LVI-GS: Tightly Coupled LiDAR–Visual–Inertial SLAM Using 3-D Gaussian Splatting

Huibin Zhao, Weipeng Guan, and Peng Lu

***Abstract*— Three-dimensional Gaussian splatting (3DGS) has shown its ability in rapid rendering and high-fidelity mapping. In this article, we introduce a tightly coupled LiDAR–visual–inertial SLAM using 3-D Gaussian splatting (LVI-GS), which leverages the complementary characteristics of light detection and ranging (LiDAR) and image sensors to capture both geometric structures and visual details of 3-D scenes. To this end, the 3-D Gaussians are initialized from colorized LiDAR points and optimized using differentiable rendering. To achieve high-fidelity mapping, we introduce a pyramid-based training approach to effectively learn multilevel features and incorporate depth loss derived from LiDAR measurements to improve geometric feature perception. Through well-designed strategies for Gaussian map expansion, keyframe selection, thread management, and custom compute unified device architecture (CUDA) acceleration, our framework achieves real-time photorealistic mapping. Numerical experiments are performed to evaluate the superior performance of our method compared with state-of-the-art 3-D reconstruction systems. Videos of the evaluations can be found on our website: https://kwanwaipang.github.io/LVI-GS/.**

***Index Terms*— 3-D Gaussian splatting (3DGS), 3-D reconstruction, light detection and ranging (LiDAR), robotics, sensor fusion, simultaneous localization and mapping (SLAM).**

I. Introduction

SIMULTANEOUS localization and mapping (SLAM) systems are indispensable across various domains, including robotics, augmented reality, and autonomous navigation [1]. These systems enable devices to understand and navigate complex environments by constructing maps while simultaneously estimating their positions within these spaces. For effective SLAM, both accurate localization and comprehensive scene reconstruction are essential.

Traditional SLAM systems represent environments using landmarks [2], point clouds [3], occupancy grids [4], signed distance function (SDF) voxel grids [5], or meshes [6]. Among these, point clouds are a straightforward scene representation readily obtainable from sensors such as cameras and light detection and ranging (LiDAR). Point-cloud-based SLAM systems can achieve accurate localization and construct either sparse or dense maps, though these tend to lack visually rich details.

The advent of neural radiance fields (NeRFs) has introduced a new approach for high-fidelity scene reconstruction [7]. NeRF implicitly represents a scene within a radiance field by optimizing a continuous volumetric scene function, which requires minimal memory. Some NeRF-based SLAM methods leverage this framework's novel view synthesis and high-fidelity reconstruction capabilities to model scenes. For instance, iMAP [8] constructs an implicit 3-D occupancy and color model usable for tracking, while NICE-SLAM [9] represents larger scenes via a coarse-to-fine approach. Enhanced methods such as Vox-Fusion [10], CoSLAM [11], and ES-SLAM [12] advance SLAM system performance to varying extents. However, due to the extensive optimization processes involved, these systems struggle to achieve real-time performance. Moreover, storing maps within multilayer perceptrons poses challenges, including catastrophic forgetting and limited boundaries, which can impede scene reconstruction.

Three-dimensional Gaussian splatting (3DGS) offers an exciting alternative, providing a continuous and adaptable representation for modeling 3-D scenes through differentiable 3-D Gaussian-shaped primitives [13], [14]. As a semi-implicit mapping approach, it trades off some novel view synthesis capabilities for significantly faster optimization and rendering speeds. Despite being based on optimization, 3DGS closely resembles point and surfel clouds, thus inheriting their efficiency, locality, and adaptability—attributes beneficial for SLAM mapping. With rendering speeds up to 200 frames/s at a 1080p resolution, 3DGS also initializes using point clouds, allowing it to leverage the sparse or dense point clouds generated by conventional SLAM systems for high-fidelity imagery [15].

Recently, several SLAM approaches integrating 3-D Gaussians have shown promising results. Methods such as SplaTAM [16], MonoGS [15], GS-SLAM [17], and Photo-SLAM [18] use sequential RGB-D or RGB data to establish complete SLAM systems. However, these techniques encounter difficulties in large-scale, uncontrolled outdoor environments characterized by challenging lighting, complex backgrounds, and rapid motion. Although LiDAR offers high-quality geometric initialization for 3-D Gaussians and is generally more robust in outdoor settings than cameras, integrating it into SLAM systems introduces unique challenges. LIV-Gaussianmap [19] and LetsGo [20] use LiDAR for initializing 3-D Gaussians, while Gaussian-LIC [21] combines a LiDAR–inertial–camera setup for comprehensive 3-D Gaussian construction. Nevertheless, systems such as

Received 6 November 2024; revised 4 January 2025; accepted 7 February 2025. Date of publication 14 March 2025; date of current version 27 March 2025. This work was supported in part by the General Research Fund under Grant 17204222 and in part by the Seed Fund for Collaborative Research and General Funding Scheme-HKU-TCL Joint Research Center for Artificial Intelligence. The Associate Editor coordinating the review process was Dr. Chuang Sun. *(Huibin Zhao and Weipeng Guan contributed equally to this work.) (Corresponding author: Peng Lu.)*

The authors are with the Adaptive Robotic Controls Lab (ArcLab), Department of Mechanical Engineering, The University of Hong Kong, Hong Kong, SAR, China (e-mail: lupeng@hku.hk.)

