

This is an Open Access document downloaded from ORCA, Cardiff University's institutional repository:<https://orca.cardiff.ac.uk/id/eprint/174989/>

This is the author's version of a work that was submitted to / accepted for publication.

Citation for final published version:

Zeng, Xiaoyi, Song, Kaiwen, Yang, Leyuan, Deng, Bailin and Zhang, Juyong 2025. Oblique-MERF: Revisiting and improving MERF for oblique photography. Presented at: International Conference on 3D Vision, Singapore, 25-28 March 2025.

Publishers page:

Please note:

Changes made as a result of publishing processes such as copy-editing, formatting and page numbers may not be reflected in this version. For the definitive version of this publication, please refer to the published source. You are advised to consult the publisher's version if you wish to cite this paper.

This version is being made available in accordance with publisher policies. See <http://orca.cf.ac.uk/policies.html> for usage policies. Copyright and moral rights for publications made available in ORCA are retained by the copyright holders.



Oblique-MERF: Revisiting and Improving MERF for Oblique Photography

Supplementary Material

A. Additional Experimental Settings

A.1. Datasets

To mitigate variations in lighting and shadows within the scene, our *Campus-Oblique* dataset is captured under consistent, cloudy conditions. We adopt a surround-style capturing method, known for its higher overlap rate, which is superior to traditional grid-style capturing techniques. The datasets were recorded at altitudes ranging from 150 to 180 meters. Meanwhile, our *Campus-extra* dataset is captured at approximately 150 meters altitude, encompassing an area of 120,000 square meters for the training set. For the test set, we captured two high-rise buildings in the area, each 80 meters in height. The scene was recorded at altitudes ranging from 40 to 80 meters.

Camera poses were estimated using colmap [7], employing a vocabulary tree for feature matching. This process was augmented with a hierarchical mapper and several rounds of triangulation and bundle adjustment to refine the camera pose estimations.

A.2. Implementation Details

For the experiments in Section 5.1 and Section 5.3, we conduct training over 80K iterations, while Section 5.2 undergoes 50K iterations, all with a batch size of 32,768 pixels. We utilize the Adam optimizer, with the learning rate exponentially decaying. The occupancy plane’s learning rate drops from 5×10^{-5} to 1×10^{-5} , while for other variables, it drops from 1×10^{-2} to 1×10^{-3} . The Adam optimizer’s hyperparameters β_1 , β_2 and ϵ are set to 0.9, 0.99 and 1×10^{-15} respectively. Training losses are initially balanced with $\lambda_1 = 1.0$, $\lambda_2 = 1.0$, $\lambda_3 = 0.01$, $\lambda_4 = 1.0$, $\lambda_5 = 0.05$, $\lambda_6 = 0.001$, $\lambda_8 = 0.1$ at the beginning. Moreover, we set $\lambda_7 = 0.0$ in the first 10K iterations for warmup training. From the 10,000th iteration onwards, λ_7 is adjusted to 1×10^{-4} , and every 2,000 iterations thereafter, we increase this specific loss weight by a factor of 1.5 up to 0.2. The parameters Σ_1 and Σ_2 for smoothness regularization are set to 0.3. We sample 100 rays for smooth loss, 2^{10} rays for entropy loss, and 2^{14} samples for sparsity loss.

Similar to Mip-NeRF 360 [1], our method incorporates hierarchical sampling during the training phase, necessitating the use of two Proposal MLPs. Each Proposal MLP comprises two layers with 64 hidden units and utilizes hash encoding. For the decoder MLP that generates density and color, we utilize a 3-layer MLP with 64 hidden units per layer. This MLP generates an 8-dimensional output vector, including density, diffuse colors, and a 4-dimensional vector for view-dependent features. For deferred view-

dependency model, we employ a 3-layer MLP with 16 hidden units, and the viewing directions are encoded using positional encoding, with the level set to 4.

A.3. Comparative Method Settings

For MobileNeRF [2], we initialized a $192 \times 192 \times 192$ grid to generate polygonal meshes while maintaining default parameters for other settings. We used the open-source version¹ for BakedSDF [14], setting the batch size to 16,384 and conducting training in two phases: 20,000 and 50,000 epochs, respectively. As for 3DGS [3], we employed its official implementation with default configurations. Due to the unavailability of the sparse SfM point cloud for the *Matrix City* [4] dataset, we did not conduct experiments on it, considering the consistency of training data poses.

