GS-LIVM: Real-Time Photo-Realistic LiDAR-Inertial-Visual Mapping with Gaussian Splatting

Supplementary Material

6. Pseudocode for Preprocessing Algorithm ALG.STORE & CUDA Batched Algorithm ALG.SOLVE in Voxel-GPR

Preprocessing algorithm for latest scanned point cloud P_f is shown in Algorithm 2. Each point in P_f is stored into a voxel structure, and the visited voxel information in this frame is recoeded simultaneously. Utilizing the cuBLAS scientific computing libraLivlry within the GPU, the pseudocode for batch-solving (3) for all voxels is shown as Algorithm 3.

Algorithm 2: Preprocessing for single-frame collected point clouds

Input: P_f, $\mathcal{C}$ $(Global\ Stored\ Point\ Cloud\ in\ Voxels)$
Output: $\mathcal{S}_{updated}(Store\ Visited\ Voxels\ in\ This\ Frame)$

$\mathcal{S}_{updated} \leftarrow [\,]$
for p **in** P_f **do**
  H_p = ALG.HASH(p^x, p^y, p^z)
  push p **to** $\mathcal{C}[H_p] \rightarrow points$
  push H_p **to** $\mathcal{S}_{updated}$
end

Algorithm 3: Batch solving μ_* and Σ_* in Voxel-GPR based on cuBLAS

Input: $\mathbf{f}$, $\mathbf{K}$, $\mathbf{K}_*$, $\mathbf{K}_{**}$
Output: μ_*, Σ_*

$\#\ Add\ Noise$
  $\mathbf{ky} \leftarrow \mathbf{K} + \sigma^2\mathbf{I}$
$\#\ Inplace\ LU\ Decomposition$
  cublasSgetrfBatched(ky)
$\#\ Calculate\ Inverse\ Matrix\ of\ \mathbf{ky}$
  kyInv $\leftarrow$ cublasSgetriBatched(ky)
$\#\ Matrix\ Multiplication$
  $k_{k*} \leftarrow$ cublasSgemmBatched($\mathbf{K}_*$, kyInv)
  $\mathbf{f}_* \leftarrow \mu_* \leftarrow$ cublasSgemmBatched(k_{k*}, $\mathbf{f}$)
  $\Sigma_* \leftarrow \mathbf{K}_{**}$ - cublasSgemmBatched(k_{k*}, $\mathbf{K}_*$)

7. Supplementary Experiments

7.1. Comparison with Latest Baselines

We compare our method with the latest SOTA multi-sensor fusion 3DGS framework [16, 40]. But neither of these methods has open-source code available, so we can only replicate the content from their papers for the comparison.

The rendering performance comparison with LiV-GS [40] is demonstrated in Table 4. We refer to the results in their preprint (indicated as $*$), and our method outperforms LiV-GS in rendering quality. LiV-GS ignores the discussion of computational efficiency or GPU usage, nor is their code publicly available. Therefore, we are unable to compare the time efficiency with LiV-GS. We add GPU usage and mapping FPS comparison with baselines to show our method's efficiency (Table 5 and Table 6) and accuracy (Table 5). Notwithstanding this code un-availablity, we trust that our results clearly underscore the significance of our approach.

Gaussian-LIC [16] relies on down-sampling the original point cloud to reduce data volume to achieve faster processing speed. In contrast, our approach leverages a more robust GPR to regress voxels, which is a more effective solution. Due to the absence of open-source code, the method used to calculate the rendering metrics in their preprint remains unclear, so we cannot guarantee a fair comparison. In terms of algorithmic efficiency, we refer to the results in their preprint, as included in Table 6. Importantly, our method achieves real-time performance even on an RTX 4060 Laptop GPU, in contrast with Gaussian-LIC [16] that was executed the more powerful RTX 3090 GPU.

Table 4. Rendering performance comparison with LiV-GS.

Methods	cp		nyl2	
	PSNR↑	SSIM↑	PSNR↑	SSIM↑
LiV-GS*	22.27	**0.775**	20.83	0.725
Ours	**23.54**	0.733	**22.09**	**0.738**

Table 5. GPU usage (Mb) under the same optimization time.

Sequence	NeRF-SLAM	MonoGS	3DGS	Ours
hku_campus_seq_00	6823	4092	4728	**2356**
Visual_Challenge	4312	2803	3067	**1302**

Table 6. Mapping FPS comparison. $*$ denotes paper's result.

