Monocular Online Reconstruction with Enhanced Detail Preservation Supplementary Materials

SONGYIN WU, University of California Santa Barbara, USA
ZHAOYANG LV, Meta Reality Labs Research, USA
YUFENG ZHU, Meta Reality Labs Research, USA
DUNCAN FROST, Meta Reality Labs Research, United Kingdom
ZHENGQIN LI, Meta Reality Labs Research, USA
LING-QI YAN, University of California Santa Barbara, USA
CARL REN, Meta Reality Labs Research, USA
RICHARD NEWCOMBE, Meta Reality Labs Research, USA
ZHAO DONG, Meta Reality Labs Research, USA

1 OPTIMIZATION LOSSES

For each keyframe, it will jointly optimize with the selected local views and global views for k_{opt} iterations, where the local views are fixed for all iterations and the global views are re-selected for every iteration. For the first k_{cam} iterations, the camera pose refinement is turned on for global views. We use the following losses for optimization:

$$\mathcal{L} = \mathcal{L}_{\text{render}} + \lambda_g \mathcal{L}_g \tag{1}$$

and $\mathcal{L}_{\text{render}}$ is defined as

$$\mathcal{L}_{\text{render}} = \frac{1}{|O|} \sum_{w \in O} (|\bar{I}_o - I_o|_1 + \lambda_{\text{dssim}} \mathcal{L}_{\text{dssim}}(\bar{I}_o, I_o)) \tag{2}$$

where O refers to the optimization window including the current view, the local views and the global views. $\mathcal{L}_{\text{dssim}}$ refers to the DSSIM [Baker et al. 2022] loss. The Gaussian loss $\mathcal{L}_g$ aims to avoid extreme anisotropic Gaussians and is defined as

$$\mathcal{L}_g = \frac{1}{|G|} \sum_{1}^{|G|} (\max(0, \max(\mathbf{s}) - \min(\mathbf{s}) * \delta_g))^2 \tag{3}$$

where we set λ_g to 1.0, δ_g to 10.0, and λ_{dssim} to 0.2. $\mathbf{s}$ refers to the Gaussian scale parameters.

2 ARIA SEQUENCES

We use Aria glasses [Engel et al. 2023] to capture three real-world sequences, demonstrating that our framework can reconstruct details across various levels. Table 1 provides basic information about the captured sequences.

Pre-processing. The raw captured images are fisheye images. We first rectify them to a perspective pinhole camera model with a resolution of 800×800. The rectified images exhibit vignette effects, which we remove by dividing them by the vignette correction map. The processed images are then used in our framework and the baselines for both tracking and mapping modules to ensure a fair comparison. An illustration is shown in Fig. 1.

Table 1. Information of Aria captured sequences.

	Number of Frames	Rectified Resolution	Type
Steak House	1114	800×800	Outdoor
Bike Shop	999	800×800	Outdoor
Focus Room	1164	800×800	Indoor

Fig. 1. Illustration of Aria captured images pre-processing. The raw captured image's resolution is 2880×2880.

3 MULTI-LEVEL OCCUPANCY HASH VOXELS ALGORITHM

Algorithm 1 describes the process for updating and querying whether a voxel is occupied at the specified scene scales. S_{init} refers to the coarsest scale, L refers to the number of levels, and n refers to the resolution of each axis (also the size of the occupancy map arrays). The pseudocode template is from Zhou et al. [2024].

4 ADDITIONAL EXPERIMENTS ABOUT POST REFINEMENT

While our method achieves high-quality results even before the post-refinement process, the post-refinement further enhances the quality. In contrast, baseline methods heavily depend on the post-refinement process, as they lack explicit mechanisms for optimizing the map with global consistency. Fig. 2 shows the visual comparison of the post-refinement process for different methods.

