LTGS: Long-Term Gaussian Scene Chronology From Sparse View Updates

Supplementary Material

A. Implementation details

A.1. Change detection

We detect fine object-level changes in 2D by using the semantic prior of the SAM model [5]. We prepare a pair of images $(I_t^i, \hat{I}_t^i)$ from captured and rendered images from the corresponding viewpoints. To find abstract differences between the rendered and captured images, we extract features from the segmentation model for both images. We used features obtained from the pretrained SAM encoder and interpolated them to match the original image resolution, after which we computed pairwise cosine similarities for comparison. We additionally use SSIM to capture structural differences, where using only semantic cosine similarity struggles to find slightly deviated objects.

To obtain coarse change masks, we need a binarization by thresholding the obtained differences. Since the binary coarse mask is highly sensitive to manually defined thresholds, which vary significantly across scenes, we adopt Otsu's method [12] to automatically determine the threshold τ_{cos}. We obtain the coarse binary masks $\mathcal{M}_{t,\text{coarse}}^i$ as follows:

$$\begin{aligned} M_{t,\text{coarse}}^i = \gamma \cdot \cos(\mathcal{E}(I_t^i), \mathcal{E}(\hat{I}_t^i)) \\ + (1-\gamma) \cdot \text{SSIM}(I_t^i, \hat{I}_t^i) \leq \tau_{\cos}, \end{aligned} \tag{1}$$

where $\mathcal{E}$ denotes the feature extractor of SAM. $\gamma = 0.7$ was used throughout our experiments.

After obtaining coarse change regions, we extract fine-grained object-level change masks to model instance-wise changes. Since precise object-level changes are difficult to capture using the coarse stages described above, we leverage the automatic mask generation from SAM [5] within the coarse binary mask $M_{t,\text{coarse}}^i$. For each generated mask, we first calculate the intersection-over-union (IoU) between its region and the coarse binary mask. We further compare the cosine similarity between the accumulated features within those regions in the extracted features of the rendered and captured images. The fine object-level masks can be obtained as follows:

$$\begin{aligned} \mathcal{O}_t^i = \bigcup_k \Big\{ o_{t,k}^i \Big| \text{IoU}(o_{t,k}^i, M_{t,\text{coarse}}^i) \geq \tau_{\text{IoU}} \wedge \\ \cos\big(\Phi(o_{t,k}^i, \mathcal{E}(I_t^i)), \Phi(o_{t,k}^i, \mathcal{E}(\hat{I}_t^i))\big) \leq \tau_{\cos} \Big\}, \end{aligned} \tag{2}$$

where $o_{t,k}^i$ denotes the kth object mask generated by automatic mask generation, and $\Phi(m, X) = \frac{1}{|m|} \sum_{p \in \Omega} m(p)\, X(p)$ denotes the average pooling operator of feature X within the interior Ω of mask m. We select and keep only the highly overlapped and semantically different masks following [4]. We further removed masks occupying only small regions to prevent noise, and we dilated the change masks to address pixel-wise errors.

A.2. Object tracking

We provide additional details of our pipeline to associate with changed instances. Given 2D object-level change masks from Sec. 3.2 of the main paper, we leverage both visual feature and SAM feature matching to associate 2D change masks. As mentioned in Sec. 3.3 of the main paper, we separate the instance matching into intra-timestep matching and cross-timestep matching. Let $\mathcal{O}_t^i = \{o_{t,1}^i, o_{t,2}^i, \ldots, o_{t,N_o}^i\}$ be the set of object-level masks of ith viewpoint at timestep t. We first compute pairwise matches across images I_t^i and I_t^j using MASt3R within the same timestep. Given the extracted MASt3R descriptors $\mathbf{d}_{t,k}^i$ and $\mathbf{d}_{t,l}^j$ for object $o_{t,k}^i$ and $o_{t,l}^j$ respectively, we find descriptor matches $\mathcal{M}_t^{(i,k)\leftrightarrow(j,l)}$ within object change masks as follows:

$$\begin{aligned} \mathcal{M}_t^{(i,k)\leftrightarrow(j,l)} = \text{match}(\mathbf{d}_{t,k}^i, \mathbf{d}_{t,l}^j), \\ o_{t,k}^i \in \mathcal{O}_t^i,\, o_{t,l}^j \in \mathcal{O}_t^j,\; i \neq j. \end{aligned} \tag{3}$$

