Urban Radiance Field Representation with Deformable Neural Mesh Primitives —Supplementary Material

Fan Lu[1] * Yan Xu[2] * Guang Chen[1] † Hongsheng Li[2,3,4] Kwan-Yee Lin[2,3] † Changjun Jiang[1]
[1]Tongji University [2]The Chinese University of Hong Kong [3]Shanghai AI Laboratory [4]CPII
{lufan,guangchen,cjjiang}@tongji.edu.cn
yanxu@link.cuhk.edu.hk hsli@ee.cuhk.edu.hk junyilin@cuhk.edu.hk

In this supplementary material, we provide: (1) more detailed discussion on the implementation details; (2) the additional experimental results; (3) the potential societal impact of our method.

S-1. Implementation Details

S-1.1. Implementation and Training

The overall framework is implemented using PyTorch [11]. The differentiable rasterization is implemented based on PyTorch3D [12]. The network F_θ is composed of 8 layers with width 256 for opacity prediction and additional 2 layers for view-dependent radiance value prediction. In our lightweight version, the layer number and width of the MLPs for opacity prediction are reduced to 2 and 64, respectively. We use positional encoding with frequency $L = 4$ to encode the view-dependent factors. As mentioned in the manuscript, we use Mip-NeRF [1] to handle the non-structure regions. Specifically, our implementation of Mip-NeRF consists of 8 layers with width 256 for density prediction and additional 2 layers for color prediction. The coarse and fine networks of Mip-NeRF are both sampled 128 times and the frequencies of positional encoding for coordinates and view directions are set to 10 and 4, respectively. We leverage Metashape [7] for point-cloud reconstruction on KITTI-360 dataset [6], which is a software for large-scale 3D reconstruction and shows better reconstruction quality than COLMAP [14, 13]. For Waymo dataset [15], we follow NPLF [10] to use accumulated LiDAR point clouds for scene voxelization. We optimize the shape for 50k iterations and the neural rendering network for 100k iterations with a batch size of 16384 rays. Adam [5] is used as the optimizer with an initial learning rate of 5×10^{-4} and an exponentially decayed factor of 0.999999.

* equal contribution.
†corresponding authors.

S-1.2. Scene Editing

Our representation supports scene editing including texture editing, object removal/insertion, *etc*. These operations are achieved by locally modifying the vertex radiance features or by DNMP removal/insertion. Specifically, for *texture editing*, we only need to edit one 2D image and finetune the vertex features corresponding to the edited regions (practically implemented using masks in the losses). During finetuning, the network parts F_θ are fixed. To achieve *object removal*, we can remove the corresponding DNMPs and fill the holes with the copied DNMPs from similar structures of the other parts. Similarly, we can also insert the DNMPs of the corresponding object to the desired positions in the scene for *object-level insertion*. **Please refer to our supplemental video for more visualization results.**

S-1.3. Loss Function

As mentioned in the manuscript, we encourage normal consistency and mesh smoothness with a regularization loss $L_{regularize}$ during the training of the auto-encoder. Specifically, $L_{regularize}$ consists of the normal consistency loss L_{normal} [4] and the Laplacian mesh smoothness loss L_{smooth} [9], *i.e.*, $L_{regularize} = L_{normal} + L_{smooth}$.

The normal consistency loss is defined for each pair of neighboring faces in the mesh. Denoting the surface normal of two neighboring faces as $\mathbf{n}_i^1, \mathbf{n}_i^2$, normal consistency loss can be calculated as

$$L_{normal} = \sum_i (1 - \cos(\mathbf{n}_i^1, \mathbf{n}_i^2)). \tag{1}$$

To calculate the Laplacian mesh smoothness loss, a Laplacian matrix $\mathbf{L} \in \mathbb{R}^{N \times N}$ (N is the number of vertices) is defined as

$$\mathbf{L}_{ij} = \begin{cases} -1, & i = j \\ w_{ij}, & \{i,j\} \in \mathbf{E} \\ 0, & otherwise \end{cases} \tag{2}$$

where $\mathbf{E}$ is the collection of edges, and

$$w_{ij} = \frac{\omega_{ij}}{\sum_{\{i,k\}\in\mathbf{E}} \omega_{ik}}, \tag{3}$$

where $\omega_{ij} = 1$ since we use uniform weights. Then, the Laplacian mesh smoothness loss can be represented as

$$L_{smooth} = \sum \|\mathbf{L}\mathbf{V}\|_2, \tag{4}$$

where $\mathbf{V} \in \mathbb{R}^{N\times 3}$ is the matrix of vertex coordinates.

