Supplementary Material for EGG-Fusion

XIAOKUN PAN, State Key Lab of CAD&CG, Zhejiang University, China
ZHENZHE LI, State Key Lab of CAD&CG, Zhejiang University, China
ZHICHAO YE*, SenseTime Research, China
HONGJIA ZHAI, State Key Lab of CAD&CG, Zhejiang University, China
GUOFENG ZHANG*, State Key Lab of CAD&CG, Zhejiang University, China

A.1 Implementation Details

Our system mainly consists of two modules: the tracking module and the dense mapping module. The tracking module is responsible for preprocessing the input RGB-D frame and performing tracking optimization, including pose estimation based on sparse-correspondences and dense alignment. The dense mapping module handles the initialization of Gaussian surfels, surfel fusion based on information filtering, and end-to-end optimization of geometry and appearance through rasterization.

To achieve overall system efficiency, we employ different implementation strategies tailored to each module. In the data preprocessing stage, we utilize CUDA to efficiently process the input RGB-D data stream, including filtering and the computation of vertex and normal maps. For pose initialization based on sparse correspondences, we adopt the frontend module from ORB-SLAM2 [Mur-Artal and Tardós 2017], which leverages ORB [Rublee et al. 2011] features to perform both 2D-2D and 2D-3D matching. This component is implemented in C++ and invoked from the main program via a Python interface. After pose initialization, dense alignment is further applied for pose refinement. This process employs PyTorch-based tensor computations to leverage GPU acceleration for per-pixel matching, local linearization, and reduction. For the differentiable optimization of Gaussian surfels, we build upon the CUDA implementations of [Dai et al. 2024; Kerbl et al. 2023], within which we integrate the functionality of surfel fusion based on information filtering. The main structure of the program is implemented in Python to orchestrate and connect the different modules.

All experiments were conducted on a machine equipped with an RTX 4090 GPU with 24GB of memory and an Intel i9-14900KF CPU with 32 threads.

A.2 Camera Pose Estimation

In the dense alignment stage, we employ PyTorch-based tensor computation to leverage GPU acceleration for per-pixel matching, local linearization, and reduction. To solve Eq.(15), we adopt a coarse-to-fine pyramid strategy. Specifically, both the rendered global model surface and the current frame image are downsampled into a multi-scale image pyramid with L_{pyr} levels, and each level is optimized for N_{pyr} iterations. Starting from the coarsest level, we perform dense alignment using a least-squares method and progressively refine the solution to the original image resolution.

We compute the Jacobian matrix $\mathbf{J}$ for both the ICP and photometric residuals, and at each iteration, the update step is computed as:

$$\delta\boldsymbol{\xi}^{(n)} = -(\mathbf{J}^\top\mathbf{J} + \lambda\mathbf{I})^{-1}\mathbf{J}^\top r(\boldsymbol{\xi}^n). \tag{1}$$

The current estimate is then updated by:

$$\boldsymbol{\xi}^{n+1} \longleftarrow \boldsymbol{\xi}^n \circ \delta\boldsymbol{\xi}^{(n)}. \tag{2}$$

The optimization terminates once the total number of iterations reaches $N_{\text{pyr}} \cdot L_{\text{pyr}}$. In our default setting, we use $N_{\text{pyr}} = 2$ and $L_{\text{pyr}} = 3$.

A.3 KeyFrame Selection

During the tracking process, we determine keyframes to serve as target images for both local and global optimization of the Gaussian surfels map. They also act as optimization targets in the sliding window optimization. The first input frame is set as a keyframe, and subsequent frames are determined as keyframes based on whether the translation t or rotation angle θ relative to the previous keyframe exceeds a predefined threshold t_k and θ_k. The default setting is $t_k = 0.3m$ and $\theta_k = 20°$.