Note for matching, we follow the fast reciprocal matching procedure from MASt3R [6]. Based on these matches, we construct a graph $G_t = (N_t, E_t)$ as:

$$\begin{aligned} N_t = \bigcup_i \mathcal{O}_t^i, \\ E_t = \big\{ \big(o_{t,k}^i,\, o_{t,l}^j, |\mathcal{M}_t^{(i,k)\leftrightarrow(j,l)}|\big) \;\big|\; i \neq j \big\}. \end{aligned} \tag{4}$$

where the nodes N_t are objects and the edge weight E_t encodes the total number of matches between every pair of objects $o_{t,k}^i$ and $o_{t,l}^j$. We then cluster the graph using the depth-first search (DFS) algorithm [16] to identify connected components, where each component corresponds to a unique global object identity. Here, we retain an edge only if the number of matches exceeds a threshold τ_{match}. Based on these clustering results, we assign instance IDs to the matched components and filter out unmatched objects for consistency. This filtering strategy gives robustness to detected instances that are inaccurate due to the artifacts in rendered images.

After obtaining object-level matches for every sequence within an identical timestep, it is essential to associate object masks across different timesteps. For each object $o_{t,k}^i$

and $o^j_{\tilde{t},l}$ where t and $\tilde{t}$ are the set of target times after intra-timestep matching, we accumulate the SAM [5] features in the object region, and build a matrix $\mathbf{S}_{k\leftrightarrow l}$ that contains cosine similarity among every pair as follows:

$$\mathbf{S}_{k\leftrightarrow l} = \cos\big(\Phi(o^i_{t,k}, \mathcal{E}(I^i_t)), \Phi(o^j_{\tilde{t},l}, \mathcal{E}(I^j_{\tilde{t}}))\big). \quad (5)$$

Based on the matrix, we leverage Hungarian matching [11] to solve an optimal assignment problem between the instances as $\pi^* = \arg\max_\pi \sum_k \mathbf{S}_{k\leftrightarrow\pi(k)}$, where $\pi(k)$ denotes the matched object in timestep $\tilde{t}$. After semantic matching, we filter out false pairs for those with the cosine similarities lower than τ_{cos} defined in Eq. (1) and Eq. (2). Note that we conduct this process identically for every possible timestep pair for $t \in [0, T]$.

A.3. Object Gaussian Template reconstruction

Given the tracked object masks $M = \{M^i_t | i = 1, ..., N_v; t = 0, ..., T\}$, we provide additional details of the construction and initialization of object-level Gaussian Splats. Here, we define the total number of instances at the initial timestep as E. For the objects that emerge in initial reconstruction, we separate those using the optimal label assignment problem introduced in FlashSplat [15]. For our task, the problem is defined as follows:

$$\min_{\{P_k\}} \mathcal{F} = \sum_i \left| \sum_k P_k \alpha_k T_k - M^i_0 \right|, \quad P_k \in \{0, 1, ..., E\}, \quad (6)$$

where α_k, T_k each denotes the alpha value and transmittance during volume rendering, and P_k denotes the per-Gaussian 3D label. Among the P_k, index 0 corresponds to the background, while the remaining indices correspond to the foreground. The above equation solves the problem of assigning the 3D label P_k by volume rendering them to the image domain to match the given multiview masks at $t = 0$. Specifically, we use the majority voting algorithm as follows:

$$\begin{aligned} P_k &= \arg\max_{n\in\{0,m\}} A_n, \\ A_m &= \sum_i \alpha_k T_k \mathbb{1}(M^i_0, m), \\ A_0 &= \sum_i \sum_{e\neq m} \alpha_k T_k \mathbb{1}(M^i_0, e), \end{aligned} \quad (7)$$

where $\mathbb{1}(M^i_0, m)$ denotes the indicator function which is equal to 1 if the pixel in mask M^i_0 belongs to object m, and 0 otherwise. Eq. (7) solves the assignment problem by allocating the label that maximizes the weighted contribution of Gaussians within the object mask regions. Please refer to the original FlashSplat [15] paper regarding the details and the derivation. We additionally filter Gaussians that are out of the object mask region after directly projecting the centers to remove floating artifacts.