S-1.4. Database Construction

To train the auto-encoder mentioned in Sec. 3.1 of the manuscript, we build a database containing a large number of meshes. To this end, we first collect a set of reconstructed point clouds (including reconstructed point clouds from Multi-View Stereo (MVS), LiDAR point clouds, *etc*.). Then, we chunk these point clouds into a set of local point-cloud patches. Next, for each point-cloud patch, we deform a unit spherical mesh to fit the shape of the patch. Specifically, we take the offsets for mesh vertices as learnable parameters and minimize the chamfer distance L_{CD} between the local point-cloud patch and the sampled points [1] on the mesh used to fit the patch. The loss function L_{CD} can be formally represented as

$$L_{CD} = \sum_i \min_{p_j^S \in \mathbf{P}^S} \left\|p_i^M - p_j^S\right\|_2^2 + \sum_j \min_{p_i^M \in \mathbf{P}^M} \left\|p_i^M - p_j^S\right\|_2^2, \tag{5}$$

where $\mathbf{P}^M$ and $\mathbf{P}^S$ are sampled points on the mesh faces and the local point cloud respectively. Finally, we create a large database of local primitive meshes with well-defined geometry to train our auto-encoder.

S-1.5. Optimization of DNMPs

To better illustrate the shape optimization process of DNMPs, we visualize several examples in Fig. S-1. It is shown that with the optimization of shape latent codes $\mathbf{z}$, DNMPs can be deformed into various basic shapes (*e.g.*, such as planes, curved surfaces, and pillars), which constitute the building blocks of entire scenes.

S-1.6. Details of Dataset

KITTI-360 dataset. We cut 5 sequences from drive `2013_05_28_drive_0009_sync` in KITTI-360 dataset [6] (denoted as `seq_1` to `seq_5`), the details are shown in Table S-1.

Waymo dataset. For Waymo dataset [15], we follow NPLF [10] to select 6 sequences containing mainly static scenes. For completeness, we also provide the details of the sequences here in Table S-2.

[1]The points are resampled in every iteration to improve the robustness.

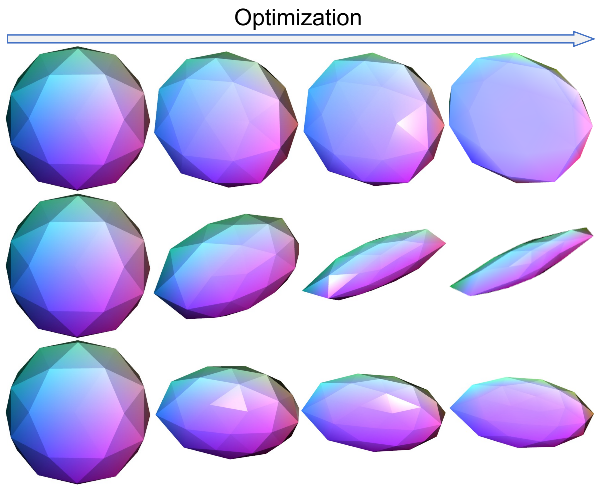

Figure S-1. Optimization of DNMPs. DNMPs can be deformed from spherical template meshes into various basic shapes (*e.g.*, plane, pillar, curved surface, *etc*.), via optimizing the shape latent codes.

Table S-1. Details of KITTI-360 dataset.