B. Complete results

B.1. Real-time rendering on oblique photography

We present the quantitative comparisons for all scenes in the *Campus-Oblique* and *Matrix City* datasets in Tab. 2 and Tab. 3, respectively. Fig. 1 provides qualitative comparisons for all three scenes of the *Campus-Oblique* dataset.

B.2. Color for extrapolation novel viewpoints

We show the qualitative comparisons on the *Campus-extra* dataset in Fig. 2.

C. Real-time Rendering

C.1. Sampling strategy

Since we obtain a multi-resolution occupancy plane, we start by determining sampling points from the coarsest level. Consistent with the training process, for a sample point $\mathbf{p}_i = (x_i, y_i, z_i)$, we project it onto the XY plane and find the nearest sample grid point to obtain the occupancy interval $[\bar{z}_{min}, \bar{z}_{max}]$. If z_i lies within this interval, the point either proceeds to the next level of occupancy plane evaluation or is selected for sampling at the current level. Then, based on the position of z_i and the projected grid cell $C = \{(x, y) | x \in [\bar{x}_{min}, \bar{x}_{max}], y \in [\bar{y}_{min}, \bar{y}_{max}]\}$, we determine the bounding box as following:

$$\text{BoundingBox}(x_i, y_i, z_i; \mathcal{P}_o) = \begin{cases} \{(x, y, z) | C \wedge z \in [z_{min}, \bar{z}_{min}]\}, & \text{if } z \in [z_{min}, \bar{z}_{min}], \\ \{(x, y, z) | C \wedge z \in [\bar{z}_{min}, \bar{z}_{max}]\}, & \text{if } z \in [\bar{z}_{min}, \bar{z}_{max}], \\ \{(x, y, z) | C \wedge z \in [\bar{z}_{max}, z_{max}]\}, & \text{if } z \in [\bar{z}_{max}, z_{max}]. \end{cases}$$

¹<https://github.com/hugoyocj/torch-baked sdf>

Here z_{\min} and z_{\max} represent the global lower and upper bounds of the scene along the z-axis, respectively. The next candidate point is determined as the subsequent intersection of the ray with this bounding box.

C.2. Visualization for Sampling Point Distribution

Fig. 3 provides a comparison of real-time rendering quality and sampling efficiency between our method and the baseline MERF [6]. Our spatial regularization technique particularly focuses on constraining the scene geometry orthogonal to the ground, facilitating the recovery of richer surface details, such as small objects on rooftops. Furthermore, we exhibit a more consistent appearance with the training dataset, as evidenced by the hues of the trees in the images. Remarkably, our optimized occupancy plane effectively excludes empty regions and concentrates sampling points in critical areas, such as geometric surfaces. This significantly reduces computational overhead and achieves consistently higher frame rates across multiple scales compared to the baseline method.

D. More Ablations

We present additional ablation experiments in Tab. 1, providing quantitative results regarding their impact on rendering quality and storage consumption.

Occupancy plane initialization For initialization, we estimate the upper and lower bounds of the occupancy plane based on camera poses or sparse point clouds. In “Init with 20% Larger” and “Init with 50% Larger”, we respectively expand the initialization boundaries of the occupancy plane by 20% and 50%. It can be seen that proper initialization of the sampling space range facilitates rapid convergence in optimization. Arbitrarily setting initial values far beyond the scene range may result in reduced spatial compression efficiency and decreased reconstruction quality.

Training strategy In “60k Training Iters”, we decrease the training epochs from 80K to 60K, while in “100k Training Iters”, we increase them from 80K to 100K. Lastly, in “Smaller Hash Table”, we decrease the number of entries in the multi-resolution hash encoding from 2^{22} to 2^{21} . It is indicated that sufficient training time and an adequate amount of feature representation are crucial for large-scale scenes. Moreover, extending the training time does not yield significant improvements in memory consumption, implying that our regularization technique achieves convergence within a finite number of epochs, while effectively balancing rendering fidelity and storage optimization.

