

Splat-based 3D Scene Reconstruction with Extreme Motion-blur

Supplementary Material

1. Ablation Study on Interpolation Methods

We evaluate the impact of different interpolation methods on deblurring performance. To generate virtual camera poses between the start and end of the exposure time (ξ_t^s and ξ_t^e), we consider several interpolation techniques: Bézier curves, cubic B-splines, and linear interpolation in $SE(3)$ space. As shown in Table 1, linear interpolation in $SE(3)$ consistently achieves the best performance across all synthetic scenes, underscoring its effectiveness in accurately modeling camera motion.

2. Visualization of Ablation Study

We analyze the output point clouds from various ablation tests corresponding to Table 3 in the main paper. As illustrated in Figure 9, our refinement step significantly reduces misalignment in the initial point cloud, leading to more accurate reconstructions. In the Bedroom scene, large structures such as walls and the ceiling exhibit notable misalignment in the initial output, whereas our refinement step achieves much better alignment in these areas. Note that the refinement process operates without relying on color information, while any erroneous color attributes are subsequently corrected during the deblurring stage. Similarly, in the Living Room scene, misalignments in finer details, such as the bookshelf, desk legs, and the pot on the desk, are effectively resolved by our refinement step, which leverages depth supervision (Section 3.2 in the main paper). Additionally, when the scale of the 3D Gaussians is not fixed during the refinement step, the resulting point cloud becomes sparser, adversely affecting both pose accuracy and deblurring performance. These findings highlight the importance of fixed-scale 3D Gaussians and robust refinement for high-quality results.

3. Comparison Details

In this section, we provide implementation details for each comparison method.

FPFH-ICP [7]. We utilize the Open3D library for both global and local ICP registration. For global alignment, given a voxel size s , we set `radius` = $s \times 2$ and `max_nn` = 30 to estimate point cloud normals. FPFH features are computed using `radius` = $s \times 5$ and `max_nn` = 100. For local refinement, we employ the point-to-plane ICP algorithm with `threshold` = $s \times 2$. Starting from the input RGB-D image sequence, we compute the camera transformation matrix for each adjacent frame pair. Subsequently, a 3D point cloud is constructed by merging depth maps based on

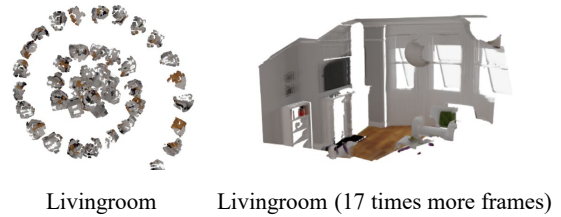


Figure 1. Reconstruction results for the living room scene: With 50 frames, SplatAM produces a spiral-shaped reconstruction. However, when the same camera trajectory is rendered with 850 frames, SplatAM successfully reconstructs the scene.

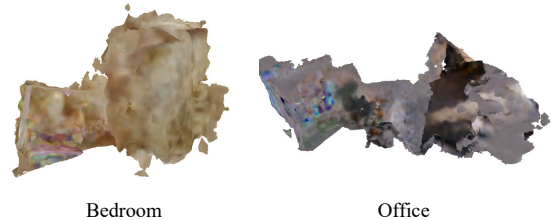


Figure 2. NICE-SLAM produces bumpy reconstructions and has high memory requirements.

the extrinsic and intrinsic camera parameters. The resulting point cloud is then downsampled using the voxel size s to ensure efficient processing.

ElasticFusion [11]. ElasticFusion is designed to process long image sequences in real time by discarding surfels with confidence values below a certain threshold. However, with the default settings, no surfels remain for our dataset. To address this, we lower the surfel confidence thresholds (`confThresh` and `confThres`) to 0.1, allowing the visualization of results even for surfels with low confidence. Additionally, we adjust the depth processing parameters to account for the characteristics of our dataset. Specifically, we increase the cutoff distance for depth processing (`depthCutoff`) to 10 meters, accommodating the wider range of input depth values. To compensate for the real-time nature of the algorithm and improve reconstruction accuracy, we increase the number of ICP iterations at each pyramid level from [10, 5, 4] to [100, 50, 40]. Finally, we observe that disabling the optical flow module enhances the quality of the 3D reconstruction in our experiments.