Authors' addresses: Songyin Wu, s_wu975@ucsb.edu, University of California Santa Barbara, USA; Zhaoyang Lv, zhaoyang@meta.com, Meta Reality Labs Research, USA; Yufeng Zhu, yufengzhu@meta.com, Meta Reality Labs Research, USA; Duncan Frost, frost@meta.com, Meta Reality Labs Research, United Kingdom; Zhengqin Li, zhl@meta.com, Meta Reality Labs Research, USA; Ling-Qi Yan, lingqi@cs.ucsb.edu, University of California Santa Barbara, USA; Carl Ren, carlren@meta.com, Meta Reality Labs Research, USA; Richard Newcombe, newcombe@meta.com, Meta Reality Labs Research, USA; Zhao Dong, zhaodong@meta.com, Meta Reality Labs Research, USA.

```
class MOHV:
    S_init, L, n, occ[L][n^3]

    def hash(x):
        return (x^((x * 2654435761)^(x * 805459861))) % n

    def query(p, s):
        cur_s = S_init
        res = True
        for l in range(L):
            if cur_s < s:
                break
            x, y, z = int(p.x / cur_s), int(p.y / cur_s), int
                (p.z / cur_s)
            x, y, z = hash(x), hash(y), hash(z)
            i = x * n * n + y * n + z
            res = res & occ[l][i]
            cur_s = cur_s / 2.0
        return res

    def update(p, s):
        cur_s = S_init
        for l in range(L):
            if cur_s < s:
                break
            x, y, z = int(p.x / cur_s), int(p.y / cur_s), int
                (p.z / cur_s)
            x, y, z = hash(x), hash(y), hash(z)
            i = x * n * n + y * n + z
            occ[l][i] = True
            cur_s = cur_s / 2.0
```

Listing 1: Pseudocode for the query and update of MOHV.

5 NON-KEYFRAMES TRACKING

Since only the poses of keyframes are globally optimized, we also deform the non-keyframe poses to align them with the refined poses of the keyframes. This concept is similar to creating submaps, where the poses of non-keyframes are adjusted based on anchor poses. Instead of explicitly creating submaps, we utilize the relative pose changes between the previous and next keyframe poses. Given a non-keyframe pose c_k, let i and j be the indices of its previous and next keyframe cameras, and $c_i, \bar{c}_i$ be the pose before and after adjustment, respectively. The deformed camera pose of $\bar{c}_k$ is calculated by

$$\bar{c}_k = c_k \times \text{interp}(c_i, c_j, \alpha_k)^{-1} \times \text{interp}(\bar{c}_i, \bar{c}_j, \alpha_k) \tag{4}$$

where $interp(\cdot)$ refers to the linear interpolation in $SE(3)$ and α refers to the relative time scale of camera k from camera i to camera j.

6 MAPPING-ONLY COMPARISON

Our method centers on developing a high-quality mapping system, using ORB-SLAM3 [Campos et al. 2021] as an example in the main paper. The reconstructed results produced by our system consistently achieve higher quality than those of the baselines, even when the baselines are equipped with better tracking systems. This demonstrates the effectiveness and robustness of our mapping framework.

Table 2. Quantitative Comparison of MonoGS and our method using the same tracking system. Our method not only shows better reconstruction quality, but also uses fewer Gaussians and is more efficient. MonoGS+O refers to MonoGS with ORB-SLAM3 tracked poses. w PR. refers to with a post-refinement process.

	PSNR	SSIM	LPIPS	# Gaussians
MonoGS+O	21.67	0.576	0.558	462,426
MonoGS+O (w PR.)	23.32	0.624	0.502	462,426
Ours	**26.64**	**0.692**	**0.320**	**348,187**

Table 3. Peak memory usage of our pipeline on Aria sequences. Memory consumption is reported in gigabytes (GB).

	SteakHouse	BikeShop	FocusRoom
# Gaussians	332,024	315,130	375,733
GPU Memory (recon)	5.73	4.71	5.36
GPU Memory (render)	0.137	0.128	0.135

Furthermore, our pipeline is designed to be compatible with a variety of tracking systems, whereas some baseline methods exhibit a strong coupling between tracking and mapping components, making it non-trivial to replace their tracking modules with the one we employ.

To further highlight the advantages of our mapping system, we conduct a comparison with MonoGS [Matsuki et al. 2024] by using the same tracking system, as MonoGS uses relatively independent tracking and mapping components. Specifically, we extract camera poses from ORB-SLAM3 and input them into both our pipeline and MonoGS, disabling the tracking modules in both systems. The results obtained through this approach for our pipeline are slightly improved compared to those reported in the main paper, due to the higher tracking accuracy achieved when the tracking and mapping processes are run separately.