Digital Object Identifier 10.1109/TIM.2025.3551585

TABLE I
SUMMARIZATION OF DIFFERENT NERF-BASED SLAM AND 3DGS-BASED SLAM

Category	Method Type	Typical Methods	Features
NeRF-based SLAM	Image based	iMAP [8]; NICE-SLAM [9]; Vox-Fusion [10]; ESLAM [12]; Co-SLAM [11]; NeRF-SLAM [22]; NeRF-VO [23]	- Strong capability for novel view synthesis - Low rendering efficiency - Expensive training cost
3DGS-based SLAM	Image based	MonoGS [15]; Photo-SLAM [18]; SplaTAM [16]; GS-SLAM [17]; GS-ICP [24]	- Fast rendering and high fidelity - Fast training speed - Noisy initialization
	LiDAR-Image based	Gaussian-LIC [21]; MM-Gaussian [25]; MM3DGS-SLAM [26]	- Fast rendering and high fidelity - Accurate initialization - Robust localization

LIV-Gaussianmap [19] and LetsGo [20] are limited to offline processing, and Gaussian-LIC [21] requires complex front-end odometry and maintaining a substantial number of keyframes.

In general, the main contributions of this study can be outlined as follows.

1) We have developed and implemented a sophisticated real-time tightly coupled LiDAR–visual–inertial SLAM using 3-D Gaussian splatting (LVI-GS) system that is capable of maintaining a dynamic hyper primitives module. This system leverages 3DGS to perform high-quality, real-time rendering in 3-D space, ensuring an efficient and accurate representation of complex environments.
2) To further enhance the performance and scalability of our system, we used a coarse-to-fine map construction approach. This method uses both RGB image pyramids and depth image pyramids to progressively refine the map at different levels of detail. In addition, we implemented an advanced thread management technique to optimize computational efficiency, ensuring smooth real-time operations even with large datasets.
3) In pursuit of improved map representation and rendering quality, we designed a robust keyframe management strategy that allows for the effective selection and processing of keyframes. Moreover, by incorporating depth loss into the system, we enhanced the accuracy of the 3-D Gaussian map, leading to more precise reconstructions and visually superior rendering results.

II. RELATED WORKS

A. NeRF-Based SLAM

Classic NeRF-based SLAM systems rely entirely on MLPs. They encode the environment within the weights of a neural network. Pixel values are predicted by decoding the MLP and integrating sampled rays through the volume via volumetric rendering. As the first NeRF-based SLAM system, iMAP [8] ensures close-to-frame-rate camera tracking by joint optimization of photometric and geometric losses. Limited by MLP capacity, subsequent works integrate implicit MLPs with structural features to address catastrophic forgetting. For example, NICE-SLAM [9] adopts a hierarchical strategy that integrates multilevel local data. Vox-Fusion uses octree to manage voxels. ESLAM [12], Co-SLAM [11], GO-SLAM [27], and Point-SLAM [28] introduce innovative approaches such as multiscale feature planes, combined smoothness, detail encoding, real-time global optimization, and dynamic neural point cloud representation, enhancing scene reconstruction. OrbeezSLAM [29], NeRF-SLAM [22], and NeRF-VO [23] improve tracking frequency and pose estimation accuracy of Nerf-based SLAM methods for odometry.

B. 3DGS-Based SLAM

The 3DGS-based methods offer faster, more realistic rendering, enhanced map capacity, and efficient optimization through dense losses compared with NeRF-based SLAM. SplaTAM [16] uses simplified 3-D Gaussians for accurate camera pose estimation and scene refinement. GS-SLAM [17] uses 3-D Gaussians, opacity, and spherical harmonics (SHs) for scene representation, featuring adaptive Gaussian management and robust camera tracking. MonoGS [15] is the first work to achieve 3DGS SLAM using a monocular camera. GS-ICP [24] SLAM uses the explicit characteristics of 3DGS using G-ICP [30] for the tracking process. Photo-SLAM [18] uses accurate pose estimation from ORB-SLAM3 [31] to reconstruct a hybrid Gaussian map. Some methods incorporate LiDAR as a sensor. Gaussian-LIC [21] realize a real-time LiDAR–inertial–camera SLAM system relying on a tightly coupled LiDAR–inertial–camera odometry. MM-Gaussian [25] develops a relocalization module that is designed to correct the system's trajectory in the event of localization failures. Meanwhile, MM3DGS-SLAM [26] uses a visual–inertial framework to optimize camera poses.

Notably, Table I summarizes the key differences between NeRF-based and 3DGS-based SLAM methods.