ORB-SLAM3 [1]. We adapted the TUM configuration file to use the PinHole camera model. Since ORB-SLAM3 does not natively support the k4, k5, and k6 distortion parameters, we preprocess the real-scene images captured with the Azure Kinect camera by undistorting them and providing the corresponding intrinsic parameters. Camera trajectories

Table 1. Ablation study on interpolation methods: This study isolates the effect of interpolation by modifying only the interpolation stage while keeping the rest of our pipeline unchanged. Notably, linear interpolation in the $SE(3)$ space consistently outperforms other methods across all scenes in our dataset, highlighting its superior effectiveness for this task.

	Bedroom			Livingroom			Office		
	PSNR \uparrow	SSIM \uparrow	LPIPS \downarrow	PSNR \uparrow	SSIM \uparrow	LPIPS \downarrow	PSNR \uparrow	SSIM \uparrow	LPIPS \downarrow
Bezier curve	25.632	0.778	0.211	26.582	0.879	0.154	25.546	0.788	0.162
Cubic B-spline	25.649	0.779	0.211	26.568	0.879	0.153	25.630	0.787	0.161
Linear interpolation	26.745	0.824	0.206	27.650	0.900	0.150	25.649	0.791	0.160

Table 2. Ablation study on the global refinement process and fixed scale of 3D Gaussians: Complementing Table 3 in the main paper, we present pose error metrics computed from aligned trajectories using the Kabsch-Umeyama algorithm [4]. This analysis highlights the impact of these design choices on pose accuracy.

	PSNR	SSIM	LPIPS	RMSE	ATE	RPE(trans)	RPE(rot)
w/o scale fix	21.656	0.703	0.308	0.027	0.127	0.051	1.418
w/o refinement	22.223	0.734	0.238	0.027	0.124	0.008	0.047
w/o depth loss	25.518	0.807	0.170	0.014	0.049	0.004	0.057
Scale fix + refinement	26.681	0.838	0.172	0.010	0.049	0.004	0.057

are saved using the `SaveTrajectoryTUM` function, and a custom function is implemented to export point clouds.

NICE-SLAM [13]. We modified the demo configuration designed for the ScanNet dataset to suit our experiments. After undistorting the input images, we updated the camera parameters accordingly and set `every_frame` to 1 in the mapping section. To enhance the output mesh resolution, the `resolution` parameter in the meshing section was increased to 1024. However, due to the lack of motion blur handling, the output exhibits bumpy and twisted geometries, as illustrated in Figure 2.

Point-SLAM [8]. We tested Point-SLAM with our synthetic dataset, but the method failed to run successfully. The failure produced results similar to SplatAM (Figure 1, left). However, when we densified the frame sequence by increasing the number of frames, the method successfully processed the data.

DroidSLAM [9]. As with other methods, we provided undistorted images as input. We modified the demo file to accept RGB-D input format. To ensure all input frames were used as keyframes, we set `filter_thresh` and `keyframe_thresh` to 0, enabling bundle adjustment on every frame. Additionally, point clouds were extracted using methods available in the visualizer code.

MonoGS [6]. We adapted the configuration file from the Replica setup to accommodate RGB-D input. To process all input frames as keyframes, we set `kf_interval` to 0 and enforced single-threaded mode.

SplatAM [5]. We modified the TUM configuration file by providing undistorted images and camera parameters. Setting `keyframe_every` to 1 resulted in a spiral-shaped reconstruction output with our dataset. To investigate further, we created an alternative video with the same camera tra-

jectory as the Living Room scene but increased the frame count by 17 \times (from 50 frames to 850 frames). With this denser video, SplatAM correctly reconstructed the scene (Figure 1, right). From this observation, we conclude that the distance between adjacent cameras plays a critical role in the reconstruction quality.

4. Additional qualitative comparisons

Comparisons with [10, 12]. We compare our deblurring performance against Deblur-GS [10] and BAD-Gaussians [12]. While both methods successfully produce deblurred outputs, our approach achieves the highest deblurring performance.

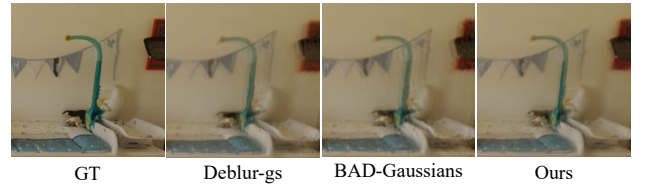


Figure 3. Deblurring comparison to [45] and [49].