	NeRF-SLAM	MonoGS	3DGS	Gaussian-LIC*	Ours
FPS	3.1	5.3	-	10.0	**12.56**

7.2. Comparison with Fully Optimized Offline 3DGS

In the rendering comparison presented in Figure 5, to ensure a fair comparison, we limit the training of the original 3DGS implementation to 3000 iterations. Interestingly, we discover that with sufficient optimization time, 3DGS

indeed achieves better metrics than our method (Table 7). This is because 3DGS involves offline optimization over an extended period to overfit the training data. Nonetheless, limitations are encountered with NVS, as demonstrated in Figure 8. Overfitting to the training dataset causes 3D Gaussians to extend beyond the observation point, which is the primary issue with offline methods. It can be observed that our method can overcome the drawbacks of overfitting.

Table 7. Comparison with the fully optimized offline 3DGS [13].

Sequence	3DGS			Ours		
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
hku_seq_00	27.827	0.861	0.204	22.430	0.719	0.247
1005_00	26.196	0.812	0.233	21.123	0.679	0.258

(a) 3DGS [13] (b) Ours

(c) 3DGS [13] (d) Ours

Figure 8. In all subfigures, the top images represent a supervised perspective with image information, while the bottom images depict an unsupervised perspective with synthesized new viewpoints. Results in (a)(c) are obtained using the offline 3DGS method, while results in (b)(d) are obtained using our method.

7.3. More Rendering and Mapping Results

We collect a Livox MID-360 (*private-360*) and a HESAI Pandar XT-32 (*private-pandar*) sequence with image resolution 640×512 to verify the performance of our algorithm in different LiDAR types additionally. The expanded comparision of rendering performance are shown in Table 8.

It is worth noting that our method can reconstruct the entire scene with limited GPU resources (8GB), which is a feature unattainable by other methods [7, 16]. In Figure 9, we present snapshots of rendered images in corresponding whole sequences. It can be observed that our method maintains excellent rendering quality even in large outdoor scenes under the real-time mapping requirement.

Since our 3D Gaussians information is initialized from point clouds, the rendering performance for sparse multi-line spinning LiDAR is inferior to that of the Livox LiDAR due to the difference in point cloud density. This difference can be observed in the Botanic Garden sequence in Figure 9.

The reconstructed 3D Gaussian models of sequences *hku_main_building* and *1005_01* are also presented in Figure 10. Sequence *hku_main_building* has a duration of 20 minutes and a trajectory length of approximately 900 meters, yet we are still able to achieve real-time map expansion. Outdoor park (sequence *1005_01*) typically contains rich structural and textural features, making photo-realistic mapping highly challenging. Using Velodyne VLP-16 LiDAR in the outdoor park, our proposed algorithm is able to achieve a good dense reconstruction model.

7.4. More Ablation Experiments

We provide a quantitative comparison of Voxel-GPR and fast Initialization modules in Table 9 and Figure 11. **Bold** values indicate the best results.

We supplement the ablation experiments with relevant quantitative results to validate the following:

- **noG/noI**: Direct input of the original point cloud.
- **G/noI**: With Voxel-GPR but without fast initialization.
- **G/I**: With Voxel-GPR and with fast initialization.

Note: The combination **noG/I** is invalid as fast initialization depends on Voxel-GPR. Table 9 shows the qualitative ablation results. Figure 11 illustrates the efficiency of our different modules in terms of convergence speed, using a single viewpoint as an example.

Expanded ablation experiments on additional sequences are presented in Table 10. In challenging environments, such as sensor degradation (*Visual_Challenge*) or very sparse point clouds (*eee_02*), the experiments continue to show that our module design is robust, enhancing photorealistic mapping effects in these scenarios.

7.5. Tracking Performance

Our photo-realistic mapping method utilizes a general SLAM framework to obtain precise camera poses. In theory, this approach can be integrated into any general LIVO framework. In our experiments, we adopt the framework from [47]. Compared to the original [47], we introduc the following improvements: 1) We replace the time sweep-based refinement method with the sensor's original timestamps at a finer granularity, resolving the crash issue when using multi-line spinning LiDAR. 2) We apply CUDA en-

Table 8. Quantitative rendering performance comparison on more datasets. The results ranked from best to worst are highlighted as first , second , and third .