	PSNR \uparrow	SSIM \uparrow	LPIPS \downarrow	VRAM(MB) \downarrow
(a) Init with 20% Larger	24.21	0.719	0.224	<u>110.8</u>
(b) Init with 50% Larger	24.01	0.690	0.245	123.4
(c) 60k Training Iters	24.22	0.708	0.231	113.9
(d) 100k Training Iters	24.40	0.719	0.217	111.6
(e) Smaller Hash Table	23.84	0.692	0.247	108.2
Ours	<u>24.33</u>	<u>0.716</u>	<u>0.222</u>	111.9

Table 1. Ablations on initialization and training strategy.

E. Further Discussion

Similar to our work, recent efforts have focused on accelerating rendering in NeRF-like methods by regularizing the sampling space. Due to the lack of open-source implementations, we cannot provide comparative experimental results. However, we will discuss the relevant methods.

Adaptive Shells [11] incorporates a spatially-varying kernel in the NeuS [10] formulation and requires extracting two layers of explicit meshes to guide its sampling. HybridNeRF [9] optimizes the surfaceness of points in a 3D voxel grid following the VolSDF’s approach [13], thereby approximating SDF representation for foreground modeling and conducting Sphere Tracing sampling within it.

Both of these methods rely on surface modeling, and when applied to large-scale datasets, adaptive sampling optimization typically requires high-resolution representation, leading to significant memory consumption and optimization challenges. In contrast, we optimize the sampling space by improving the volume rendering formulation and, considering the characteristics of oblique photography, offer a straightforward, memory-efficient, and easily applicable representation.

References

- [1] Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In *CVPR*, 2022. 1, 4
- [2] Zhiqin Chen, Thomas Funkhouser, Peter Hedman, and Andrea Tagliasacchi. Mobilenerf: Exploiting the polygon rasterization pipeline for efficient neural field rendering on mobile architectures. In *CVPR*, 2023. 1, 4
- [3] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42(4), 2023. 1, 4
- [4] Yixuan Li, Lihan Jiang, Linning Xu, Yuanbo Xiangli, Zhenzhi Wang, Dahua Lin, and Bo Dai. Matrixcity: A large-scale city dataset for city-scale neural rendering and beyond. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3205–3215, 2023. 1, 4
- [5] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a mul-

- tiresolution hash encoding. *ACM transactions on graphics (TOG)*, 41(4):1–15, 2022. 4
- [6] Christian Reiser, Rick Szeliski, Dor Verbin, Pratul Srinivasan, Ben Mildenhall, Andreas Geiger, Jon Barron, and Peter Hedman. Merf: Memory-efficient radiance fields for real-time view synthesis in unbounded scenes. *ACM Transactions on Graphics (TOG)*, 42(4):1–12, 2023. 2, 4, 5
- [7] Johannes L. Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4104–4113, 2016. 1
- [8] Haithem Turki, Deva Ramanan, and Mahadev Satyanarayanan. Mega-nerf: Scalable construction of large-scale nerfs for virtual fly-throughs. In *CVPR*, pages 12922–12931, 2022. 4
- [9] Haithem Turki, Vasu Agrawal, Samuel Rota Bulò, Lorenzo Porzi, Peter Kotschieder, Deva Ramanan, Michael Zollhöfer, and Christian Richardt. Hybridnerf: Efficient neural rendering via adaptive volumetric surfaces. In *Computer Vision and Pattern Recognition (CVPR)*, 2024. 2
- [10] Peng Wang, Lingjie Liu, Yuan Liu, Christian Theobalt, Taku Komura, and Wenping Wang. Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. In *Advances in Neural Information Processing Systems*, pages 27171–27183. Curran Associates, Inc., 2021. 2
- [11] Zian Wang, Tianchang Shen, Merlin Nimier-David, Nicholas Sharp, Jun Gao, Alexander Keller, Sanja Fidler, Thomas Muller, and Zan Gojcic. Adaptive shells for efficient neural radiance field rendering. *ACM Transactions on Graphics (TOG)*, 42:1 – 15, 2023. 2
- [12] Linning Xu, Yuanbo Xiangli, Sida Peng, Xingang Pan, Nanxuan Zhao, Christian Theobalt, Bo Dai, and Dahua Lin. Grid-guided neural radiance fields for large urban scenes. In *CVPR*, pages 8296–8306. IEEE, 2023. 4
- [13] Lior Yariv, Jiatao Gu, Yoni Kasten, and Yaron Lipman. Volume rendering of neural implicit surfaces. In *Thirty-Fifth Conference on Neural Information Processing Systems*, 2021. 2
- [14] Lior Yariv, Peter Hedman, Christian Reiser, Dor Verbin, Pratul P. Srinivasan, Richard Szeliski, Jonathan T. Barron, and Ben Mildenhall. Baked sdf: Meshing neural sdf for real-time view synthesis. In *ACM SIGGRAPH 2023 Conference Proceedings, SIGGRAPH 2023, Los Angeles, CA, USA, August 6-10, 2023*, pages 46:1–46:9. ACM, 2023. 1, 4