A.4 Surfels Selection for Fusion

To determine the set of visible surfels within the current camera frustum, we define *visibility* in a geometric sense, without considering occlusion. Two criteria are used: 1) The projection of the surfel S_i onto the image plane falls within the valid image region. 2) The normal of the surfel is oriented towards the camera. Formally, the visible surfel set is defined as:

$$\mathcal{S}^{\text{vis}} = \left\{ S_i \in \mathcal{S} \;\middle|\; \Pi(\mathbf{p}_i') \in \Omega \wedge \mathbf{n}_i \cdot \mathbf{R}_t^z < 0 \right\}, \tag{3}$$

Here, $\mathbf{p}_i'$ is the center of surfel S_i in the current frame's coordinate system. $\Omega = \{(u, v) \in \mathbb{R}^2 \mid 0 \le u < W, ; 0 \le v < H\}$ denotes the image coordinate domain, and $\mathbf{R}_t^z$ refers to the z-axis of the rotation component of the current camera pose.For each $S_i \in \mathcal{S}^{\text{vis}}$, we define the set of surface surfels from the current viewpoint using the following criterion:

$$\mathcal{S}^{\text{surf}} = \left\{ S_i \in \mathcal{S}^{\text{vis}} \;\middle|\; \left| \left[\mathbf{p}_i'\right]_z - \bar{D}_t(\mathbf{u}_i) \right| < \delta_s \right\}, \tag{4}$$

where $\bar{D}_t$ is the rendered depth from the current viewpoint, and δ_s denotes the surface thickness threshold, $\mathbf{p}_i^{'}$ and $\mathbf{u}_i$ have been define in Sec. 3.2.2. Then we check the $\bar{D}_t$ with depth value under current view to identify whether it is re-measured.

A.5 Detail of Rotation Matrix Update

Regarding the details of Eq.(8), we omit the explicit form of the rotation matrix conversion from a rotation vector, denoted as $\Delta\mathbf{R}(\mathbf{n}, \theta)$.

*Guofeng Zhang and Zhichao Ye are Corresponding Authors

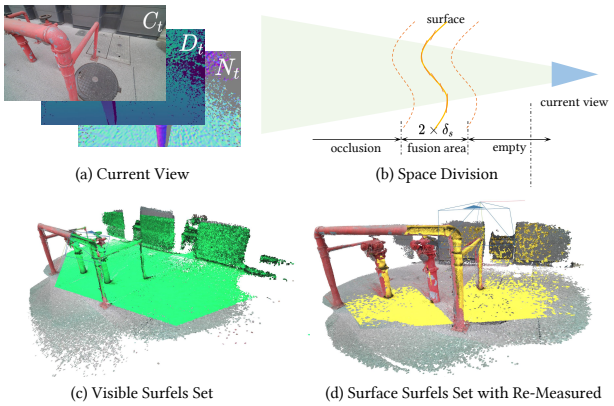

Fig. 1. Surfels Selection for Fusion

In our implementation, we adopt the cross-product form of the Rodrigues' rotation formula:

$$\Delta\mathbf{R}(\mathbf{n},\theta) = \cos\theta \cdot \mathbf{I} + (1-\cos\theta)\cdot \mathbf{n}\mathbf{n}^\top + \sin\theta \cdot [\mathbf{n}]_\times \tag{5}$$

Here, $[\mathbf{n}]_\times$ denotes the skew-symmetric matrix of a vector $\mathbf{n}$:

$$[\mathbf{n}]_\times = \begin{bmatrix} 0 & -n_z & n_y \\ n_z & 0 & -n_x \\ -n_y & n_x & 0 \end{bmatrix} \tag{6}$$

A.6 Running Time on Replica

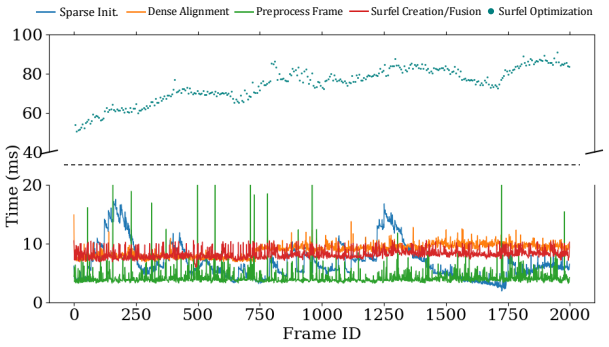

Fig. 2. Runtime of our system. We plotted the time consumption of the main modules of the system on Replica `office0`.

We report the time consumption for each frame in the major modules on the `office0`, as shown in the Fig. 2. We provide statistics for the camera tracking, which includes sparse-correspondence-based pose initialization and dense alignment. Each frame undergoes preprocessing to prepare data for the mapping stage. In the mapping stage, each frame contributes to adding new surfels to the global map and fusing them with existing surfels. Finally, we optimize the Gaussian surfels using frame batches at a certain frequency. This part is comparatively more time-consuming than the other modules but the is still significantly less than that of other methods.