Sequence	LiDAR Type	Nerf-SLAM [26]			MonoGS [21]			3DGS [13]			Ours		
		PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM ↑	LPIPS↓	PSNR↑	SSIM ↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
hkust_campus_02	Livox Avia	10.728	0.389	0.716	12.408	0.426	0.582	17.629	0.692	0.412	15.745	0.634	0.382
hku1	Livox Avia	9.232	0.384	0.716	8.928	0.286	0.750	19.548	0.708	0.328	14.990	0.600	0.424
1018_00	Velodyne VLP-16	12.186	0.387	0.699	12.721	0.634	0.417	23.452	0.720	0.253	16.971	0.688	0.338
private-360	Livox MID-360	13.681	0.407	0.598	8.912	0.263	0.778	22.428	0.693	0.302	17.777	0.618	0.439
private-pandar	Pandar XT-32	9.492	0.328	0.610	7.414	0.289	0.623	18.647	0.658	0.369	16.234	0.600	0.330

Table 9. Quantitative ablation results of the proposed modules.

Settings	Count	MT(s)	Mem(Mb)	PSNR↑	SSIM↑
noG/noI	3,294,762	2867	7943	14.258	0.459
G/noI	1,170,324	334	2592	19.234	0.684
G/I	1,170,324	**202**	**2495**	**22.163**	**0.732**

Table 10. Quantitative ablation comparison of hyperparameters in various sequences, such as duration (DT), mapping time (MT), interval of 3D Gaussians in voxel side n_s, convergence variance of Voxel-GPR η, structure similarity loss (SS), and delta depth similarity loss (DDS).

Sequence	DT (s)	MT (s)	n_s	η	SS	DDS	PSNR	SSIM
Visual_ Challenge	162	166	3	0.1	✗	✗	18.479	0.608
		162	3	0.3	✗	✗	17.326	0.617
		162	3	0.3	✓	✗	17.599	0.626
		162	3	0.3	✓	✓	18.127	0.613
		162	4	0.3	✓	✓	18.422	0.659
eee_02	321	321	3	0.2	✗	✗	12.766	0.436
		321	3	0.3	✗	✗	12.136	0.430
		321	3	0.3	✓	✗	12.395	0.434
		321	3	0.3	✓	✓	12.512	0.441
		321	4	0.3	✓	✓	14.328	0.473

capsulation for certain large matrix multiplications to accelerate optimization. To ensure the fairness of comparisons, we conduct tracking experiments on the improved [47].

We compare improved method with filter-based method R^{3}LIVE [18], FAST-LIO2 [42], and graph optimization based method LVI-SAM [31]. In tracking performance evaluation, we focus on key performance metrics Relative Pose Error (RPE) and Absolute Trajectory Error (ATE) [32], considering full transformations that encompass both rotation and translation.

To assess the adaptability of our dataset for tracking, we meticulously select 7 representative sequences from the Botanic dataset [20] and conduct comprehensive evaluations on cutting-edge algorithms, including those presented in [18, 31, 42], by comparing them against ground truth data. The data used for our tracking algorithm comes from a Velodyne VLP-16 multi-line spinning LiDAR. Table 11 presents the evaluation metrics on 7 sequences from the Botanic Garden [20] dataset. Our method, lacking a loop closure detection module, exhibits lower accuracy on some sequences compared to LVI-SAM [31]. Overall, comparisons with above state-of-the-art methods reveal that our approach demonstrates clear advantages on certain sequences. Figure 12 illustrates the trajectory comparison for the sequence *1006_01* within the Botanic Garden [20] dataset, featuring a trajectory length of approximately 730 meters. In the top right corner of Figure 12, an enlarged view of a *three-way junction* on the **X-Y** plane is presented, while the bottom right corner displays an elevation map of the same junction on the **Z** plane. From the visual representation, it is evident that our trajectory aligns most closely with the ground truth, showcasing the highest qualitative tracking precision. More comparison results are available on Figure 13.

7.6. Time Consumption and Memory Usage on Ultra-Long Time Sequence

Addressing SLAM and photo-realistic mapping in ultra-long time sequence is a challenging issue. As the scale of the map increases, insufficient available storage space leads to a slowdown or interruption in the mapping process. To deal with these problems, we conduct tests on two ultra-long sequences in test dataset. With the limited GPU resources, we set n_r=2 and obtain the mapping time, rendering metrics, as shown in Table 12. The time and GPU memory consumption of each algorithm module over time are depicted in Figure 14. In the later stages of the long time series, most of the time is spent on rendering the ultra-large 3DGS map, map feedback, and optimization. It can be seen that our method is still able to achieve real-time map expansion in ultra-long sequences, with resource consumption in ultra-large scenes remaining within acceptable levels.