Sequence	First frame	Last frame
`seq_1`	715	794
`seq_2`	880	959
`seq_3`	1102	1181
`seq_4`	2170	2249
`seq_5`	2900	2979

S-2. Additional Experimental Results

S-2.1. Comparison with MVS

To verify the effectiveness of our method, we directly recover the meshes [7] based on the point clouds from Multi-View Stereo (MVS) instead of using our DNMP-based shape reconstruction. Then we attach radiance features to the mesh vertices and adopt the same rendering pipeline as our methods for fair comparisons. The experimental results for novel view synthesis on the KITTI-360 dataset are shown in Table S-3. Our method significantly outperforms the counterpart that uses meshes from MVS. We also provide qualitative comparison results in Fig. S-2. It is shown that the rendered images based on the meshes from MVS have much more flaws which are usually caused by the ubiquitous holes and noises in the outdoor point clouds.

S-2.2. Comparison with the Baselines with Depth Supervision

We further train the competitive baseline methods (*i.e*., Mip-NeRF [1] and Mip-NeRF 360 [2] by adding the extra depth supervision for better comparison. The evaluation results on KITTI and Waymo datasets are shown in

Table S-2. Details of Waymo dataset.

Set	Segment Name/ID	First frame	Last frame
validation 0000	segment-10247954040621004675_2180_000_2200_000_with_camera_labels	0	80
validation 0000	segment-10713922294950850361_1844_790_1864_790_with_camera_labels	135	197
validation 0000	segment-11037651371539287009_77_670_97_670_with_camera_labels	0	164
validation 0001	segment-13469905891836363794_4429_660_4449_660_with_camera_labels	0	197
validation 0002	segment-14333744981238305769_5658_260_5678_260_with_camera_labels	0	198
validation 0002	segment-14663356589561275673_935_195_955_195_with_camera_labels	0	197

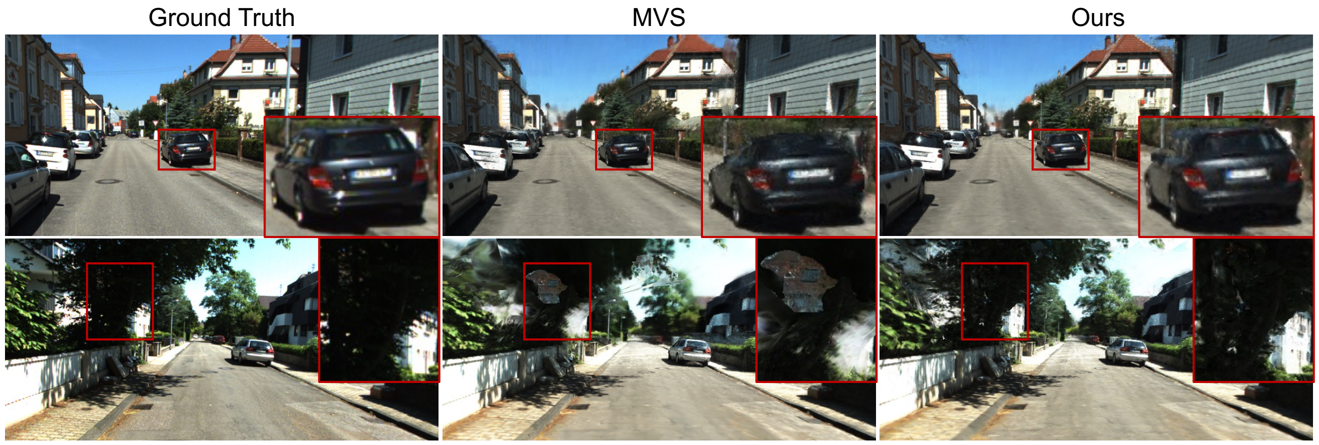

Figure S-2. Qualitative comparison of novel view synthesis on KITTI-360 dataset with neural rendering using meshes reconstructed from Multi-View Stereo (MVS). Cropped patches are scaled to highlight details. Reconstructed meshes from MVS are incomplete (*e.g.*, the upper part of the car in the cropped patches in the first row) and contain occlusions (*e.g.*, the building in the bottom-right corner of the cropped patches in the second row), leading to blurry and unreal rendering results.

Table S-3. Comparison with neural rendering using meshes reconstructed from Multi-View Stereo (MVS). The comparison is based on KITTI-360 dataset.

