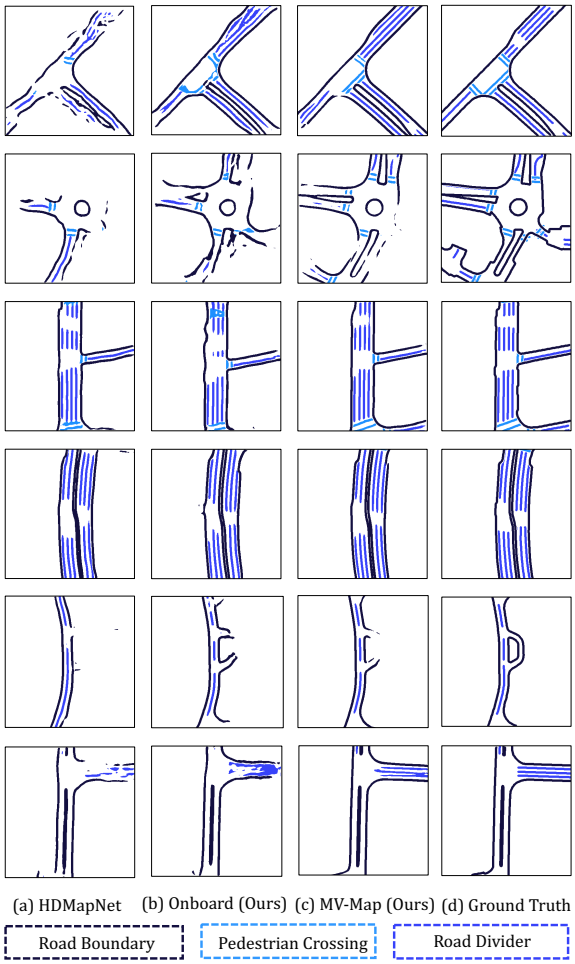

Figure A: Qualitative results for HD map generation. We compare the generated results from HDMapNet [7], our onboard model, and the offboard fused results from MV-Map. Compared with other approaches, our MV-Map achieves better fidelity for complex road topology.

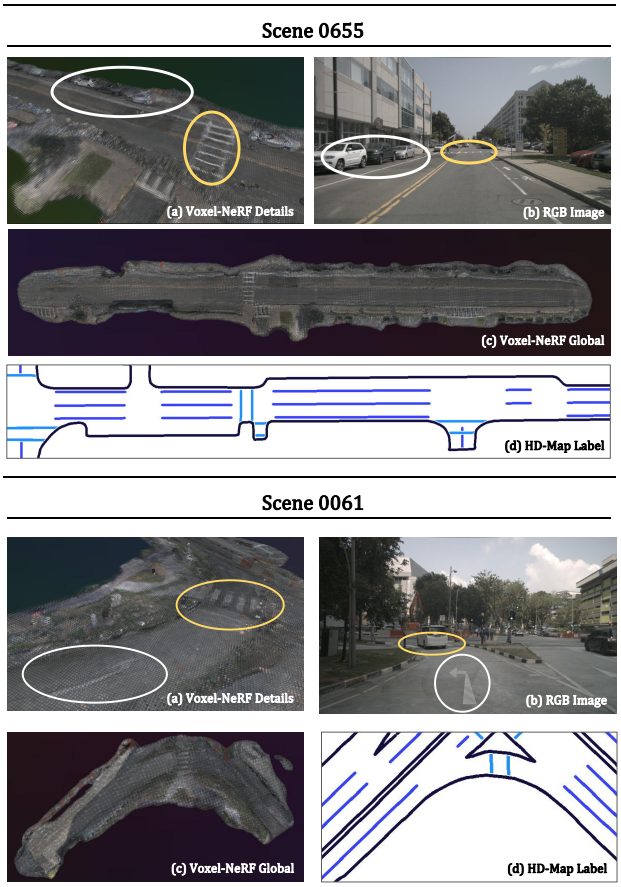

Figure B: Visualization of reconstruction results by Voxel-NeRF on the scene 0655 and 0061 from nuScenes. **(a)** NeRF's results in the highlighted regions; **(b)** images captured by the ego vehicle; **(c)** NeRF's results for the whole scene; **(d)** ground-truth HD map labels. As highlighted here, our Voxel-NeRF optimizes the 3D structure of the whole scene with high quality and multi-view consistency.