Table 2 presents the quantitative comparison between our method and MonoGS under the same tracking system, and Fig. 3 provides the corresponding visual comparisons. Although the quality of MonoGS improves with access to a better tracking system, it still underperforms relative to our method, exhibiting missing details despite utilizing more Gaussians, operating at a slower speed, and incorporating a post-refinement process.

7 MEMORY USAGE

We report the number of Gaussians, the peak memory usage during online reconstruction, and the memory consumption during rendering after reconstruction in Table. 3, to provide a clearer understanding of the performance and scalability of our method. The memory usage results indicate the potential to extend our approach to even larger scenes, while the cost of rendering after reconstruction remains significantly lower. Additional discussions on scalability are provided in Sec. 9.

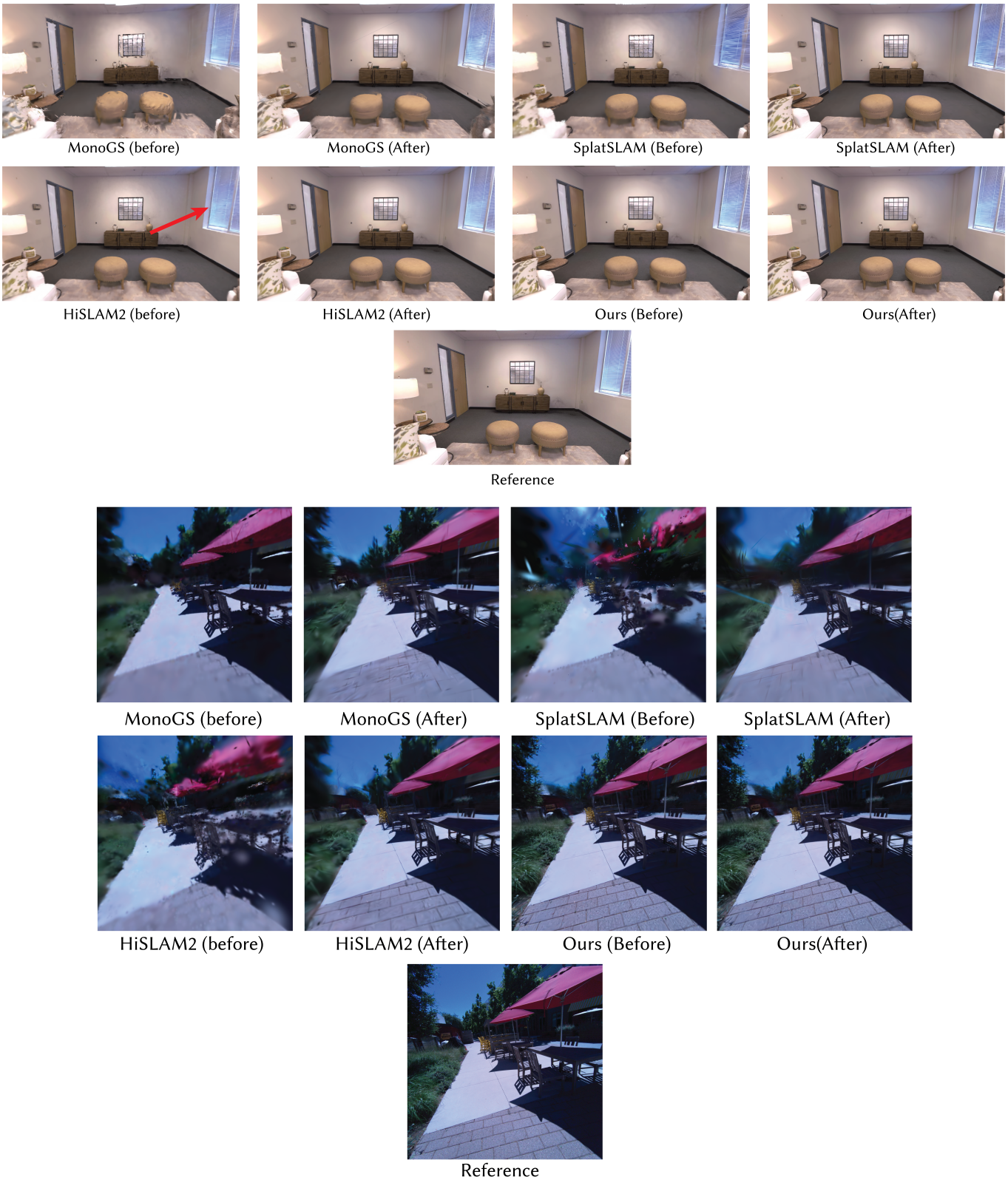

Fig. 2. Qualitative comparison of baselines and our method about the importance of the post-refinement module. Our method does not highly rely on the post-refinement process while baselines improve a lot through such post-refinement process.