Methods	PSNR↑	SSIM↑	LPIPS↓
MVS [7]	21.82	0.817	0.341
Ours	**23.41**	**0.846**	**0.305**

Table S-4. Performance comparison of novel view synthesis with other competitive baselines on KITTI-360 dataset. † means the method is trained with additional depth supervision.

Method	PSNR↑	SSIM↑	LPIPS↓
Mip-NeRF† [1]	23.26	0.815	0.404
Mip-NeRF 360† [2]	**23.43**	0.837	0.354
Ours	<u>23.41</u>	**0.846**	**0.305**

Table S-5. Performance comparison of novel view synthesis with other competitive baselines on Waymo dataset. † means the method is trained with additional depth supervision.

Method	PSNR↑	SSIM↑	LPIPS↓
Mip-NeRF† [1]	27.02	0.881	0.449
Mip-NeRF 360† [2]	27.46	**0.895**	0.393
Ours	**27.62**	<u>0.892</u>	**0.381**

Table S-4, S-5, respectively. The supervision from depth speeds up their convergence and modestly improves their performances. Their PSNRs and SSIMs are comparable to ours but their performance regarding LPIPS is still inferior to ours. Besides, our method is also much more efficient in terms of runtime and memory consumption.

S-2.3. Comparison with Large-scale NeRFs

Table S-6 shows the performance of BungeeNeRF [17] and Mega-NeRF [16] on KITTI-360 dataset. Mega-NeRF extends vanilla NeRFs by decomposing the scene into multiple regions, while the performance is still inferior to ours. BungeeNeRF improves view synthesis by training NeRFs from far to near in a progressive manner, which is suitable for bird-eye-view images. We find that it performs similarly to vanilla NeRFs in our scenes. We deem that their progressive strategy does not fit well to the vehicle-captured data of urban scenes.

S-2.4. Comparison with MobileNeRF

Table S-7 shows the novel view synthesis performance of MobileNeRF [3] on KITTI-360 dataset. The performance of the continuous version of MobileNeRF (Stage 1 with high computational overhead) is inferior to ours, and its discretized version (Stage 3) performs worse. We infer that this

Table S-6. Comparison with large-scale NeRFs. The comparison is based on KITTI-360 dataset.

Methods	PSNR↑	SSIM↑	LPIPS↓
BungeeNeRF [17]	22.02	0.788	0.429
Mega-NeRF [16]	23.15	0.826	0.326
Ours	**23.41**	**0.846**	**0.305**

could be caused by MobileNeRF's non-tight mesh representation, which degrades the generalization ability if trained with limited views.

Table S-7. Comparison with MobileNeRF.

Methods		PSNR↑	SSIM↑	LPIPS↓
MobileNeRF	Stage 1	23.18	0.823	0.350
	Stage 2	22.10	0.784	0.412
	Stage 3	20.80	0.752	0.445
Ours		**23.41**	**0.846**	**0.305**

S-2.5. Analysis of Storage Budgets

In terms of storage budgets, the methods like Point-NeRF [18] need to store point-wise features of a dense point cloud for a scene. Their feature numbers can be $5\times$ more than ours, and their feature dimension is usually higher (typically $> 2\times$) than ours. Instant-NGP [8] reduces storage budgets by using multiresolution hash tables, while the overall storage budget is still about $2\times$ of ours thanks to our compact surface-aligned representation (463MB versus 243MB for one sequence in our scenarios). Moreover, we can further reduce the storage budgets by increasing the DNMPs' radii with acceptable rendering quality compromises. As shown in Table 2 in the main manuscript (columns under *DNMP radius*) and Fig. S-3, our performance is tolerant to large DNMP radii. For example, increasing the radius to 2 meters only leads to $< 5\%$ degradation in PSNR but reduces $\sim 10\times$ vertex feature numbers.