III. METHODOLOGY

Our framework shown in Fig. 1 implements the complete system functionality through two parallel threads. One thread handles odometry, while the other performs real-time optimization of 3-D Gaussians. Both the threads collaboratively maintain a shared hyper primitives module. Between these two threads, data such as 3-D point clouds, camera poses, camera images, and depth information are exchanged. Our input data come from a LiDAR, an RGB camera, and an IMU. Specifically, the LiDAR and camera are used throughout the

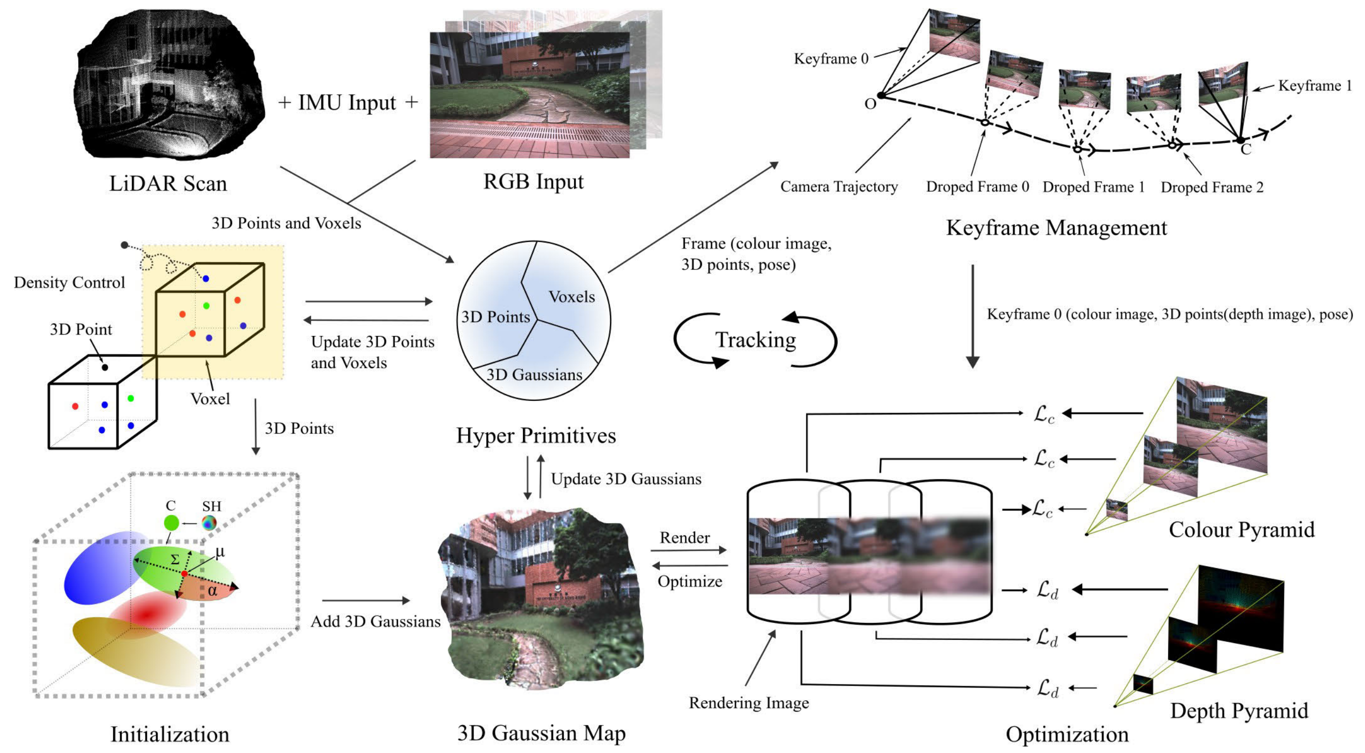

Fig. 1. Overview of the LVI-GS system.

entire SLAM pipeline. While the IMU is exclusively used in the localization module to provide high-frequency state predictions via the ESIKF.

A. Hyper Primitives

We maintain a hyper primitives module, consisting of 3-D point clouds, voxels, and 3-D Gaussians. To efficiently access 3-D point clouds for 3-D Gaussian initialization, the map points are organized into fixed-size voxels (e.g., 0.1 × 0.1 × 0.1 m). Voxel activation is determined by the presence of recently added points (e.g., within the last second). An activated voxel indicates recent activity, whereas a deactivated voxel signifies no recent updates.

In addition, within the visual–inertial odometry (VIO) module, points are removed if their projection or photometric errors exceed a specified threshold [32]. For each point in the point cloud, we identify its position within the grid; if a point already exists in that position, it is discarded. We also regulate the number of points within each voxel to maintain a controlled density. Through this initial filtering process, as odometry proceeds, the obtained point cloud avoids the redundant addition of 3-D Gaussians.