Fig. 3. Qualitative comparison of MonoGS and our method using the same tracking system. Our results show better reconstruction quality with finer details.

Fig. 4. Failure cases of our method. *Left*: Misaligned ghosting artifacts caused by severely inaccurate and shifted tracked camera poses. *Middle*: Overfitting artifacts arising in the presence of dynamic objects within the scene. *Right*: Low-quality reconstruction in regions that are poorly observed.

8 LIMITATIONS

Our pipeline is capable of reconstructing scenes with fine details at interactive frame rates. However, it still encounters failure cases under certain challenging scenarios. Representative visual examples are presented in Fig. 4.

Shifted camera poses: Although our pipeline incorporates a camera refinement module, it cannot effectively handle cases where the tracked poses are significantly shifted or completely lost. The left image in Fig. 4 shows the reconstructed result under highly inaccurate camera poses, leading to severe duplicated ghosting artifacts.

Dynamic objects: The middle image illustrates a shadow cast by the person capturing the scene, which should not be present in the reconstructed result. Since our pipeline does not explicitly detect dynamic objects, it overfits to these artifacts during reconstruction. Incorporating a dynamic object detector to identify and remove such elements would likely improve the reconstruction quality, and we leave this for future work.

Unobserved regions: As shown in the right image, our pipeline fails to reconstruct plausible results in unobserved regions due to the absence of strong priors for completing such areas. Incorporating a pretrained generative model could potentially improve reconstruction in these regions; however, the associated computational overhead presents a significant challenge. We leave this direction for future work.

9 DISCUSSION OF LARGE-SCALE SCENES

The actual scale of the scenes used in our experiments ranges from single-room indoor environments (e.g., TUM, Replica, and the FocusRoom sequence in the Aria captured datasets) to large-scale outdoor scenes (e.g., BikeShop and SteakHouse in the Aria captured datasets).

To accommodate larger scenes, our MOHV module efficiently manages the Gaussian distributions by controlling their quantity and removing redundant Gaussians as sequences or scenes expand, ensuring that Gaussians are placed only where necessary. Additionally, the global cameras are not uniformly selected from all historical keyframes; instead, we design a heuristic weighting strategy that prioritizes recent and high-error regions over distant, well-reconstructed areas. Together, these two modules prevent the uncontrolled growth of Gaussians and ensure that optimization focuses on recently under-reconstructed areas, thereby demonstrating the scalability of our mapping system.

However, scaling to very large scenes does require a reliable tracking system without significant pose drift, or alternatively, explicit integration of loop closure mechanisms into the mapping process, as seen in methods like SplatSLAM [Sandström et al. 2024] and PhotoSLAM [Huang et al. 2024]. The primary goal of this work is to develop a mapping system that operates effectively with general tracking pipelines where loop closure may not be available; thus, we leave those explorations as future work.

10 IMPLEMENTATION

We implement our framework in Python using PyTorch [Paszke et al. 2019] and gsplats [Ye et al. 2024] for Gaussian optimization. We optimize $k_{opt} = 25$ steps for each keyframe where the first $k_{cam} = 8$ steps also optimize camera poses, and each step contains 6 views, including one current view, one local view, and four global views. We use multi-resolution optimization for large scenes (Aria) and full resolution optimization for small scenes (TUM and Replica) as it achieves higher quality.

For TUM-RGBD [Sturm et al. 2012] dataset, we select *fr1_desk*, *fr2_xyz*, and *fr3_office* for evaluation. For Replica [Straub et al. 2019] dataset, we select *room0*, *room1*, *room2*, *office0*, *office1*, *office2*, *office3*, and *office4* for evaluation.

We use officially released codes for all baselines (unless specified) including offline 3DGS evaluation. Offline 3DGS uses Aria tracked poses as they are more accurate than ORB-SLAM3 tracked poses.