S-2.6. Limitations in Indoor Scenes

For indoor scenarios, we usually could acquire the initial 3D reconstruction from either ToF cameras or MVS algorithms. (1) ToF cameras have difficulties in measuring the depth of specular or absorbing materials, e.g., the monitors in Fig. S-4, which results in empty holes in the initial 3D reconstruction. Nevertheless, our method can still perform well (shown in the 2nd and 3rd columns of Fig. S-4) with such incomplete initialization, thanks to the completion ability of hierarchical DNMPs. (2) When using classic MVS for initial 3D reconstruction, the missing reconstruction for textureless regions like white walls could be too large for our method to recover the scene geometry reasonably, leading to unsatisfactory results. We could exploit

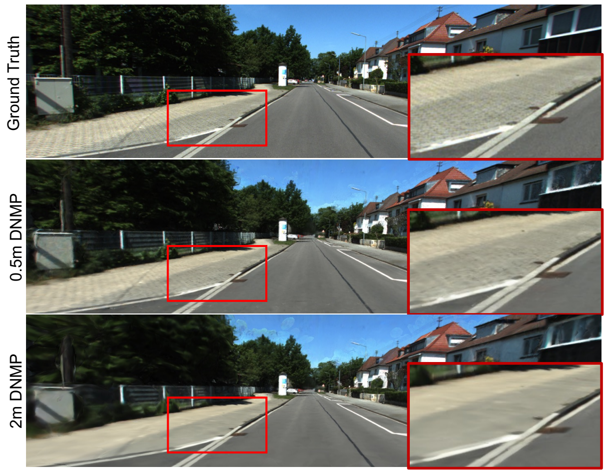

Figure S-3. View synthesis results based on DNMPs of different sizes. Even only based on the DNMP with a dimension of 2 meters, the synthesized images are still of acceptable quality, though the texture details are lost compared to results of 0.5m-DNMPs.

depth completion techniques or develop primitive growing to mitigate this issue in the future.

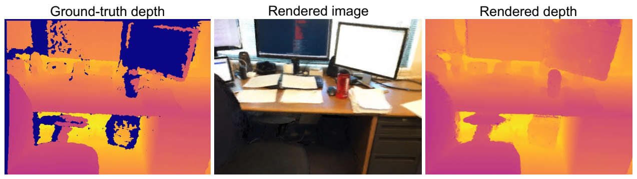

Figure S-4. A sample of rendering results on ScanNet dataset.

S-2.7. Novel Semantic Synthesis

Our representation can also be adapted to novel semantic synthesis by additionally estimating the semantic labels with the implicit function (similar to the radiance-value estimation). Fig. S-5 visualizes several cases. The successful application of our method to this task shows the potential for being deployed in semantic mapping, low-cost semantic annotation systems *etc*.

S-2.8. More Visualization Results

We provide additional visualization comparisons on KITTI-360 dataset and Waymo dataset in Fig. S-6, S-7, re-

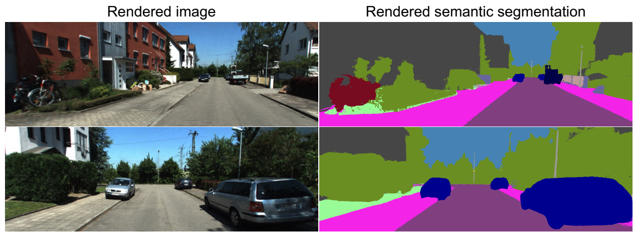

Figure S-5. Novel semantic synthesis.

spectively. Please refer to our supplemental video for more visualization results.

S-3. Potential Negative Societal Impact

This paper proposes a novel neural representation for novel view synthesis in urban scenes. We believe this technology has a positive impact on society. However, there is no guarantee that it could not be used in applications with negative social impacts. For example, this technique can be applied to synthesize fake scenes, which can further be used to generate fake events and news.