Comparison with Open3D reconstruction pipeline. We compare our method with that of Choi et al. [3] using the Open3D implementation. Their approach fails due to inaccurately estimated poses caused by motion blur (Figure 4).

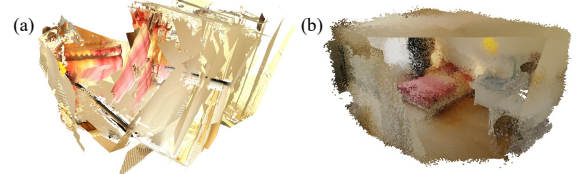


Figure 4. Comparison of (b) ours to (a) Choi et al.

Applicability of camera parameters to other networks. The camera parameters estimated by our method can be effectively utilized in RGB-only networks. We apply our estimated parameters to BAGS and Deblurring 3D GS, yet our method consistently outperforms them in terms of deblurring (Figure 5).

5. Limitations

Like other COLMAP or SLAM methods, one limitation our method has involves textureless scenes. Estimating the



GT BAGS Deblurring 3DGS Ours
Figure 5. Comparison with BAGS and Deblurring 3D GS.

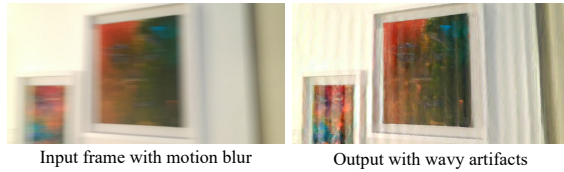


Figure 6. Failure case with a featureless scene.

camera pose becomes difficult, and the outputs suffer from wavy artifacts caused by deblurring ambiguity (Figure 6).

References

- [1] Carlos Campos, Richard Elvira, Juan J Gómez Rodríguez, José MM Montiel, and Juan D Tardós. Orb-slam3: An accurate open-source library for visual, visual-inertial, and multi-map slam. *IEEE Transactions on Robotics*, 37(6):1874–1890, 2021. 1
- [2] Liangyu Chen, Xiaojie Chu, Xiangyu Zhang, and Jian Sun. Simple baselines for image restoration. *arXiv preprint arXiv:2204.04676*, 2022. 4
- [3] Sungjoon Choi, Qian-Yi Zhou, and Vladlen Koltun. Robust reconstruction of indoor scenes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5556–5565, 2015. 2
- [4] Michael Grupp. evo: Python package for the evaluation of odometry and slam. <https://github.com/MichaelGrupp/evo>, 2017. 2
- [5] Nikhil Keetha, Jay Karhade, Krishna Murthy Jatavallabhula, Gengshan Yang, Sebastian Scherer, Deva Ramanan, and Jonathon Luiten. Splatam: Splat track & map 3d gaussians for dense rgb-d slam. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21357–21366, 2024. 2
- [6] Hidenobu Matsuki, Riku Murai, Paul HJ Kelly, and Andrew J Davison. Gaussian splatting slam. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18039–18048, 2024. 2
- [7] Radu Bogdan Rusu, Nico Blodow, and Michael Beetz. Fast point feature histograms (fpfh) for 3d registration. In *2009 IEEE international conference on robotics and automation*, pages 3212–3217. IEEE, 2009. 1
- [8] Erik Sandström, Yue Li, Luc Van Gool, and Martin R Oswald. Point-slam: Dense neural point cloud-based slam. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 18433–18444, 2023. 2
- [9] Zachary Teed and Jia Deng. DROID-SLAM: Deep Visual SLAM for Monocular, Stereo, and RGB-D Cameras. *Advances in neural information processing systems*, 2021. 2
- [10] Chen Wenbo and Liu Ligang. Deblur-gs: 3d gaussian splatting from camera motion blurred images. *Proc. ACM Comput. Graph. Interact. Tech. (Proceedings of I3D 2024)*, 7(1), 2024. 2
- [11] Thomas Whelan, Renato F Salas-Moreno, Ben Glocker, Andrew J Davison, and Stefan Leutenegger. Elasticfusion: Real-time dense slam and light source estimation. *The International Journal of Robotics Research*, 35(14):1697–1716, 2016. 1
- [12] Lingzhe Zhao, Peng Wang, and Peidong Liu. BAD-Gaussians: Bundle Adjusted Deblur Gaussian Splatting. 2024. 2
- [13] Zihan Zhu, Songyou Peng, Viktor Larsson, Weiwei Xu, Hujun Bao, Zhaopeng Cui, Martin R Oswald, and Marc Pollefeys. Nice-slam: Neural implicit scalable encoding for slam. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12786–12796, 2022. 2



Figure 7. Comparison of deblurring results using NAFNet [2] and our method: The performance of pretrained networks is highly dependent on the training dataset. As a result, NAFNet struggles to restore high-frequency details when the input data is extremely blurry.