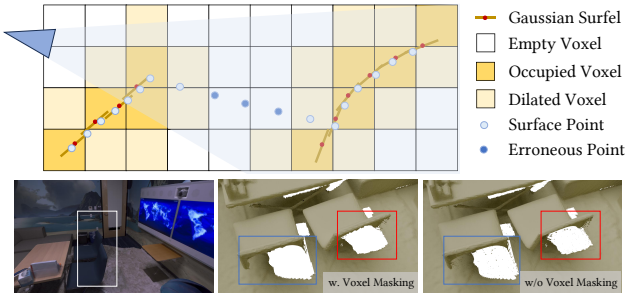

Fig. 3. Meshing the Gaussian Surfel based Scene Map with Voxel Masking

A.7 Meshing with Voxel Masking

As emphasized in [Dai et al. 2024], the alpha decay property based on surfel ellipsoid centers leads to erroneous depth estimates in regions with depth discontinuities. Consequently, when meshing the Gaussian surfel based map, the edges often contain noisy or scattered triangles. [Dai et al. 2024] addresses this issue using a volumetric cutting strategy to prevent such artifacts during screened Poisson surface reconstruction. Similarly, when we follow the TSDF-based meshing approaches in [Yang et al. 2022; Zhu et al. 2022], we encounter the same problem. Therefore, we adopt a voxel masking strategy suitable for voxel-based surface extraction.

As illustrated in Fig. 3, we construct an occupancy grid of the scene based on the surfel centers, and then determine whether points from the rendered depth map should contribute to TSDF integration by checking if they fall within the occupied voxels. This is based on the fact that the scene surfaces generally lie within the spatial vicinity of the surfels.

To mitigate the quantization errors introduced by voxelization where points may be near surfels but still fall into empty voxels, we further apply voxel dilation. This ensures more accurate selection of surface points for TSDF-based meshing.

A.8 Rendering Results on Replica

The quality of view synthesis under training views across 8 scenes in the Replica dataset is shown in Tab. 1. Among NeRF-based and GS-based methods, our method achieved the best average rendering quality and the best rendering quality in most scenes. We believe this is due to the Gaussian surfel tightly fitting the scene surface with the geometric regularization while end-to-end optimization.

A.9 Detail of Reconstruction Results on ScanNet++

Due to space constraints in the article, we only presented the average rendering metrics on ScanNet++ [Yeshwanth et al. 2023]. As shown in Tab. 2, we provide the complete results for a more comprehensive analysis.

A.10 More Results

We present the complete mesh reconstruction results on Replica [Straub et al. 2019] and ScanNet++ [Yeshwanth et al. 2023] to demonstrate the superiority of our method in both reconstruction detail and overall accuracy, as shown as in Fig. 5 and Fig. 4.