References

[1] Jonathan T Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P Srinivasan. Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5855–5864, 2021. 1, 2, 3

[2] Jonathan T Barron, Ben Mildenhall, Dor Verbin, Pratul P Srinivasan, and Peter Hedman. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5470–5479, 2022. 2, 3

[3] Zhiqin Chen, Thomas Funkhouser, Peter Hedman, and Andrea Tagliasacchi. Mobilenerf: Exploiting the polygon rasterization pipeline for efficient neural field rendering on mobile architectures. *arXiv preprint arXiv:2208.00277*, 2022. 3

[4] Mathieu Desbrun, Mark Meyer, Peter Schröder, and Alan H Barr. Implicit fairing of irregular meshes using diffusion and curvature flow. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 317–324, 1999. 1

[5] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations*, 2015. 1

[6] Yiyi Liao, Jun Xie, and Andreas Geiger. Kitti-360: A novel dataset and benchmarks for urban scene understanding in 2d and 3d. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022. 1, 2

[7] Agisoft LLC. Agisoft metashape, 2022. 1, 2, 3

[8] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Trans. Graph.*, 41(4):102:1–102:15, July 2022. 4

[9] Andrew Nealen, Takeo Igarashi, Olga Sorkine, and Marc Alexa. Laplacian mesh optimization. In *Proceedings of the 4th international conference on Computer graphics and interactive techniques in Australasia and Southeast Asia*, pages 381–389, 2006. 1

[10] Julian Ost, Issam Laradji, Alejandro Newell, Yuval Bahat, and Felix Heide. Neural point light fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18419–18429, 2022. 1, 2

[11] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019. 1

[12] Nikhila Ravi, Jeremy Reizenstein, David Novotny, Taylor Gordon, Wan-Yen Lo, Justin Johnson, and Georgia Gkioxari. Accelerating 3d deep learning with pytorch3d. *arXiv:2007.08501*, 2020. 1

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

[14] 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

[15] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, et al. Scalability in perception for autonomous driving: Waymo open dataset. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2446–2454, 2020. 1, 2

[16] Haithem Turki, Deva Ramanan, and Mahadev Satyanarayanan. Mega-nerf: Scalable construction of large-scale nerfs for virtual fly-throughs. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12922–12931, 2022. 3, 4

[17] Yuanbo Xiangli, Linning Xu, Xingang Pan, Nanxuan Zhao, Anyi Rao, Christian Theobalt, Bo Dai, and Dahua Lin. Bungeenerf: Progressive neural radiance field for extreme multi-scale scene rendering. In *European conference on computer vision*, pages 106–122. Springer, 2022. 3, 4

[18] Qiangeng Xu, Zexiang Xu, Julien Philip, Sai Bi, Zhixin Shu, Kalyan Sunkavalli, and Ulrich Neumann. Point-nerf: Point-based neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5438–5448, 2022. 4