G.1. Onboard Model

As SimpleBEV [4] was not originally proposed for HD map construction (despite its strong performance), we inherit their encoder design and train our own onboard models.

Backbone. Same as [4], we adopt ResNet-50 [5] as the backbone to extract the feature maps for 6 surrounding images per frame on nuScenes. The third and final stages of the ResNet output are used. We apply an additional convolution layer to generate the final feature map with a length and width of 1/8 compared to the original size of the images.

Feature lifting. For feature lifting, we use a sampling-based BEV encoder to lift the 2D image feature into BEV space. We first construct a local 3D voxel grid shaped $400 \times 400 \times 6$ around the ego vehicle. The voxel grid size is 0.15m for the short-range (60m $\times$ 30m) setting and 0.25m for the long-range (100m $\times$ 100m) setting. Then, for each grid point, we acquire its positions on the image plane with intrinsic and extrinsic matrices, bi-linear sample the image features, and fill them back into each voxel grid. Finally, we reduce the voxel features into a BEV feature map with an additional voxel encoder to make it a 2D BEV feature. During this process, the height range of the sampled voxel grid in our BEV encoder is -4m to 2m relative to the sensor origin. The final output BEV feature map shapes 128 $\times$ 400 $\times$ 400, where 128 is the feature dimension, and 400 is the length and width of the BEV feature map.

Decoder. To maintain generality and comparability, we used the same decoder as HDMapNet [7]. The main structure contains three blocks from ResNet18 [5] to generate the final prediction.

Loss function. We use Focal loss [8], which is a dynamically scaled cross-entropy loss as our segmentation loss to solve the imbalance distribution between the most common road label and the crossing label that is relatively scarce and hard to learn:

$$\mathbf{FL}(p_t) = -\alpha_t(1-p_t)^{\gamma}\log(p_t), \tag{G}$$

where we set the factor $\alpha_t = 1$ and $\gamma = 2$. During training, the scaling factor can reduce the impact of dominant categories (*e.g.*, road segments) and increase the loss assigned to challenging ones (*e.g.*, pedestrian crossing).

Training. During training, we initialize the backbone ResNet-50 from the ImageNet1k [2] pretrained checkpoint. It is then trained for 16 epochs with an AdamW [10] optimizer, with an initial learning rate of 1e-3 under a 1-cycle schedule and focal loss [8] as loss function. We train our model with 4×A100 GPUs with 2 samples per GPU. The total training process takes around 10 hours.

G.2. Voxel-NeRF

Architecture. Our Voxel-NeRF models are trained with a fixed voxel size of 0.5m. The height range of our voxel is set to -4m to 2m relative to the height value of sensor origins. The near plane and the far plane of our NeRF model are 0.1m and 64m, respectively. The RGB network has a width of 128 and a depth of 3 layers. Within our model, each voxel encodes the feature with dimension 12. The first 3 channels represent the diffuse color, and the rest 9 channels concatenate the viewing directions to decode the final RGB color c with an RGB MLP.

Training. We reconstruct all 850 scenes in nuScenes and train 30,000 iterations with AdamW [11] optimizer and 1e-3 learning rate for each scene. Please note that our training is *without* the coarse-to-fine strategy proposed in [16]. As our scene scale is predefined and does not need the coarse stage to find a tight bounding box for further optimization. When we use the total-variance loss (Sec. 4.3, main paper), the balance loss weights λ_1 is 1 and λ_2 is 1e-5. The training process takes around 15 minutes for each scene on a single A40 GPU.

G.3. Uncertainty Network

Architecture. We show our uncertainty network design in Fig. 4 (main paper). It has a U-Net-like architecture which takes in the 128-channel BEV feature map with channels, and a 6-channel augmented input from NeRF (as Sec. 4.3, main paper), and outputs a final 128-channel feature map. Then the per-pixel confidence weight and the KL divergence prediction are output by two independent 1×1 convolution layers with kernel size 1.