B. 3-D Gaussian Splatting

Our scene representation is 3DGS, mapping it with a set of anisotropic Gaussian $\mathcal{G}$. Each Gaussian includes opacity $o \in \mathbb{R}$, center position $\mu \in \mathbb{R}^3$, SHs $\mathrm{SH} \in \mathbb{R}^{16}$, and 3-D covariance matrix $\Sigma \in \mathbb{R}^{3\times 3}$. Because every Gaussian shape is an ellipsoid, we parameterize the 3-D Gaussian's covariance as

$$\Sigma = \mathrm{RSS}^T R^T \tag{1}$$

where $S \in \mathbb{R}^3$ is a vector to describe 3-D scale, and $R \in \mathrm{SO}(3)$ represents the rotation matrix. Given a center position μ and 3-D covariance matrix Σ, a Gaussian distribution is defined as

$$\mathcal{G}(x) = \exp\left(-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)\right). \tag{2}$$

Our ultimate goal is to project 3-D Gaussians onto a 2-D plane for rendering to obtain high-fidelity images. This process is commonly referred to as "splitting." We first obtain the camera's position transformation $P_{WC} \in \mathrm{SE}(3)$ between the world and the camera. Then, the function to project 3-D Gaussian (μ_W, Σ_W) in the world coordinates to splatted Gaussian (μ_C, Σ_C) on the image plane will be

$$\mu_C = \pi(P_{WC} \cdot \mu_W) \tag{3}$$

where π is the projection operation to pixel coordinates. Then we can calculate the covariance matrix of a splatted Gaussian

$$\Sigma_c = J R_{WC} \Sigma_W R_{WC}^T J^T \tag{4}$$

where $J \in \mathbb{R}^{2\times 3}$ is the Jacobian of the affine approximation of the projective transformation. And $R_{WC} \in \mathrm{SO}(3)$ is the rotation component of world-to-camera transformation P_{WC}. For a pixel $\rho = [u, v]^T$, the influence weight of a splatted Gaussian on its transparency is defined as

$$\alpha = o \cdot \exp\left(-\frac{1}{2}(\mu_C - \rho)^T \Sigma_C{}^{-1}(\mu_C - \rho)\right). \tag{5}$$

3DGS uses a volumetric rendering approach by directly rasterizing 3-D Gaussians, rather than relying on ray traversal. This method differs from mesh- or point-cloud-based techniques, as it can effectively disregard empty spaces during the rendering process without requiring precise knowledge of object surface locations. Through the amalgamation of $\mathcal{N}$ 3-D Gaussians via splatting and blending, a pixel's color $\mathcal{C}_\rho$ is determined

$$\mathcal{C}_\rho = \sum_{i\in\mathcal{N}} c_i \alpha_i \prod_{j=1}^{i-1}(1-\alpha_j) \tag{6}$$

where c denotes the RGB color of the Gaussian computed from SH. Similarly, using the same method, we can also obtain the depth $\mathcal{D}_\rho$ through

$$\mathcal{D}_\rho = \sum_{i\in\mathcal{N}} d_i \alpha_i \prod_{j=1}^{i-1}(1-\alpha_j). \tag{7}$$

We also render a visibility image and it is used to determine the visibility of the current pixel.

$$\mathcal{V}_\rho = \sum_{i\in\mathcal{N}} \alpha_i \prod_{j=1}^{i-1}(1-\alpha_j). \tag{8}$$

C. Keyframe Management

We obtain point clouds through the hyper primitives module, packaging every N_l points along with the corresponding camera pose and image into a single frame for future selection. All the colorized LiDAR points are used for initializing the 3-D Gaussians. We conduct keyframe selection on the images, evaluating those free from motion blur. Images that exhibit rotation or translation beyond a specified threshold are added as keyframes to the keyframe sequence

$$\|\mathbf{R}_i - \mathbf{R}_{i-1}\| > \tau_r \text{ or } \|\mathbf{T}_i - \mathbf{T}_{i-1}\| > \tau_t. \tag{9}$$

In addition, we render the visibility image from the current viewpoint and calculate the visibility probability for each pixel. If $\mathcal{V}_\rho \le \tau_\alpha$, the current pixel is considered to exhibit transparency, indicating that it has not yet accumulated a sufficiently stable 3-D Gaussian. Therefore, a 3-D Gaussian is initialized at the corresponding position only when this condition is met. This ensures accurate rendering of the pixel from the current viewpoint while avoiding the addition of redundant 3-D Gaussians at already stable positions. It improves computational efficiency and optimizes the rendering results.

The original 3DGS framework [14] initializes all the point clouds in a scene as 3-D Gaussians at once. In contrast, our approach uses a progressive initialization process. However, this strategy introduces a potential risk of prematurely pruning 3-D Gaussians due to limited observations from multiple viewpoints. To mitigate this issue, we add a keyframe delay module to adjust the training sequence of keyframes. Specifically, when a keyframe is added to our system, its point cloud is initialized as 3-D Gaussians. However, the images associated with this keyframe are not immediately used for 3-D Gaussian training. Instead, training begins only after the subsequent keyframe is introduced. This ensures that 3-D Gaussians can be trained with more views, thereby reducing the likelihood of premature pruning and improving the robustness.