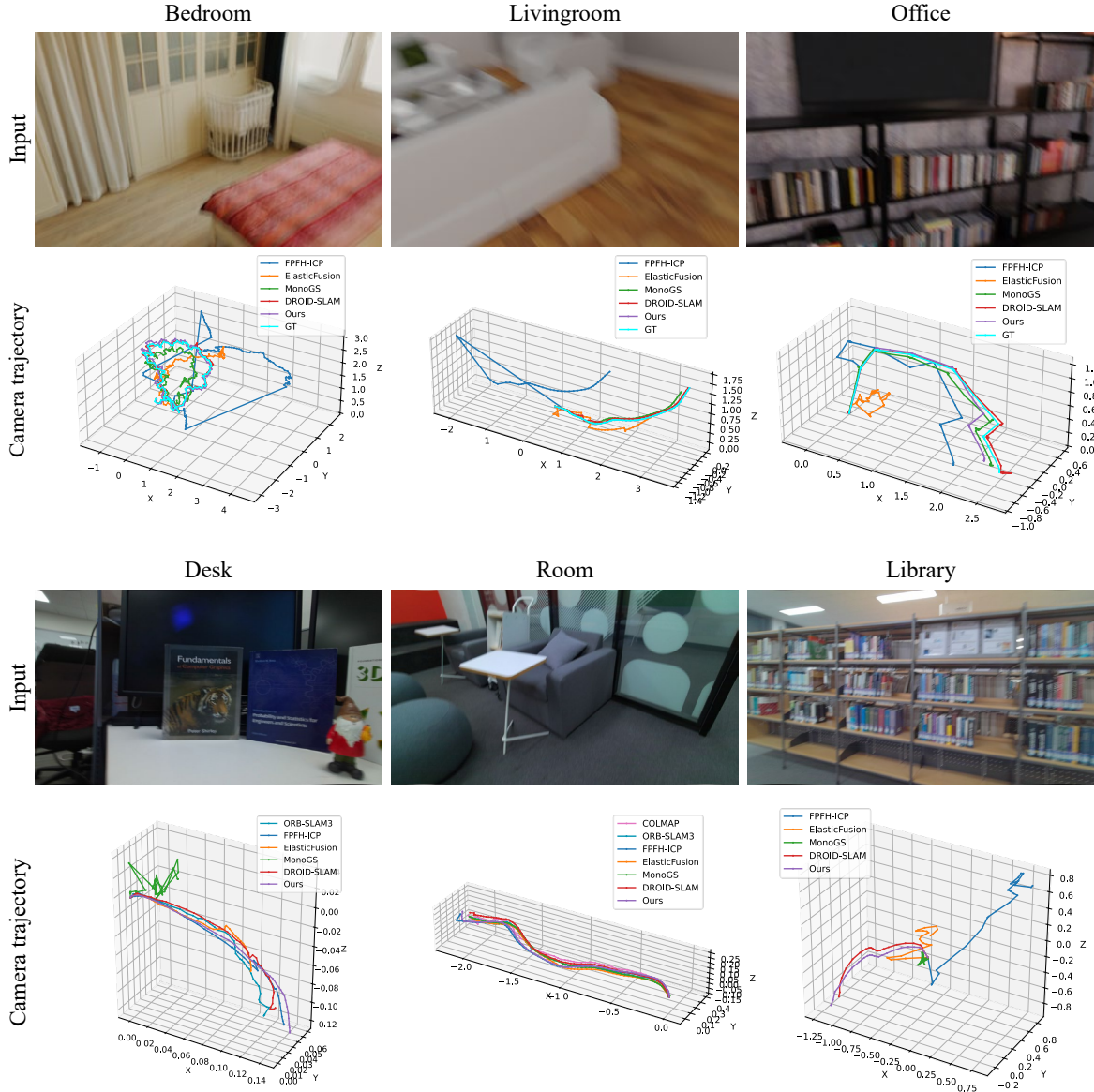


Figure 8. Visualization of estimated camera trajectories: The top row displays trajectories for synthetic scenes, while the bottom row shows trajectories for real-world scenes, as estimated by each method.