	<i>Scene-1</i>			<i>Scene-2</i>			<i>Scene-3</i>		
	PSNR \uparrow	SSIM \uparrow	LPIPS \downarrow	PSNR \uparrow	SSIM \uparrow	LPIPS \downarrow	PSNR \uparrow	SSIM \uparrow	LPIPS \downarrow
InstantNGP [5]	23.48	0.653	0.415	23.47	0.587	0.535	<u>22.28</u>	0.560	0.565
Nerfacto [1]	21.24	0.570	0.366	23.80	<u>0.685</u>	<u>0.256</u>	21.57	0.598	0.384
Mega-NeRF [8]	23.65	0.648	0.437	24.88	0.632	0.388	21.86	0.494	0.573
MobileNeRF [2]	20.46	0.446	0.490	21.21	0.420	0.508	-	-	-
BakedSDF [14]	21.14	0.526	0.501	22.37	0.499	0.546	20.20	0.455	0.671
3DGS [3]	<u>24.32</u>	0.766	0.215	23.82	0.714	0.245	21.09	0.540	0.503
MERF [6]	23.31	0.670	0.264	23.53	0.647	0.284	21.67	0.535	0.485
Ours	24.33	<u>0.716</u>	<u>0.222</u>	<u>24.20</u>	0.680	0.261	22.53	<u>0.589</u>	<u>0.410</u>

Table 2. Quantitative results on *Campus-Oblique* datasets. “-” means that due to the huge amount of data exceeding the memory capacity, corresponding comparative experiments cannot be performed.

	<i>Block-A</i>			<i>Block-D</i>		
	PSNR \uparrow	SSIM \uparrow	LPIPS \downarrow	PSNR \uparrow	SSIM \uparrow	LPIPS \downarrow
InstantNGP [5]	24.04	0.636	0.716	22.15	0.587	0.697
Nerfacto [1]	23.99	0.689	<u>0.454</u>	22.80	0.659	0.433
Mega-NeRF [8]	25.24	0.669	0.603	24.16	0.631	0.597
Grid-NeRF [12]	<u>25.37</u>	<u>0.705</u>	0.536	<u>24.75</u>	0.702	0.487
MobileNeRF [2]	16.55	0.508	0.671	21.48	0.526	0.544
BakedSDF [14]	22.37	0.603	0.679	21.81	0.561	0.625
MERF [6]	24.51	0.673	0.522	24.48	0.680	<u>0.385</u>
Ours	25.72	0.710	0.430	24.88	<u>0.701</u>	0.382

Table 3. Quantitative results on *Matrix City* [4] datasets.



Figure 1. Comparison on rendering quality for novel views between Oblique-MERF and other methods on the *Campus-Oblique* and *Matrix City* [4] dataset. From left to right are ground truth, Oblique-MERF, MERF, Mega-NeRF, and Instant-NGP.