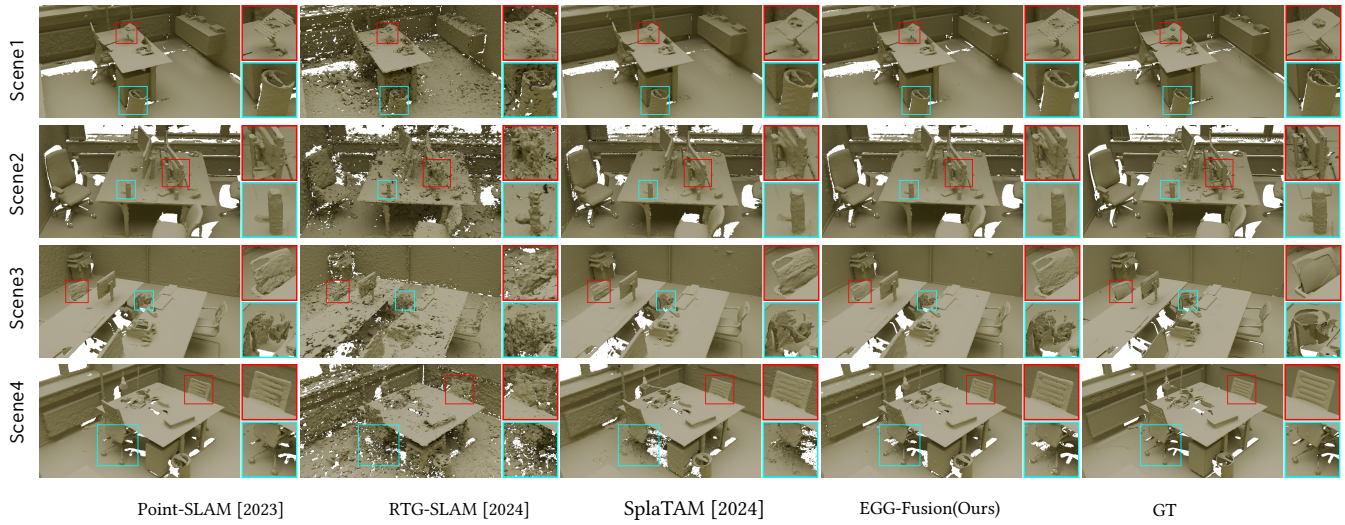

Fig. 4. Reconstruction results on ScanNet++ dataset.

Table 1. Comparison of train view synthesis on Replica. * indicates that the result is taken from [Zhu et al. 2023]

Method	Metric	Rm 0	Rm 1	Rm 2	Off 0	Off 1	Off 2	Off 3	Off 4	Avg.
NICE-SLAM* [2022]	PSNR↑	22.12	22.47	24.52	29.07	30.34	19.66	22.23	24.94	24.42
	SSIM↑	0.689	0.757	0.814	0.874	0.886	0.797	0.801	0.856	0.809
	LPIPS↓	0.330	0.271	0.208	0.229	0.181	0.235	0.209	0.198	0.233
Vox-Fusion [2022]	PSNR↑	26.16	28.16	28.03	32.49	32.45	26.86	27.27	29.61	28.88
	SSIM↑	0.898	0.904	0.918	0.942	0.952	0.934	0.941	0.946	0.929
	LPIPS↓	0.293	0.271	0.232	0.216	0.207	0.227	0.187	0.199	0.229
Point-SLAM [2023]	PSNR↑	32.40	34.08	35.50	38.26	39.16	33.99	33.48	33.49	35.17
	SSIM↑	0.974	0.977	0.982	0.983	0.986	0.960	0.960	0.979	0.975
	LPIPS↓	0.113	0.116	0.111	0.100	0.118	0.156	0.132	0.142	0.124
SplaTAM [2024]	PSNR↑	32.09	33.62	35.00	38.16	39.04	31.88	30.14	31.69	33.95
	SSIM↑	0.972	0.970	0.982	0.982	0.982	0.965	0.950	0.947	0.969
	LPIPS↓	0.077	0.097	0.074	0.086	0.092	0.099	0.118	0.155	0.100
RTG-SLAM [2024]	PSNR↑	30.91	33.41	34.49	39.02	39.24	32.78	32.73	35.56	34.76
	SSIM↑	0.962	0.976	0.981	0.989	0.989	0.980	0.981	0.984	0.980
	LPIPS↓	0.147	0.116	0.120	0.082	0.099	0.144	0.138	0.123	0.121
Ours	PSNR↑	31.21	34.14	34.94	39.75	39.69	32.98	33.14	35.95	35.23
	SSIM↑	0.966	0.979	0.983	0.990	0.991	0.983	0.980	0.988	0.983
	LPIPS↓	0.131	0.089	0.098	0.058	0.065	0.111	0.106	0.108	0.096

References

Pinxuan Dai, Jiamin Xu, Wenxiang Xie, Xinguo Liu, Huamin Wang, and Weiwei Xu. 2024. High-Quality Surface Reconstruction Using Gaussian Surfels. arXiv:2404.17774

Nikhil Keetha, Jay Karhade, Krishna Murthy Jatavallabhula, Gengshan Yang, Sebastian Scherer, Deva Ramanan, and Jonathon Luiten. 2024. SplaTAM: Splat, Track & Map 3D Gaussians for Dense RGB-D SLAM. arXiv:2312.02126 [cs]

Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 2023. 3D Gaussian Splatting for Real-Time Radiance Field Rendering. arXiv:2308.04079

Raul Mur-Artal and Juan D Tardós. 2017. Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE transactions on robotics* 33, 5 (2017), 1255–1262.