After registration and geometric verification as presented in Sec. 3.3 of the main paper, we initialize Gaussians for objects that do not exist in the initial reconstruction. We first extract point clouds for new objects from the global scene reconstruction of MASt3R [6] with estimated poses from the hierarchical localization pipeline [13]. In the original implementation of MASt3R, camera parameters were optimized jointly with per-view depth maps and global scales. We modify the optimization loop to operate only on depth maps with scale and offset parameters, while the camera poses remain fixed. To reduce noise, we retain only the point clouds with per-pixel confidence values greater than 1.5, and we randomly downsample the point cloud by a factor of 4, as per-pixel point clouds are overly dense, which is inefficient for optimization.

A.4. Hyperparameters

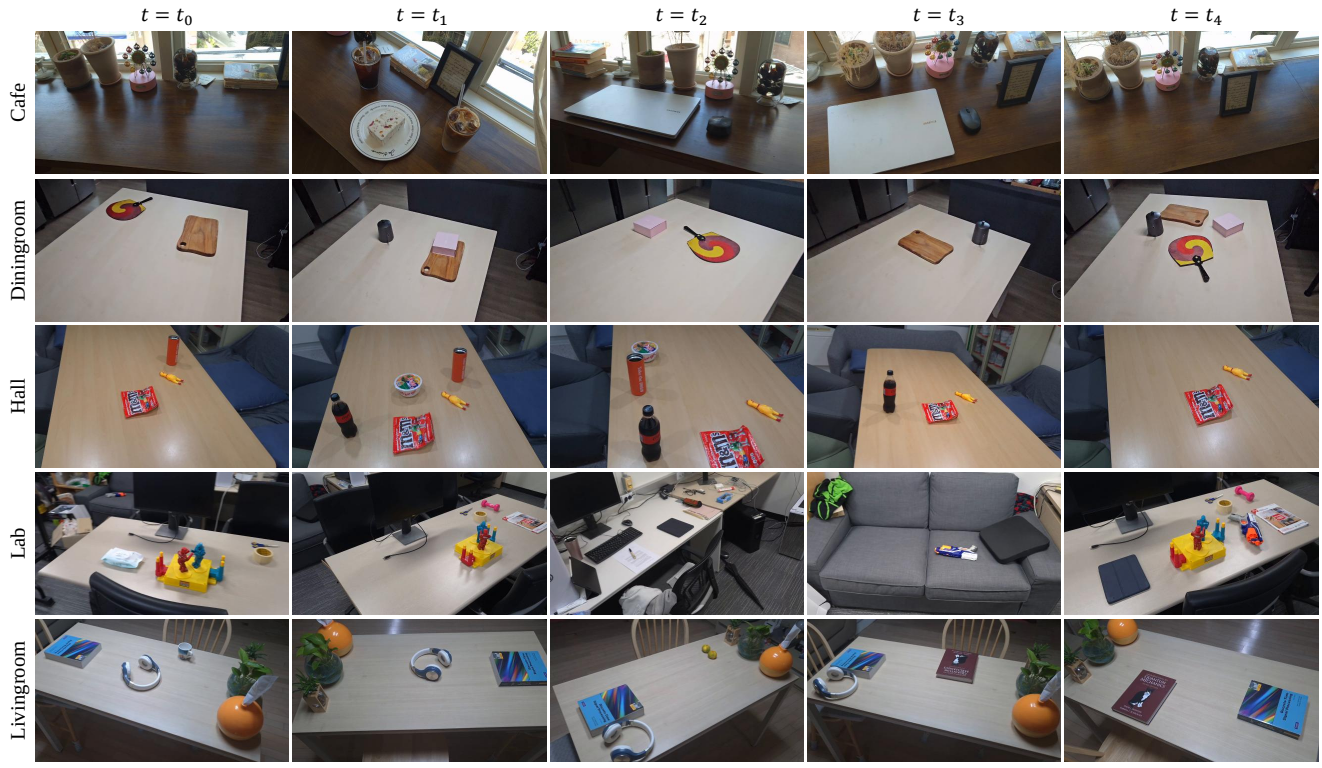
In this section, we discuss how key hyperparameters are set and evaluate their influence on performance. We analyze the effect of the τ_{cos} in Eq. (1) and Eq. (2), and the connectivity threshold τ_{match} in Sec. A.2. Additionally, we evaluate the geometric verification threshold in Sec. 3.3 of the main paper, where we compare the chamfer distance between the two Gaussian object templates to $\tau_{overlap}$ to examine the geometric consistency across the temporal track. We demonstrate the results of variations on above hyperparameters in Tab. 1.