7.7. Hyper Parameters in Our Experiments

The hyper parameters used in all our sequences are listed in Table 13. For more configuration details, please refer to the *config* file in our GitHub repository.

8. Limitations

In this paper, real-time performance is our primary focus, with all modules specifically designed to minimize the time complexity of the algorithm, thus ultimately enabling real-time photo-realistic mapping in large-scale open environments. However, the study does have some limitations:

Figure 9. Illustration of the rendering performance of the entire sequence from a dataset. It is noteworthy that these illustrations are all rendered based on the real-time mapping results. These scenes are *hkust_campus_seq_00 (Livox AVIA)*, *Visual_Challenge (Livox AVIA)*, *eee_03 (Ouster 16)*, *1005_00 (Livox AVIA)*, *hku_park_00 (Livox AVIA)*, *hku_main_building (Livox AVIA)*, *1006_01 (Livox AVIA)*, *1006_01 (Velodyne VLP-16)*, *1005_01 (Velodyne VLP-16)*, and *eee_03 (Ouster 16)*, respectively.

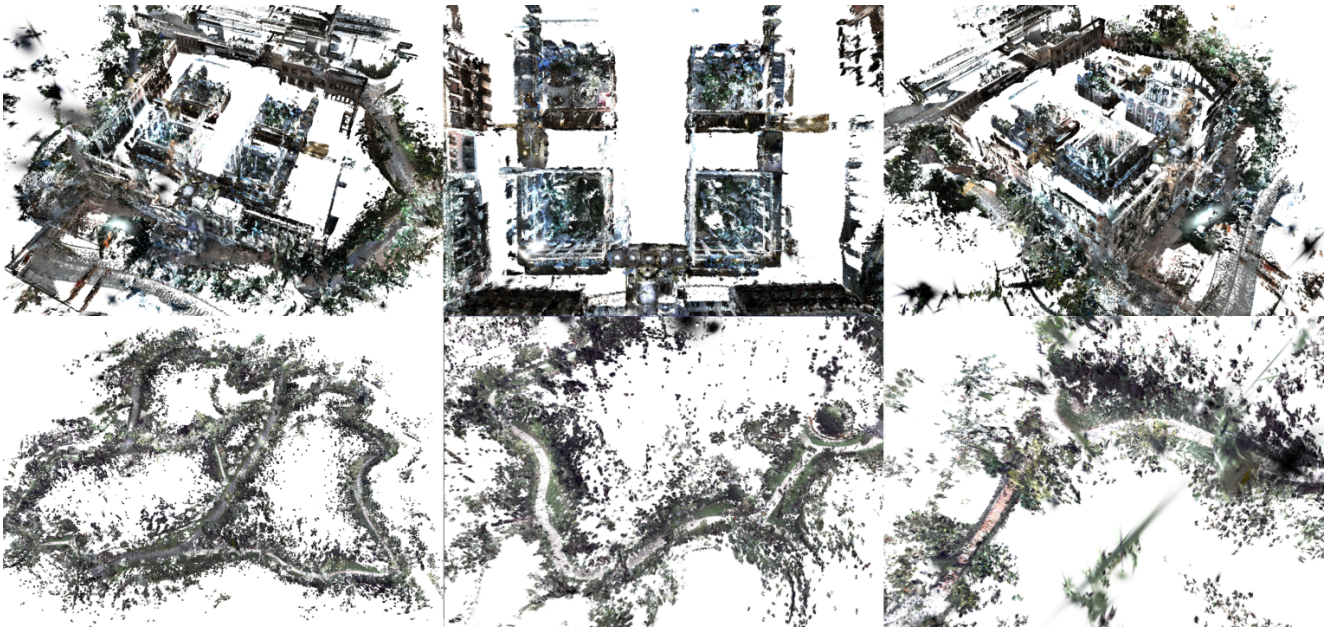

Figure 10. More illustrations of the 3D Gaussians model of the entire sequence. These two scenes are *hku_main_building ((Livox AVIA))* and *1005_01 (Velodyne VLP-16)*, respectively.