Zhexi Peng, Tianjia Shao, Yong Liu, Jingke Zhou, Yin Yang, Jingdong Wang, and Kun Zhou. 2024. RTG-SLAM: Real-time 3d reconstruction at scale using gaussian splatting. In *ACM SIGGRAPH 2024 Conference Papers*. 1–11.

Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. 2011. ORB: An efficient alternative to SIFT or SURF. In *2011 International conference on computer vision*. Ieee, 2564–2571.

Erik Sandström, Yue Li, Luc Van Gool, and Martin R. Oswald. 2023. Point-SLAM: Dense Neural Point Cloud-based SLAM. arXiv:2304.04278 [cs]

Julian Straub, Thomas Whelan, Lingni Ma, Yufan Chen, Erik Wijmans, Simon Green, Jakob J. Engel, Raul Mur-Artal, Carl Ren, Shobhit Verma, Anton Clarkson, Mingfei Yan, Brian Budge, Yajie Yan, Xiaqing Pan, June Yon, Yuyang Zou, Kimberly Leon, Nigel Carter, Jesus Briales, Tyler Gillingham, Elias Mueggler, Luis Pesqueira, Manolis Savva, Dhruv Batra, Hauke M. Strasdat, Renzo De Nardi, Michael Goesele, Steven Lovegrove, and Richard Newcombe. 2019. The Replica Dataset: A Digital Replica of Indoor Spaces. arXiv:1906.05797

Xingrui Yang, Hai Li, Hongjia Zhai, Yuhang Ming, Yuqian Liu, and Guofeng Zhang. 2022. Vox-Fusion: Dense Tracking and Mapping with Voxel-based Neural Implicit Representation. In *2022 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*. 499–507. arXiv:2210.15858 [cs] doi:10.1109/ISMAR55827.2022.00066

Chandan Yeshwanth, Yueh-Cheng Liu, Matthias Nießner, and Angela Dai. 2023. Scannet++: A high-fidelity dataset of 3d indoor scenes. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 12–22.

Zihan Zhu, Songyou Peng, Viktor Larsson, Zhaopeng Cui, Martin R. Oswald, Andreas Geiger, and Marc Pollefeys. 2023. NICER-SLAM: Neural Implicit Scene Encoding for RGB SLAM. arXiv:2302.03594

Zihan Zhu, Songyou Peng, Viktor Larsson, Weiwei Xu, Hujun Bao, Zhaopeng Cui, Martin R. Oswald, and Marc Pollefeys. 2022. NICE-SLAM: Neural Implicit Scalable Encoding for SLAM. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, New Orleans, LA, USA, 12776–12786. doi:10.1109/CVPR52688.2022.01245

Table 2. Comparison of rendering capabilities on ScanNet++, including both training views and test views (views not seen during training).

Methods	Metrics	Novel View					Training View				
		Avg.	S1	S2	S3	S4	Avg.	S1	S2	S3	S4
Point-SLAM [2023]	PSNR ↑	17.68	15.00	21.63	15.87	18.21	24.35	24.71	23.13	24.77	24.79
	SSIM ↑	0.623	0.611	0.702	0.560	0.618	0.800	0.805	0.783	0.813	0.800
	LPIPS ↓	0.548	0.558	0.480	0.614	0.539	0.373	0.367	0.383	0.375	0.366
SplaTAM [2024]	PSNR ↑	24.75	24.08	26.41	25.19	23.33	27.30	27.82	25.42	28.22	27.75
	SSIM ↑	0.900	0.886	0.930	0.888	0.897	0.940	0.946	0.924	0.943	0.947
	LPIPS ↓	0.208	0.211	0.175	0.253	0.195	0.130	0.119	0.158	0.130	0.112
RTG-SLAM [2024]	PSNR ↑	24.77	24.27	25.44	26.09	23.28	27.54	28.22	24.69	29.29	27.96
	SSIM ↑	0.882	0.876	0.886	0.883	0.882	0.925	0.936	0.889	0.937	0.936
	LPIPS ↓	0.255	0.249	0.261	0.285	0.225	0.184	0.165	0.238	0.176	0.155
Ours	PSNR ↑	25.70	25.50	26.55	26.72	23.96	29.06	29.97	26.08	30.59	29.45
	SSIM ↑	0.907	0.906	0.901	0.900	0.903	0.944	0.953	0.922	0.950	0.943
	LPIPS ↓	0.212	0.196	0.231	0.250	0.190	0.141	0.121	0.178	0.145	0.116