For the change detection, we set $\tau_{cos} = 0.9$ in Eq. (1) and Eq. (2), which were used throughout our experiments. We sweep the values between [0.8, 0.95], referring to the cosine similarity threshold in [4], originally set to 0.88. We set $\tau_{match} = 50$, considering that higher threshold values tend to neglect small objects while tracking. In geometric verification stage, $\tau_{overlap} = 0.15$ worked well to distinguish the geometrically matched objects. Severely lower $\tau_{overlap}$ interrupts the object from being rigidly tracked.

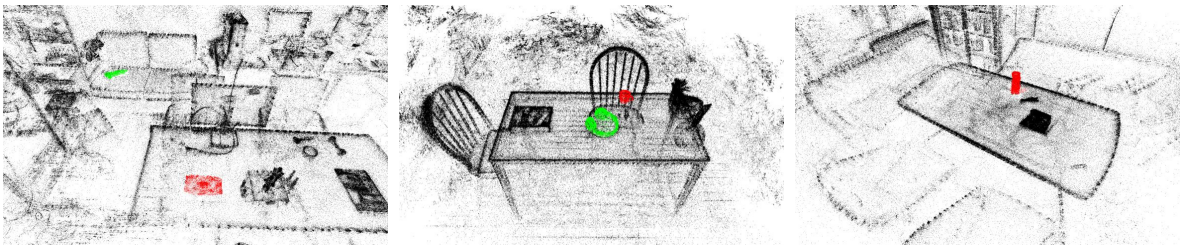
To initialize Gaussian primitives, we use the dense point cloud's position and its corresponding color as the initial position and color. We convert the RGB color into spherical harmonics (SH) coefficients and initialize higher-order SH components with zeros. We use low initial opacity values as $\alpha = 0.1$ for all Gaussians. Rotations are initialized as

τ_{cos}	PSNR	τ_{match}	PSNR	$\tau_{overlap}$	PSNR
0.80	22.79	10	23.40	0.05	23.29
0.85	22.76	50	23.43	0.1	23.39
0.90	23.43	100	23.33	0.15	23.43
0.95	23.33	200	23.22	0.20	23.41

Table 1. **Effects of the hyperparameter.**



(a) Examples of Capture



Lab Livingroom Hall

(b) Initial Reconstruction

Figure 1. **Examples of our datasets.** (a) Illustration of long-term captures of our dataset for each scene. (b) Initial reconstruction of Gaussian splats at $t = 0$ and tracked instances at the initial timestep.

identity quaternions, i.e., $q = (1, 0, 0, 0)$ for all Gaussians, while the scales are determined by the pairwise squared distance as done in the original 3DGS [3]. For refinement, we update the Gaussian Splats for 5000 iterations using the same learning rate as done in the official 3DGS [3] implementation. We skipped several techniques that were used in the original 3DGS such as opacity resetting, cloning and pruning operations to preserve the original reconstruction.