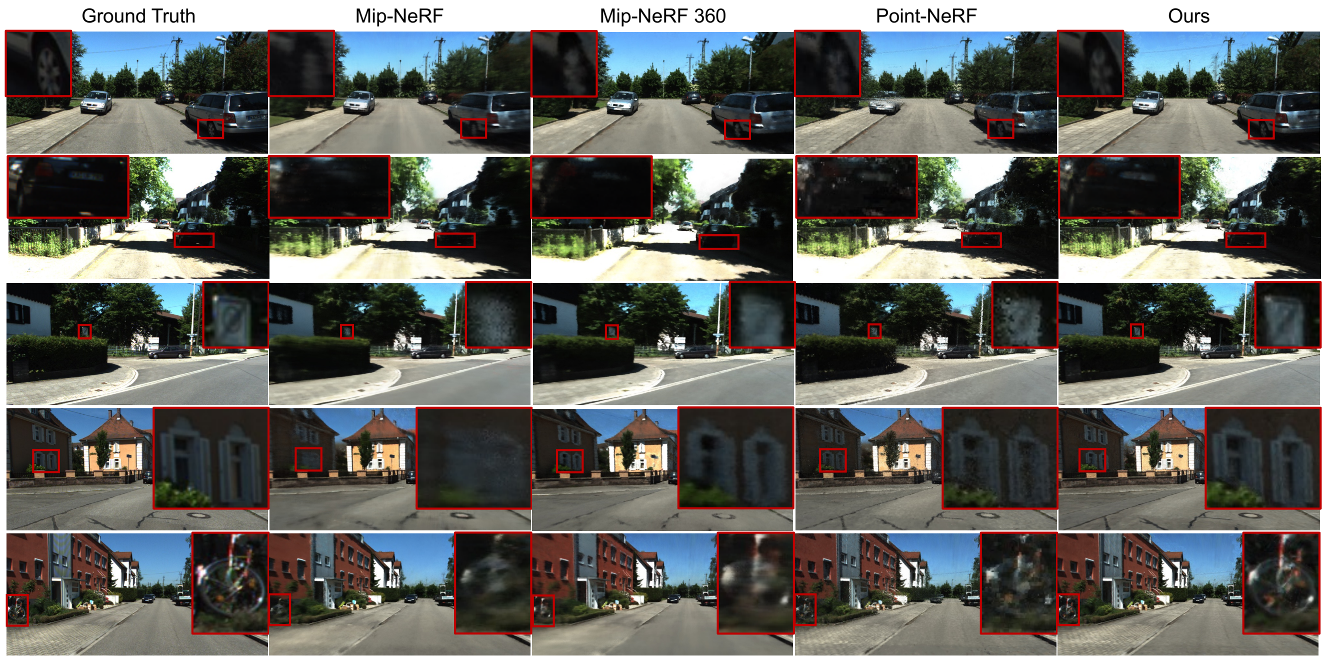

Figure S-6. Qualitative comparison of novel view synthesis on KITTI-360 dataset. Cropped patches are scaled to highlight details. Due to the explicit and accurate surface modeling, the proposed method significantly outperforms baseline methods and can effectively recover texture details.

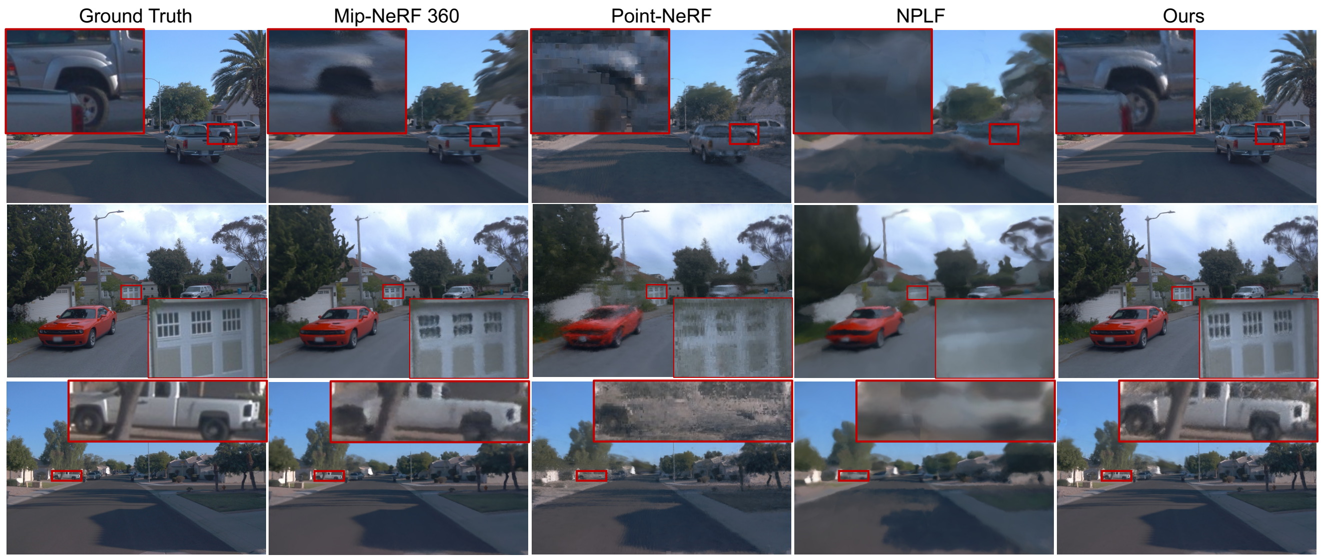

Figure S-7. Qualitative comparison of novel view synthesis on Waymo dataset. Cropped patches are scaled to highlight details. Due to the explicit and accurate surface modeling, the proposed method significantly outperforms baseline methods and can effectively recover texture details.