Training. Due to our storage constraints, we train the uncertainty network on a small subset (50 scenes) of the full training set for 5 epochs, but we manage to evaluate the full validation set of nuScenes with 150 sequences, which enables a fair comparison with other methods. As explained in Sec. 4.4 and Sec. 5.1 (main paper), we train the uncertainty network on short video clips with 5 frames and deploy it to all the frames in a nuScenes sequence during the inference time. A key detail to handle the varying sequence lengths for inference time is to set the *batch size* to 5 during the training time. The final loss is a weighted sum of the segmentation loss and the auxiliary KL divergence loss (Sec. 4.2, main paper), the loss weights are 1 and 0.1, respectively. The network is trained with an AdamW [10] optimizer with a learning rate of 1e-3. The training process takes around 30 minutes on a single A100 GPU.

G.4. MV-Map with LiDAR

For the implementation of the LiDAR MV-Map, we maintain the hyperparameters of the onboard model and the uncertainty network at the same values as the unimodal model. The onboard model's training process requires around 12 hours to complete using 4×A100 GPUs, while the uncertainty network training process takes an additional hour using a single A100 GPU.

As discussed in Sec. C, when training with LiDAR signals, our Voxel-NeRF applies an extra depth loss term with $\lambda_3 = 0.1$. We keep other hyperparameters the same and train our Voxel-NeRF for 30,000 iterations for each scene, which takes 15 minutes on a single A100 GPU.

G.5. MV-Map with Monocular Depth

We experiment using monocular depth for uncertainty network in Sec. 5.3 (main paper). Specifically, we replace the termination depth from NeRF (Eqn. D) with the depth generated by *NeWCRFs* [18] to estimate the 3D positions of surface points. The other implementation details are untouched.

References

[1] Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuScenes: A multimodal dataset for autonomous driving. In *CVPR*, 2020. 2

[2] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *CVPR*, 2009. 5

[3] Kangle Deng, Andrew Liu, Jun-Yan Zhu, and Deva Ramanan. Depth-supervised NeRF: Fewer views and faster training for free. In *CVPR*, 2022. 2

[4] Adam W Harley, Zhaoyuan Fang, Jie Li, Rares Ambrus, and Katerina Fragkiadaki. Simple-BEV: What really matters for multi-sensor BEV perception? In *ICRA*, 2023. 3, 4

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 4

[6] Alex H. Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. PointPillars: Fast encoders for object detection from point clouds. In *CVPR*, 2019. 2

[7] Qi Li, Yue Wang, Yilun Wang, and Hang Zhao. HDMapNet: An online hd map construction and evaluation framework. In *ICRA*, 2022. 3, 4

[8] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017. 5

[9] Yicheng Liu, Yuan Yuantian, Yue Wang, Yilun Wang, and Hang Zhao. VectorMapNet: End-to-end vectorized HD map learning. In *ICML*, 2023. 3

[10] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017. 5

[11] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR*, 2019. 5

[12] 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. 1

[13] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan Barron, Ravi Ramamoorthi, and Ren Ng. NeRF: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020. 1

[14] Thomas Neff, Pascal Stadlbauer, Mathias Parger, Andreas Kurz, Joerg H. Mueller, Chakravarty R. Alla Chaitanya, Anton S. Kaplanyan, and Markus Steinberger. DONeRF: Towards real-time rendering of compact neural radiance fields using depth oracle networks. In *Computer Graphics Forum*, 2021. 1

[15] Charles R Qi, Yin Zhou, Mahyar Najibi, Pei Sun, Khoa Vo, Boyang Deng, and Dragomir Anguelov. Offboard 3D object detection from point cloud sequences. In *CVPR*, 2021. 2

[16] Cheng Sun, Min Sun, and Hwann-Tzong Chen. Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction. In *CVPR*, 2022. 5

[17] Matthew Tancik, Vincent Casser, Xinchen Yan, Sabeek Pradhan, Ben Mildenhall, Pratul Srinivasan, Jonathan T. Barron, and Henrik Kretzschmar. Block-NeRF: Scalable large scene neural view synthesis. In *CVPR*, 2022. 1

[18] Weihao Yuan, Xiaodong Gu, Zuozhuo Dai, Siyu Zhu, and Ping Tan. NeWCRFs: Neural window fully-connected CRFs for monocular depth estimation. In *CVPR*, 2022. 5