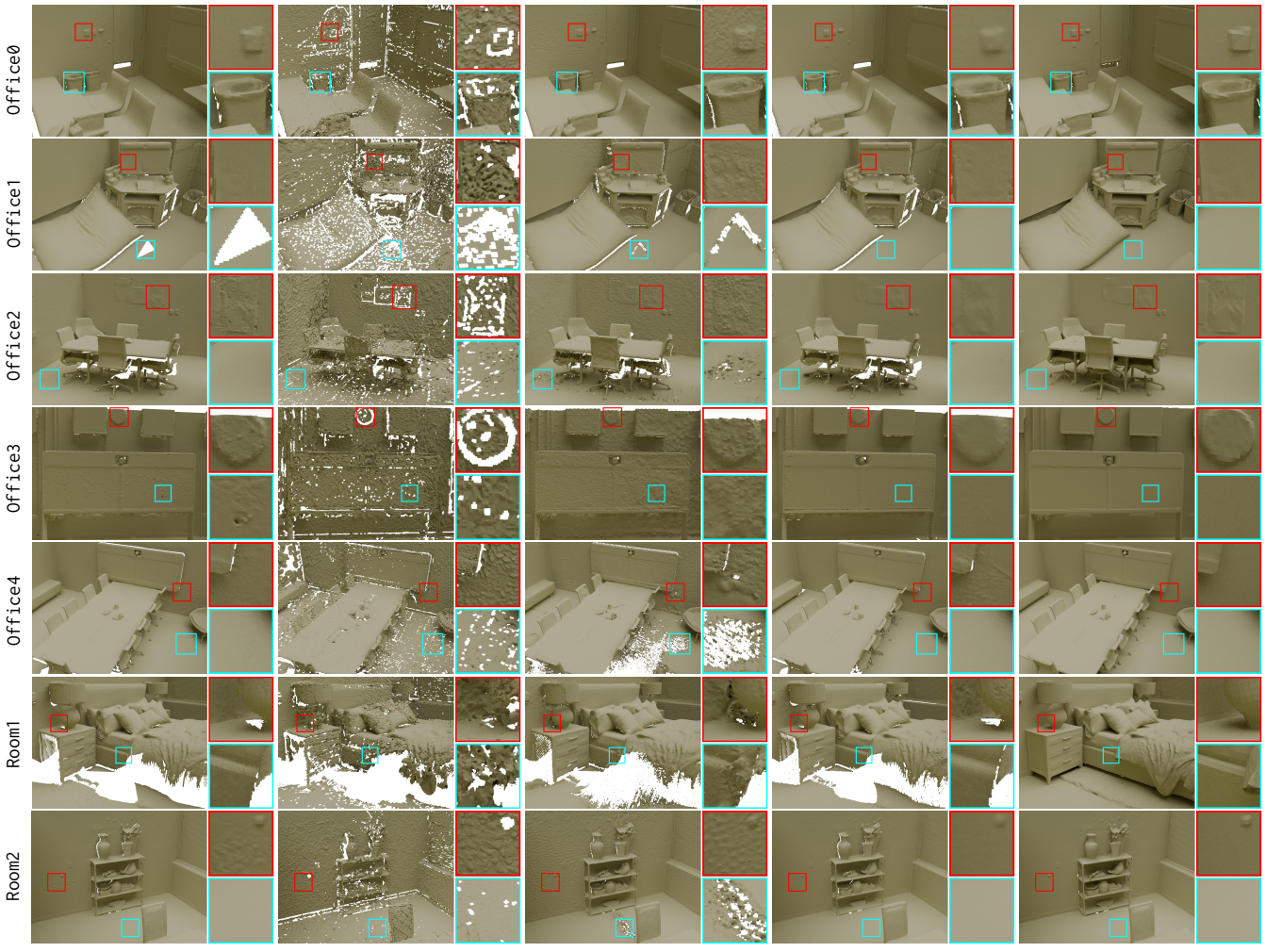

Fig. 5. Reconstruction results on Replica dataset.