D. Pyramid-Based Training

In our large-scale 3-D Gaussian scene representation, we adopt a progressive training approach to optimize the efficiency of training the 3-D Gaussian fields while maintaining rendering quality. Using color and depth images at varying resolutions, we construct pyramids of color and depth images, improving the training process by gradually refining the level of detail. Specifically, we partition the Gaussian map into multiscale representations to capture different levels of detail. The input color and depth images undergo repeated downsampling, allowing us to train the 3-D Gaussian progressively from coarse to fine resolution. During training, we prioritize low-resolution data to optimize coarse-level detail. After a certain number of iterations, we progressively reduce the level of downsampling, eventually using the original input resolution to complete the training. This method ensures efficient training while preserving the high-quality representation of the 3-D Gaussian scene at various levels of detail

$$\begin{aligned} l_0 &: \left(I_r^n, D_r^n, \mathrm{P}^n(I_{\mathrm{gt}}), \mathrm{P}^n(D_{\mathrm{gt}})\right) \\ l_1 &: \left(I_r^{n-1}, D_r^{n-1}, \mathrm{P}^{n-1}(I_{\mathrm{gt}}), \mathrm{P}^{n-1}(D_{\mathrm{gt}})\right) \\ &\vdots \\ l_n &: \left(I_r^0, D_r^0, \mathrm{P}^0(I_{\mathrm{gt}}), \mathrm{P}^0(D_{\mathrm{gt}})\right). \end{aligned} \tag{10}$$

Here, l denotes the level of the pyramid, I_r represents the rendered color image, and D_r denotes the rendered depth image. $P(I_{\mathrm{gt}})$ represents the color pyramid, while $P(D_{\mathrm{gt}})$ denotes the depth pyramid.

E. Gaussian Mapping

Upon receiving each keyframe, we initialize 3-D Gaussians. For the first frame, we process the entire point cloud, extracting the 3-D coordinates of the points as the centers of the 3-D Gaussians. We calculate the squared Euclidean distance of each point to the origin, ensuring a minimum value to prevent zero distances. Opacity parameters are initialized using an inverse Sigmoid function. For color information, we initialize a tensor to store features extracted from the point cloud's color data, with RGB channels corresponding to SHs coefficients. Although we use SHs, the initial SH degree is set to 0. As optimization iterations and keyframes increase, the SH degree gradually increases to better fit multiple perspectives, capped at 3.

We optimize each received keyframe once as a submap. Subsequently, in managing the keyframe sequence, upon receiving a new frame, we shuffle all the keyframes randomly and then select one frame for optimization at random. To ensure consistency in optimizing each keyframe and maintaining map integrity, we assign an upper limit to the number of optimization iterations for each keyframe. Keyframes that reach this limit are removed from the keyframe sequence.

We optimize the parameters of 3-D Gaussians, including rotation, scaling, density, and SH coefficients, by minimizing the image loss $\mathcal{L}_c$ and depth loss $\mathcal{L}_d$,

$$\mathcal{L} = \mathcal{L}_c + \lambda_d \mathcal{L}_d. \tag{11}$$

The image loss is composed of the illuminance error and the structural similarity (SSIM) error between two images

$$\mathcal{L}_c = (1-\lambda)\mathcal{L}_1(I, \mathcal{C}(\mathcal{G}, T_c)) + \lambda\mathcal{L}_{\text{ssim}}. \tag{12}$$

And depth loss is defined as $\mathcal{L}_1$ loss between rendered depth $\mathcal{D}$ and the LiDAR measurement's depth $\mathcal{D}_{\text{lidar}}$

$$\mathcal{L}_d = \sum \|\mathcal{D} - \mathcal{D}_{\text{lidar}}\|. \tag{13}$$

IV. Experimental Setup

A. Datasets

We conduct extensive experiments on nine datasets containing LiDAR–visual–inertial data. These include hku2(f0), LiDAR_Degenerate(f1), and Visual_Challenge(f2), which are derived from the FAST-LIVO [33] dataset, captured using two industrial cameras and a Livox Avia LiDAR. Additional datasets include hku_campus_seq_00(r0), degenerate_seq_00(r1), degenerate_seq_01(r2), degenerate_seq_02(m0), hku_campus_seq_01(m1), and hkust_campus_00(m2) from the R3LIVE [32] dataset, captured using a global shutter camera and a Livox Avia LiDAR. In particular, the m1, m2, and m3 sequences are truncated to the first 100 s to serve as test datasets. Among these, f1 and m0 represent indoor scenes, while the remaining datasets are outdoor scenes. All the images were set to a resolution of 640 × 512.