B. Datasets

In this section, we provide some additional details for the datasets that we have introduced in our main manuscript. We casually captured video sequences using Galaxy S24 with fixed focal lengths and manually changed several objects between the captures for 5 sequences. After converting video frames to images, we regularly sampled 300-400 images for initial reconstruction to cover the scene of interest, and obtain camera poses using COLMAP SfM [14]. Fig. 1 illustrates the example of our datasets for every scene, captured from different timesteps. In total, our dataset contains five scenes that cover a diverse set of indoor spaces (Cafe, Diningroom, Livingroom, Hall, Lab). We additionally visualized the initial reconstruction of sampled scenes, with tracked instances using our pipeline, which serve as the initial object Gaussian templates before refinement. For evaluation, we selected every 8th frame following the conventional evaluation protocol of neural rendering [9, 10].

C. Additional performance analysis

C.1. Lighting changes

To ensure robust change estimation, we use semantic differences and structural differences rather than directly comparing the RGB values. We validate our framework's robustness against varying illumination by applying different exposure, tone, and contrast curves to our dataset, as demonstrated in Fig. 2. To account for the exposure changes in the different input images, we used the exposure compensation provided in the official 3DGS [3] implementation. As a result, our framework successfully isolates object-level modifications without being degraded by global illumination shifts, maintaining high rendering fidelity.

C.2. Geometry estimation errors

The robustness experiments in the main paper primarily analyze descriptor noise, related to the matching error. To further simulate the actual estimation failures, we conduct an additional analysis by assuming errors in depth maps and camera poses during reconstruction, which closely mimics real-world failure scenarios, as depicted in Fig. 3. While moderate noise is refined during the final refinement step, MASt3R [6] point clouds with severe errors and unstable registration are naturally treated as separate instances and refined independently. Consequently, this isolation strategy prevents initial pose or depth inaccuracies from corrupting the global scene geometry, effectively absorbing the errors to yield coherent structural reconstructions.

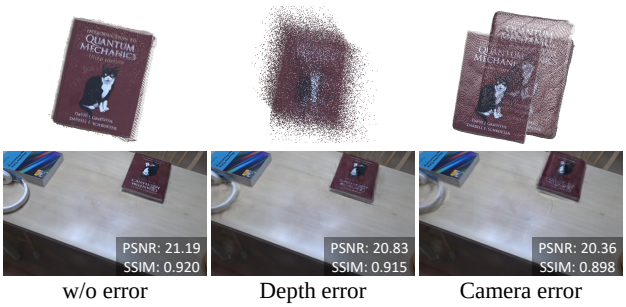
C.3. Lightweight representation capability

In our main experiments, we utilize 3 images per timestep across 4–5 timesteps per scene. To further evaluate the impact of input sparsity, we conduct an additional performance analysis by varying the number of images per timestep on the LIVINGROOM scene, as detailed in Tab. 2. As a reference baseline, we measured the performance of standard 3DGS [3] using 100 densely sampled images per timestep. Notably, our approach achieves comparable reconstruction quality using only sparse updates, eliminating the need for dense image captures.

Although we incorporate pretrained modules, our pipeline remains lightweight by leveraging reusable priors from the existing scene. As demonstrated in Tab. 3, our framework effectively models dynamic scenes while maintaining real-time rendering performance after optimization. Furthermore, it significantly reduces the memory footprint required for long-term environment maintenance, keeping memory usage comparable to the initial static 3DGS [3]. Ultimately, these results demonstrate that our updating mechanism not only renders photorealistic images but also provides a highly scalable and memory-efficient solution for long-term scene representations.

Pre-change render | Post-change input | Novel-view render

Figure 2. **Different lighting results with estimated change.**



w/o error | Depth error | Camera error

Figure 3. **Geometry estimation errors and rendering results.**

D. Additional comparative studies

D.1. Additional qualitative comparisons

We present additional qualitative results for both CL-NeRF [18] and our datasets in Fig. 4. We illustrate scenes that are not covered in Fig. 3 of the main manuscript. By incorporating reusable priors into scene representations, we achieve a notable reduction of artifacts, particularly in under-constrained regions where other baselines often struggle. Compared to the baselines, our method produces cleaner geometry and more photorealistic synthesis under novel viewpoints. This improvement highlights the effectiveness of our method in reducing ambiguity and enabling stable reconstructions across diverse scenes.

D.2. Per-scene quantitative comparisons

We provide the evaluation results of all scenes in terms of PSNR, SSIM, and LPIPS. As demonstrated in Tab. 4, Tab. 5 and Tab. 6, our framework achieves the best results for most scenes.