Depth Estimator. We employ the pretrained model from DepthCov [Dexheimer and Davison 2023] for our error compensation module. Fig. 5 shows a visualization of the depth estimator. Importantly, we do not estimate depth values for the entire image; instead, we only estimate depth at selected pixels identified by our error compensation algorithm. This strategy reduces the depth estimation cost per frame by approximately 70x (727.40 ms per frame for full-frame depth estimation versus 10.44 ms per frame in our pipeline for a 800×800 resolution image). Note that error compensation and depth estimation are performed only on keyframes.

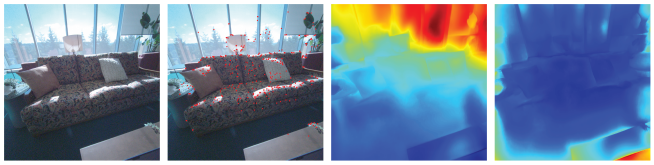

Fig. 5. Visualization of the depth estimation. The images from left to right are: the input image, the sparse tracked pixels from the tracking system, the estimated depth map, and the corresponding variance map. Red indicates higher values and blue indicates lower values, where higher variance corresponds to lower confidence in the depth estimation.

REFERENCES

Allison H Baker, Alexander Pinard, and Dorit M Hammerling. 2022. Dssim: a structural similarity index for floating-point data. *arXiv preprint arXiv:2202.02616* (2022).

Carlos Campos, Richard Elvira, Juan J Gómez Rodríguez, José MM Montiel, and Juan D Tardós. 2021. Orb-slam3: An accurate open-source library for visual, visual–inertial, and multimap slam. *IEEE Transactions on Robotics* 37, 6 (2021), 1874–1890.

Eric Dexheimer and Andrew J. Davison. 2023. Learning a Depth Covariance Function. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Jakob Engel, Kiran Somasundaram, Michael Goesele, Albert Sun, Alexander Gamino, Andrew Turner, Arjang Talattof, Arnie Yuan, Bilal Souti, Brighid Meredith, et al. 2023. Project aria: A new tool for egocentric multi-modal ai research. *arXiv preprint arXiv:2308.13561* (2023).

Huajian Huang, Longwei Li, Hui Cheng, and Sai-Kit Yeung. 2024. Photo-SLAM: Real-time Simultaneous Localization and Photorealistic Mapping for Monocular Stereo and RGB-D Cameras. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 21584–21593.

Hidenobu Matsuki, Riku Murai, Paul H. J. Kelly, and Andrew J. Davison. 2024. Gaussian Splatting SLAM. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. arXiv:1912.01703 [cs.LG] https://arxiv.org/abs/1912.01703

Erik Sandström, Keisuke Tateno, Michael Oechsle, Michael Niemeyer, Luc Van Gool, Martin R Oswald, and Federico Tombari. 2024. Splat-SLAM: Globally Optimized RGB-only SLAM with 3D Gaussians. *arXiv preprint arXiv:2405.16544* (2024).

Julian Straub, Thomas Whelan, Lingni Ma, Yufan Chen, Erik Wijmans, Simon Green, Jakob J Engel, Raul Mur-Artal, Carl Ren, Shobhit Verma, et al. 2019. The Replica dataset: A digital replica of indoor spaces. *arXiv preprint arXiv:1906.05797* (2019).

Jürgen Sturm, Nikolas Engelhard, Felix Endres, Wolfram Burgard, and Daniel Cremers. 2012. A benchmark for the evaluation of RGB-D SLAM systems. In *2012 IEEE/RSJ international conference on intelligent robots and systems*. IEEE, 573–580.

Vickie Ye, Ruilong Li, Justin Kerr, Matias Turkulainen, Brent Yi, Zhuoyang Pan, Otto Seiskari, Jianbo Ye, Jeffrey Hu, Matthew Tancik, and Angjoo Kanazawa. 2024. gsplat: An Open-Source Library for Gaussian Splatting. *arXiv preprint arXiv:2409.06765* (2024). arXiv:2409.06765 [cs.CV] https://arxiv.org/abs/2409.06765

Yang Zhou, Songyin Wu, and Ling-Qi Yan. 2024. Unified Gaussian Primitives for Scene Representation and Rendering. arXiv:2406.09733 [cs.GR] https://arxiv.org/abs/2406.09733