Table 11. Quantitative tracking performance comparison of LiDAR-IMU SLAM method [42] and LiDAR-Visual-IMU fusion SLAM method [18, 31] on BotanicGarden dataset [20].

	1005_00		1005_01		1005_07		1006_01		1008_03		1018_00		1018_13	
	RPE↓	ATE↓	RPE↓	ATE↓	RPE↓	ATE↓	RPE↓	ATE↓	RPE↓	ATE↓	RPE↓	ATE↓	RPE↓	ATE↓
R3LIVE [18]	1.165	3.153	1.151	1.451	2.112	3.893	0.934	3.505	2.050	3.383	0.165	0.378	0.133	0.366
FAST-LIO2 [42]	1.048	2.665	0.652	0.483	0.947	0.751	1.047	1.521	0.852	0.798	0.241	0.187	0.245	0.308
LVI-SAM [31]	0.347	0.312	0.147	0.129	0.127	0.257	0.462	0.410	0.272	0.252	0.139	0.044	0.152	0.051
OURS	0.174	0.291	0.058	0.073	0.061	0.496	0.202	0.702	0.068	0.414	0.055	0.075	0.050	0.077

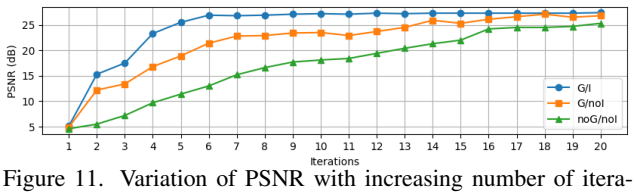

Figure 11. Variation of PSNR with increasing number of iterations.

Table 12. Duration (DT), mapping time (MT), count of 3D Gaussians, maximum memory cost (Mem), and rendering metrics on ultra long sequence.

Sequence	DT (s)	MT (s)	Count	Mem (Mb)	PSNR	SSIM
hku_main_building	1170	1170	2,674,234	4489	15.234	0.538
hkust_campus_00	1073	1090	3,302,675	5812	15.124	0.496

The initialization of the 3DGS model is solely reliant on the point cloud data, which results in initialization failure in areas lacking point cloud coverage, leading to missing regions. Future research will explore integrating image data to aid in the initialization process. Moreover,

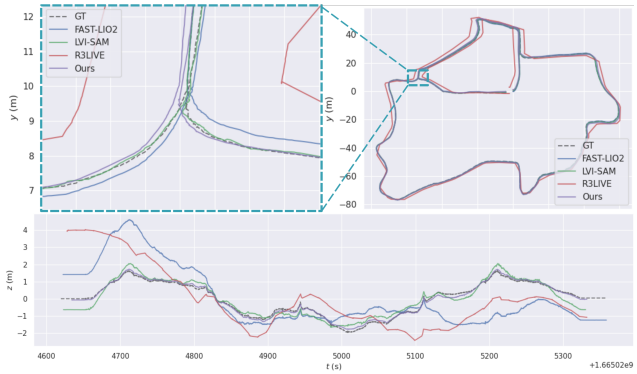

Figure 12. The trajectory comparison for the Botanic Garden sequence *1006_01* is illustrated, featuring a detailed illustration of a *three-way intersection* within the scene in the upper left, with the offset along the **Z** plane for this junction depicted below. It is evident from this comparison that our trajectory aligns most closely with the actual trajectory, demonstrating a high level of tracking precision.