# image	2	3	4	5	6	3DGS [3] (∼100)
PSNR	23.06	23.46	23.92	24.21	24.41	25.66
Time	6m 2s	7m 11s	8m 42s	10m 4s	11m 20s	31m 32s
Peak VRAM	7.8 GB	8.7 GB	9.2 GB	10.3 GB	11.7 GB	8.3 GB

Table 2. **Performance analysis for different number of images.**

	3DGS [3] (Single time)	4DGS [17]	CL-Splats [1]	LTGS (Ours)
Memory	125.8 MB	172.8 MB	192.9 MB	128.5 MB
FPS	190.3	82.6	34.1	102.8

Table 3. **Memory and dynamic rendering FPS comparison.**

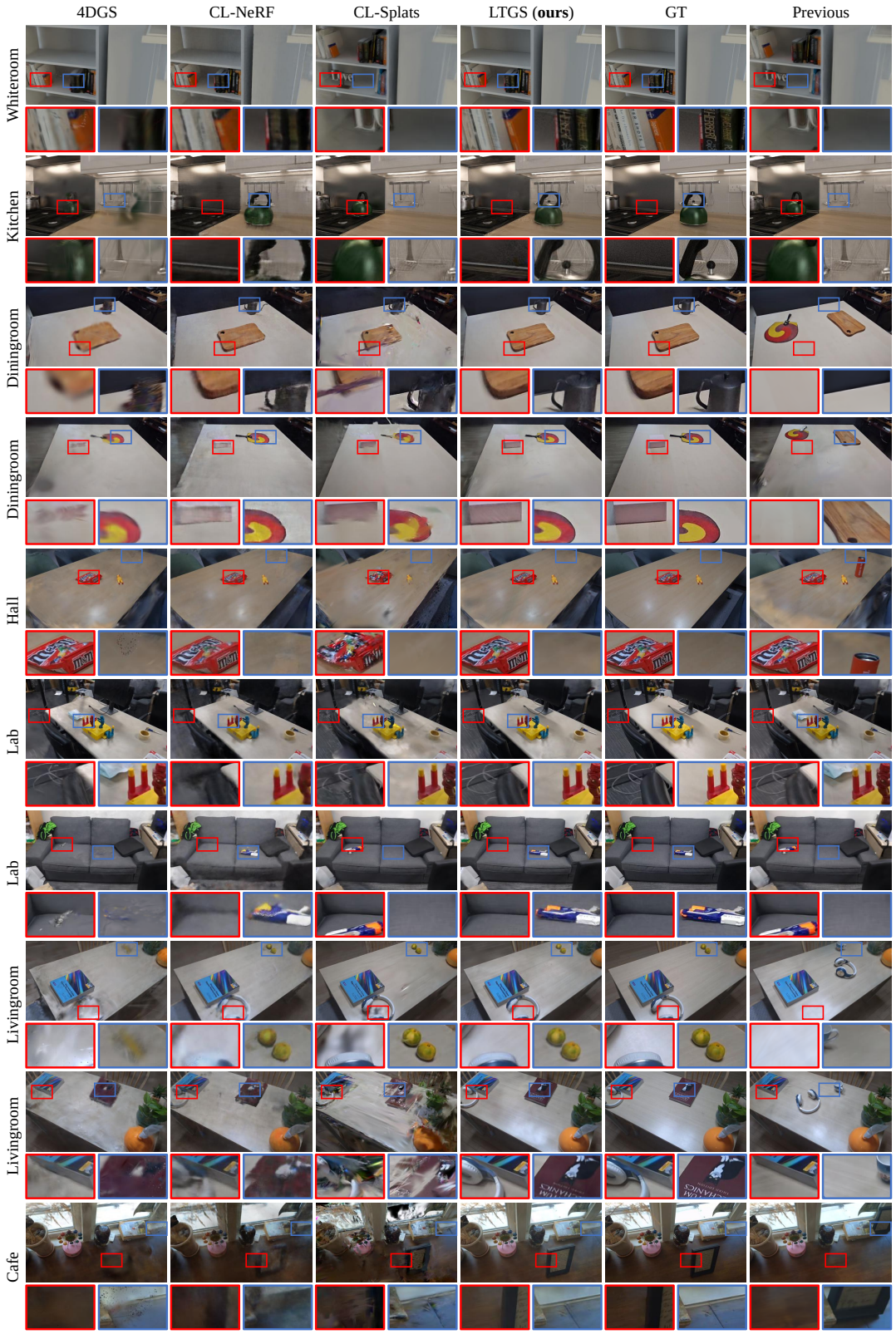


Figure 4. **Additional qualitative comparisons.** We illustrate the results of our method and baselines using CL-NeRF dataset and our dataset.

Method	CL-NeRF dataset			Our dataset				
	Whiteroom	Kitchen	Rome	Cafe	Diningroom	Livingroom	Hall	Lab
3DGS [3]	20.66	24.55	<u>28.38</u>	15.97	18.44	20.44	22.35	20.60
InstantSplat [2]	19.35	17.66	19.92	17.71	22.14	19.30	18.66	18.98
4DGS [17]	24.84	25.41	28.15	18.01	<u>22.84</u>	22.51	<u>23.33</u>	20.76
NSC [7]	18.13	17.40	26.36	15.10	18.21	19.81	18.63	15.86
3DGS-CD [8]	23.86	22.20	24.77	17.94	22.15	21.00	22.22	21.41
CL-NeRF [18]	26.22	<u>25.81</u>	24.55	16.53	22.66	22.32	22.40	20.87
CL-Splats [1]	**26.84**	24.90	25.79	<u>18.71</u>	19.29	**23.94**	22.21	<u>21.47</u>
LTGS (ours)	<u>26.67</u>	**26.21**	**28.65**	**20.77**	**25.22**	<u>23.50</u>	**25.22**	**22.64**

Table 4. **PSNR comparisons on CL-NeRF dataset and our dataset.** The first and second best results are highlighted in **bold** and <u>underlined</u>, respectively.

Method	CL-NeRF dataset			Our dataset				
	Whiteroom	Kitchen	Rome	Cafe	Diningroom	Livingroom	Hall	Lab