Figure 9. Our refinement step (Section 3.2 in the main paper) effectively corrects misalignments, ensuring accurate alignment while preserving dense point clouds.



Figure 10. Reconstructed point clouds for synthetic scenes: Our method demonstrates superior performance by producing dense and well-aligned point clouds compared to other methods.

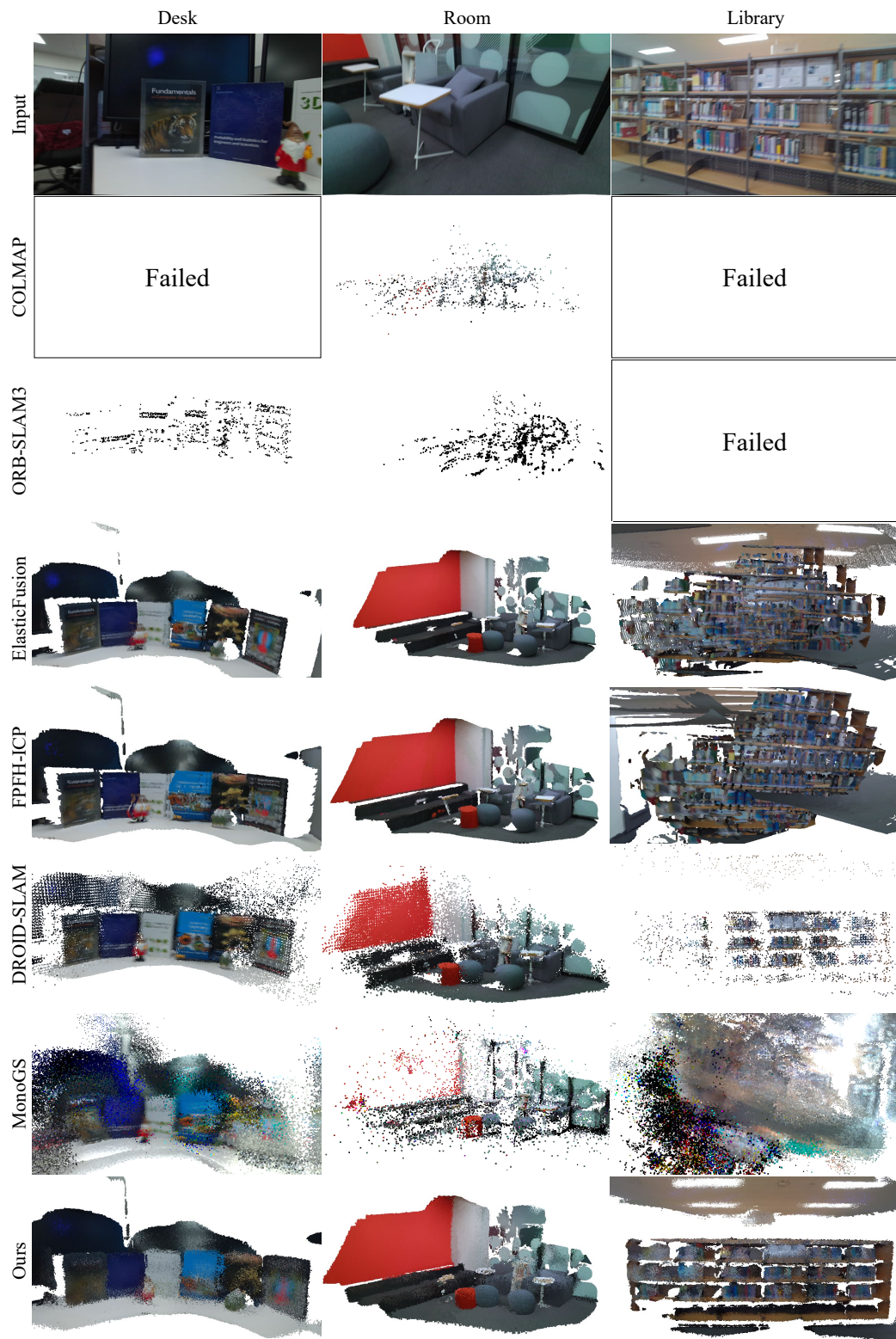


Figure 11. We show reconstructed point cloud from each method for real scenes. Our method provides dense and well-aligned point clouds.