B. Implementation Details

We run our system on a desktop with Intel Core i7 13700K 2.50 GHz, a single NVIDIA GeForce RTX 4070 Ti, and 32-GB RAM. All our code is written in C++, with the 3-D Gaussian components relying on the LibTorch framework. Time-critical rasterization and gradient computations are executed using compute unified device architecture (CUDA).

C. Baseline Methods

We first conduct a comparative analysis of several existing SLAM methods based on NeRF and Gaussian splatting techniques, including NeRF-SLAM [22], MonoGS [15], SplaTAM [16], and Gaussian-LIC [21]. In addition, we evaluate popular open-source RGB- or RGB-D-based SLAM methods, such as MonoGS [15] and Photo-SLAM [18], as well as RGB-D-based methods, including SplaTAM [16] and Gaussian-SLAM [34], on a new dataset. Since our dataset lacks depth camera data, we project LiDAR point clouds onto the pixel plane for each frame to construct sparse pseudodepth maps, which were used for evaluating the RGB-D-based algorithms. Furthermore, most of the methods (except Photo-SLAM) experience bad estimations in the tracking modules. To address this, we use our tracking module's pose estimation to replace the pose input of these methods and focus solely on comparing the mapping performance.

D. Metrics

Quantitative measurements in terms of peak signal-to-noise ratio (PSNR), SSIM, and learned perceptual image patch similarity (LPIPS) are adopted to analyze the performance of photorealistic mapping. We also report the computing resources' requirements by showing the optimization time.

V. Evaluation

A. Quantitative Evaluation

In this section, we quantitatively evaluate the mapping performance of our LVI-GS based on the rendering results against the ground-truth images. Table II summarizes the quantitative performance of the baseline methods. Some results are derived from [21]. From these results, it is evident that NeRF-SLAM [22], despite achieving acceptable performance by incorporating additional depth information from Droid-SLAM [35], continues to focus on generating full-resolution images using neural implicit representations. In contrast, SplaTAM [16] emphasizes faster execution by using isotropic 3-D Gaussians to model scenes, intentionally disregarding view-dependent effects. While this optimization significantly enhances processing speed, it compromises visual quality and leads to performance degradation in complex, unbounded environments. Compared with other methods, our approach achieves an improvement in the quality of rendering images.

Table III shows comparison of several open-source RGB- and RGB-D-based methods across various datasets. Our experiments confirm similar trends: For RGB-D methods, such as SplaTAM [16] and Gaussian-SLAM [34], the initialization of 3-D Gaussians relies heavily on depth maps. The inherent sparsity of pseudodepth maps derived from LiDAR point clouds results in inaccurate Gaussian initialization, consequently leading to suboptimal mapping performance. MonoGS performs effectively in slower moving indoor scenes but exhibits reduced mapping quality as the scene size increases or motion speed escalates. Photo-SLAM [18], leveraging ORB-SLAM3 [31] for continuous feature point initialization, demonstrates relatively better metrics than other RGB- or RGB-D-based 3DGS SLAM methods. Our algorithm achieves the best rendering results compared with the aforementioned methods.

B. Qualitative Evaluation

In this section, we present the rendering images generated by the methods tested in Section V-A. Each method was run for the same number of iterations and rendering from identical viewpoints, facilitating a qualitative comparison to examine visual differences among the approaches. Comparisons to RGB-D-based 3DGS SLAM methods were omitted due to their limited rendering quality.

We also compare MonoGS [15], Photo-SLAM [18], our proposed method, and the ground truth, as illustrated in Fig. 2. Our observations show that MonoGS [15] performs adequately in indoor environments; however, it exhibits noticeable blurring in outdoor scenes. In contrast, our method demonstrates a marked improvement over Photo-SLAM [18] in restoring texture details on surfaces such as floors and walls. By leveraging

TABLE II
QUANTITATIVE PERFORMANCE COMPARISON OF DIFFERENT METHODS ON SEQUENCES HKU2(F0), LIDAR_DEGENERATE(F1), VISUAL_CHALLENGE(F2), HKU_CAMPUS_SEQ_00(R0), DEGENERATE_SEQ_00(R1), DEGENERATE_SEQ_01(R2)

Method	**Metrics**	**View**						**Avg.**
		f0	f1	f2	r0	r1	r2	
NeRF-SLAM [22]	PSNR↑	25.56	25.47	17.01	19.53	21.20	15.02	20.63
	SSIM↑	0.629	0.702	0.513	0.645	0.534	0.711	0.622
	LPIPS↓	0.281	0.272	0.512	0.455	0.364	0.328	0.369
MonoGS [15]	PSNR↑	23.58	23.45	17.04	16.19	15.03	15.03	18.39
	SSIM↑	0.699	0.762	0.671	0.490	0.512	0.457	0.599
	LPIPS↓	0.643	0.757	0.643	0.710	0.755	0.767	0.713
SplaTAM [16]	PSNR↑	25.51	27.40	17.84	17.10	19.24	18.30	20.90
	SSIM↑	0.709	0.798	0.652	0.526	0.666	0.597	0.658
	LPIPS↓	0.214	0.235	0.346	0.448	0.295	0.456	0.332
Gaussian-LIC [21]	PSNR↑	**29.89**	31.28	23.90	**25.27**	22.47	23.49	26.05
	SSIM↑	**0.820**	**0.840**	**0.830**	**0.805**	0.794	0.811	**0.817**
	LPIPS↓	**0.155**	0.178	**0.173**	**0.212**	0.192	0.187	0.183
Our LVI-GS	PSNR↑	27.90	**33.90**	**25.38**	24.47	**22.60**	**23.84**	**26.35**
	SSIM↑	0.731	0.826	**0.830**	0.776	**0.818**	**0.850**	0.805
	LPIPS↓	0.200	**0.043**	0.237	0.244	**0.159**	**0.170**	**0.176**