3DGS [3]	0.821	**0.645**	**0.899**	<u>0.776</u>	0.866	<u>0.897</u>	<u>0.911</u>	<u>0.835</u>
InstantSplat [2]	0.699	0.443	0.660	0.723	0.857	0.795	0.809	0.739
4DGS [17]	0.827	0.644	0.885	0.772	<u>0.893</u>	0.882	0.892	0.812
NSC [7]	0.710	0.536	0.849	0.705	0.795	0.813	0.804	0.658
3DGS-CD [8]	0.815	0.563	0.803	0.719	0.792	0.797	0.751	0.812
CL-NeRF [18]	0.829	0.636	0.725	0.696	0.863	0.881	0.859	0.775
CL-Splats [1]	**0.848**	0.627	0.840	0.749	0.873	0.836	0.866	0.819
LTGS (ours)	**0.848**	**0.645**	<u>0.892</u>	**0.845**	**0.911**	**0.922**	**0.924**	**0.840**

Table 5. **SSIM comparisons on CL-NeRF dataset and our dataset.** The first and second best results are highlighted in **bold** and <u>underlined</u>, respectively.

Method	CL-NeRF dataset			Our dataset				
	Whiteroom	Kitchen	Rome	Cafe	Diningroom	Livingroom	Hall	Lab
3DGS [3]	0.522	0.537	**0.116**	<u>0.323</u>	0.313	<u>0.216</u>	<u>0.234</u>	<u>0.273</u>
InstantSplat [2]	0.561	0.579	0.257	0.328	<u>0.295</u>	0.367	0.377	0.346
4DGS [17]	0.539	0.547	0.148	0.363	0.306	0.307	0.299	0.334
NSC [7]	0.585	0.619	0.192	0.441	0.385	0.426	0.431	0.510
3DGS-CD [8]	0.527	0.571	0.213	0.337	0.366	0.345	0.375	0.318
CL-NeRF [18]	0.521	<u>0.536</u>	0.339	0.463	0.324	0.328	0.350	0.428
CL-Splats [1]	<u>0.492</u>	0.542	0.214	0.345	0.308	0.323	0.303	0.280
LTGS (ours)	**0.478**	**0.522**	<u>0.128</u>	**0.246**	**0.253**	**0.185**	**0.204**	**0.260**

Table 6. **LPIPS comparisons on CL-NeRF dataset and our dataset.** The first and second best results are highlighted in **bold** and <u>underlined</u>, respectively.