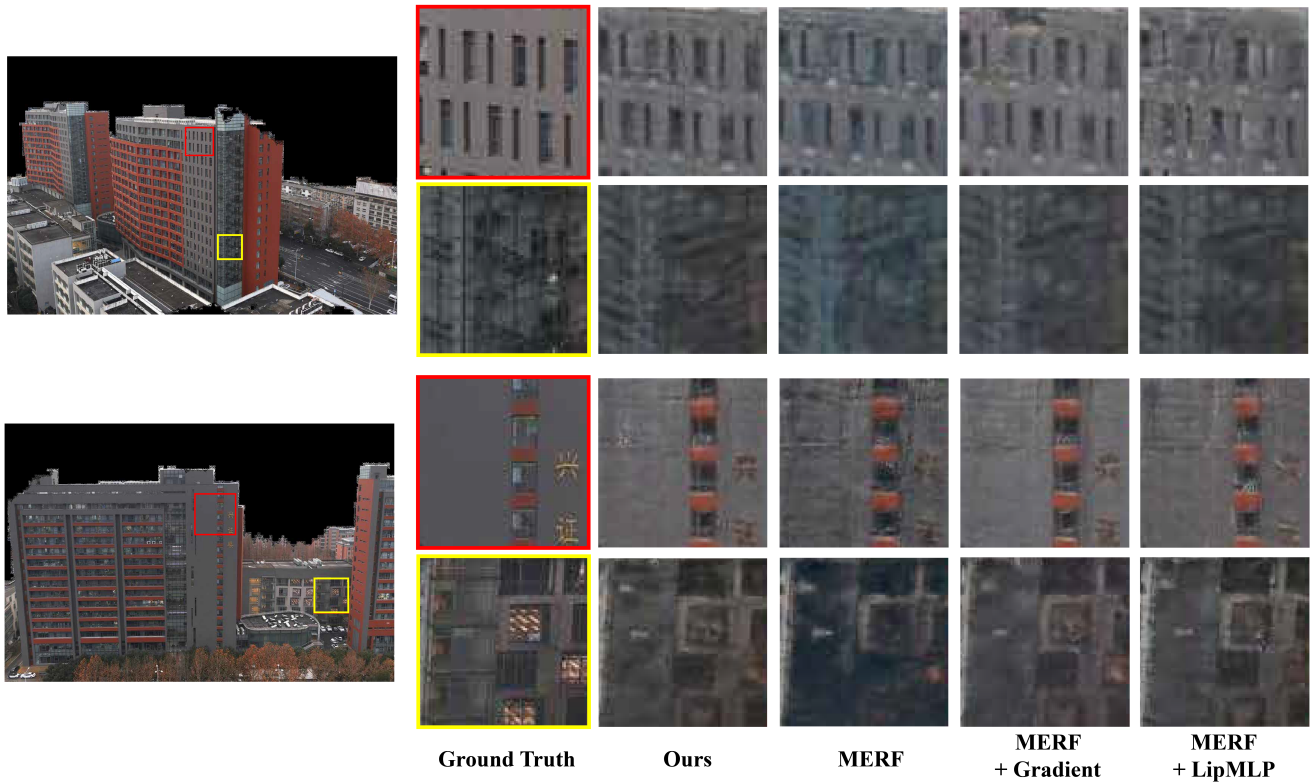


Figure 2. Rendering quality comparison between Oblique-MERF and other MERF variants for test views on the *Campus-extra* dataset. There are fewer artifacts in the rendering results of our method.

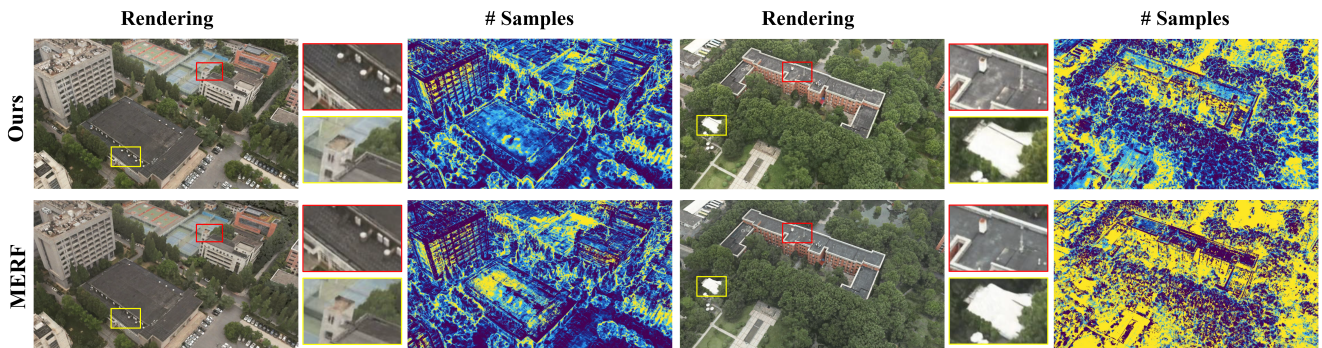



Figure 3. Visualization of Oblique-MERF and the baseline MERF [6] for real-time rendering and sampling point distribution on the *Campus-Oblique* dataset. In “#sample”, we utilize the color bar  to represent the number of sampling points, transitioning from blue to yellow to indicate an increase in sampling points from 4 to 16.