TABLE III
QUANTITATIVE PERFORMANCE COMPARISON OF RGB- OR RGB-D-BASED 3DGS SLAM SYSTEMS ON SEQUENCES DEGENERATE_SEQ_02(M0), HKU_CAMPUS_SEQ_01(M1), AND HKUST_CAMPUS_00(M2)

Metrics	**Method**				
	MonoGS [15]	SplaTAM [16]	Gaussian-SLAM [34]	Photo-SLAM [18]	Our LVI-GS
PSNR↑ (m0)	21.90	12.78	6.14	27.77	**28.99**
PSNR↑ (m1)	11.19	14.03	9.23	19.41	**20.17**
PSNR↑ (m2)	15.94	9.81	8.33	21.16	**23.93**
PSNR↑ Avg.	16.34	12.21	7.90	22.78	**24.36**
SSIM↑ (m0)	0.803	0.254	0.265	0.889	**0.899**
SSIM↑ (m1)	0.231	0.467	0.349	0.658	**0.728**
SSIM↑ (m2)	0.532	0.244	0.274	0.698	**0.784**
SSIM↑ Avg.	0.522	0.322	0.296	0.748	**0.804**
LPIPS↓ (m0)	0.804	0.860	0.907	0.311	**0.173**
LPIPS↓ (m1)	0.231	0.632	0.869	0.271	**0.211**
LPIPS↓ (m2)	0.532	0.827	0.925	0.378	**0.223**
LPIPS↓ Avg.	0.522	0.773	0.900	0.320	**0.202**

the denser spatial point clouds generated by LiDAR in texture-rich regions, our method achieves superior detail restoration within the same number of training iterations. Fig. 3 provides further examples of rendering details across four frames from sequence hkust_campus_00(m2) [32]. Even in challenging cases, such as scenes containing dense textures, glass surfaces, tree branches, and steps, our method consistently maintains high rendering quality.

In addition, we evaluate our LVI-GS in unstructured environments, as shown in Fig. 4, which depicts a forest scenario on the HKU campus. We select four different viewpoints to visualize and render the results. The rendered results for this sequence, compared with the ground truth, yield PSNR, SSIM, and LPIPS values of 19.04, 0.677, and 0.298, respectively. These results demonstrate the superior performance of our algorithm in unstructured scenarios, both quantitatively and qualitatively. They highlight the robustness of LVI-GS and its potential for real-world applications.

Furthermore, Fig. 5 demonstrates our mapping performance in an environment featuring dynamic objects, specifically highlighting a sequence in which a pedestrian moves through the scene. Rendered images from views A and B are compared against ground-truth images captured by the camera. The results indicate that our algorithm can mitigate the impact caused by dynamic objects, although minor ghosting artifacts persist. These artifacts arise from the inclusion of a keyframe containing the pedestrian during the training process. However, as the pedestrian moves out of view, subsequent training incorporates keyframes without them, thereby reducing the severity of the artifacts.

C. Ablation Study and Runtime Analysis

Table IV illustrates the impact of depth loss and different pyramid levels on training outcomes. For this ablation study, we selected a segment from sequence m1 that contains