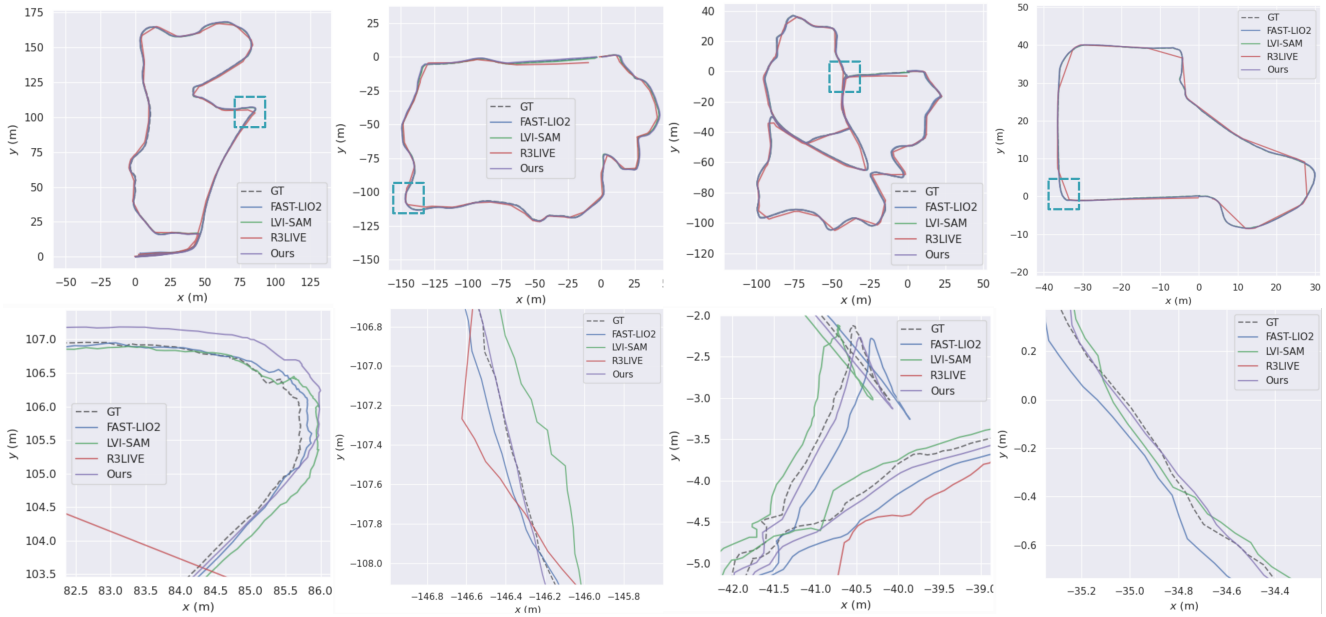

Figure 13. More trajectory comparisons. The first row of images shows the trajectory of the entire scene, while the second row of images displays the enlarged results within the corresponding blue boxes. The four scenes are *1005_00*, *1005_07*, *1008_03*, *1008_13*, respectively.

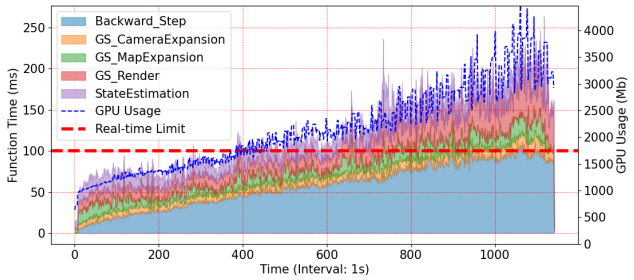

Figure 14. Time consumption and GPU memory consumption on ultra long scequence *hku_main_building* (Duration: 1170 s).

due to the real-time constraints, the reconstruction quality may not match that of images obtained through actual data acquisition. Improving the reconstruction quality will be part of our future research endeavors. Regarding the pre-processing of the sky area in our method, LiDAR does not capture data in the sky region during scanning. Consequently, this data is inherently excluded. Furthermore, other studies have proposed strategies to address the sky region, primarily leveraging semantic information for correction. To address the sky region appropriately is crucial for outdoor scenes, and this would be part of our future work.

Table 13. Hyper parameters used in all datasets.

Name	Value	Description
size	0.2	Side length of each voxel.
τ	10	Point count threshold for voxel processing.
n_s	3	Interval between each side within the voxel.
n_r	3	Number of neighboring Gaussians per side.
η	0.3	Convergence threshold for Voxel-GPR.
λ_{SSIM}	0.2	Weight for the photometric SSIM loss.
λ_p	0.1	Weight for the structural similarity loss.
λ_d	0.1	Weight for delta depth similarity loss.
$\mathcal{T}$	50	Size of the sliding window for recent images.
k_{curr}	1	Number of frames selected from the current window.
k_{his}	1	Number of frames selected from historical data.
$lr_{position}$	0.0005	Learning rate for position parameters.
lr_{color}	0.0025	Learning rate for color parameters.
$lr_{opacity}$	0.025	Learning rate for opacity parameters.
lr_{scale}	0.0025	Learning rate for scale parameters.
$lr_{rotation}$	0.0025	Learning rate for rotation parameters.