References

[1] Jan Ackermann, Jonas Kulhanek, Shengqu Cai, Xu Haofei, Marc Pollefeys, Gordon Wetzstein, Leonidas Guibas, and Songyou Peng. Cl-splats: Continual learning of gaussian splatting with local optimization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2025. 4, 6

[2] Zhiwen Fan, Kairun Wen, Wenyan Cong, Kevin Wang, Jian Zhang, Xinghao Ding, Danfei Xu, Boris Ivanovic, Marco Pavone, Georgios Pavlakos, Zhangyang Wang, and Yue Wang. Instantsplat: Sparse-view gaussian splatting in seconds, 2024. 6

[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. 3, 4, 6

[4] Jae-Woo Kim and Ue-Hwan Kim. Towards generalizable scene change detection. In *Proceedings of the Computer Vision and Pattern Recognition Conference (CVPR)*, pages 24463–24473, 2025. 1, 2

[5] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollár, and Ross Girshick. Segment anything. *arXiv:2304.02643*, 2023. 1, 2

[6] Vincent Leroy, Yohann Cabon, and Jerome Revaud. Grounding image matching in 3d with mast3r, 2024. 1, 2, 4

[7] Haotong Lin, Qianqian Wang, Ruojin Cai, Sida Peng, Hadar Averbuch-Elor, Xiaowei Zhou, and Noah Snavely. Neural scene chronology. In *CVPR*, 2023. 6

[8] Ziqi Lu, Jianbo Ye, and John Leonard. 3dgs-cd: 3d gaussian splatting-based change detection for physical object rearrangement. *IEEE Robotics and Automation Letters*, 2025. 6

[9] Ben Mildenhall, Pratul P. Srinivasan, Rodrigo Ortiz-Cayon, Nima Khademi Kalantari, Ravi Ramamoorthi, Ren Ng, and Abhishek Kar. Local light field fusion: Practical view synthesis with prescriptive sampling guidelines. *ACM Transactions on Graphics (TOG)*, 2019. 3

[10] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020. 3

[11] James Munkres. Algorithms for the assignment and transportation problems. *Journal of the society for industrial and applied mathematics*, 5(1):32–38, 1957. 2

[12] Nobuyuki Otsu et al. A threshold selection method from gray-level histograms. *Automatica*, 11(285-296):23–27, 1975. 1

[13] Paul-Edouard Sarlin, Cesar Cadena, Roland Siegwart, and Marcin Dymczyk. From coarse to fine: Robust hierarchical localization at large scale. In *CVPR*, 2019. 2

[14] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 3

[15] Qiuhong Shen, Xingyi Yang, and Xinchao Wang. Flashsplat: 2d to 3d gaussian splatting segmentation solved optimally. *European Conference of Computer Vision*, 2024. 2

[16] Robert Tarjan. Depth-first search and linear graph algorithms. *SIAM journal on computing*, 1(2):146–160, 1972. 1

[17] Guanjun Wu, Taoran Yi, Jiemin Fang, Lingxi Xie, Xiaopeng Zhang, Wei Wei, Wenyu Liu, Qi Tian, and Xinggang Wang. 4d gaussian splatting for real-time dynamic scene rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 20310–20320, 2024. 4, 6

[18] Xiuzhe Wu, Peng Dai, Weipeng Deng, Handi Chen, Yang Wu, Yan-Pei Cao, Ying Shan, and Xiaojuan Qi. Cl-nerf: continual learning of neural radiance fields for evolving scene representation. *Advances in Neural Information Processing Systems*, 36:34426–34438, 2023. 4, 6