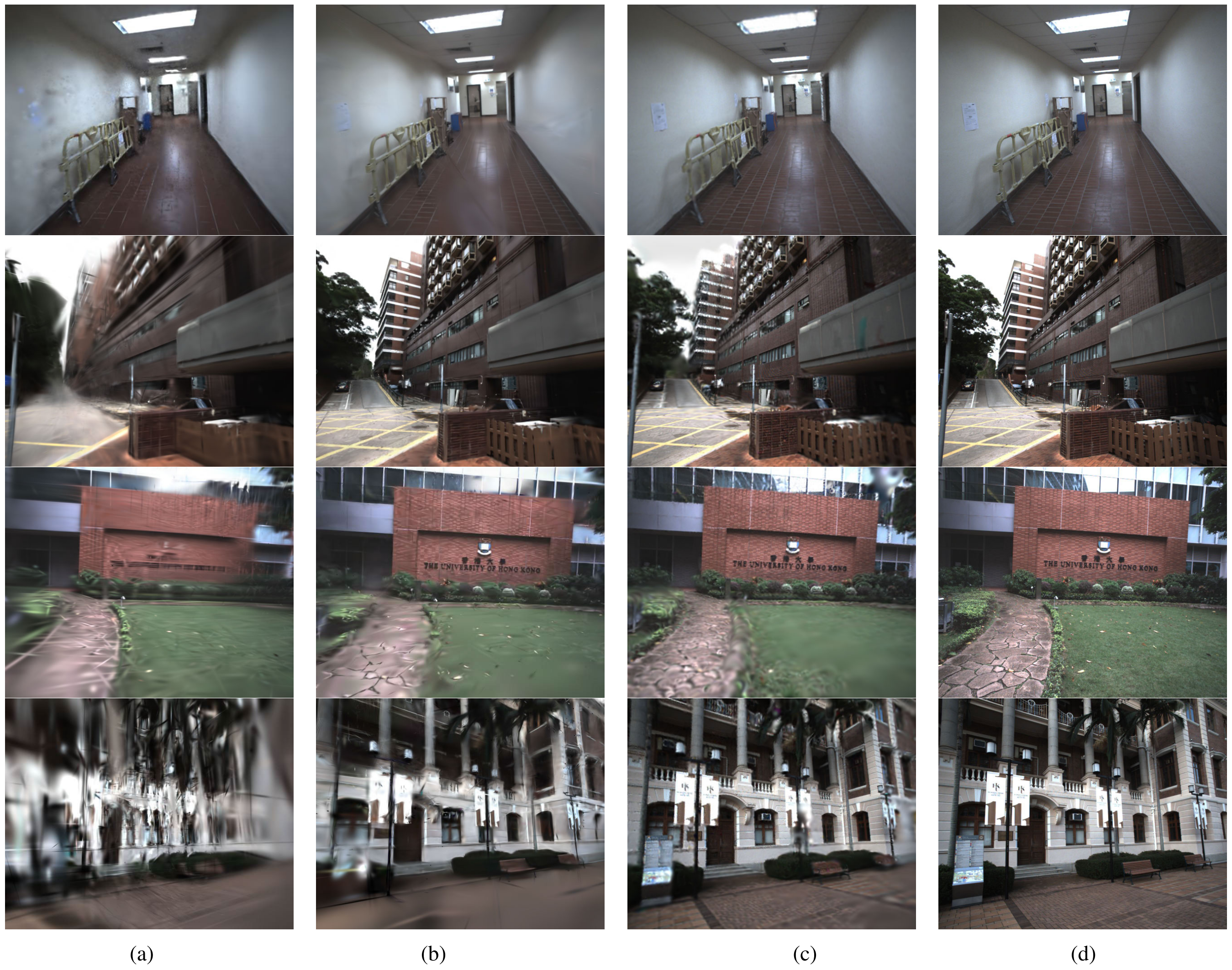

Fig. 2. Qualitative comparison of state-of-the-art 3DGS-based SLAM methods. (a) MonoGS [15]. (b) Photo-SLAM [18]. (c) Ours. (d) Ground truth.

TABLE IV
ABLATION STUDY ON THE EFFECT OF DEPTH LOSS AND PYRAMID-BASED TRAINING

Control Groups			Metrics		
#	Depth Loss	Pyramids	PSNR↑	SSIM↑	LPIPS↓
(1)	w/	w/o	20.40	0.706	0.318
(2)	w/	$l = 1$	21.77	0.732	0.271
(3)	w/	$l = 2$	**22.36**	**0.747**	**0.269**
(4)	w/	$l = 3$	18.91	0.632	0.475
(5)	w/o	$l = 2$	21.55	0.739	0.302

abundant point clouds and distant scenes to minimize interference from insufficient point clouds and limited depth variation. Our findings indicate that evaluation metrics improve as the pyramid level increases, reaching optimal performance at $l = 2$. We also conducted comparisons without depth loss and observed a modest improvement in rendering quality with its inclusion.

Alongside the quantitative results, Fig. 6 presents rendering images for visual comparison. When depth loss is omitted, quality degradation is evident in regions with significant depth variation. In addition, when the pyramid level l is set to 3, lower resolution pyramid images used in training lead to a substantial loss of scene detail, resulting in insufficiently trained fine-grained features.

We also conduct an ablation study to assess the impact of the keyframe selection and keyframe delay modules, as summarized in Table V. Disabling the keyframe selection module causes every frame passed to the mapping module to be directly used for training. Consequently, the point cloud of each frame is initialized as 3-D Gaussian points, leading to the repeated addition of 3-D Gaussians. This redundancy degrades rendering quality and significantly increases training time. Similarly, disabling the keyframe delay module may result in the erroneous pruning of certain 3-D Gaussians, negatively impacting the overall performance metrics.

In our runtime analysis, we monitor PSNR and SSIM metrics for a specific keyframe across iterations, as depicted in Fig. 7. Fig. 8 shows that photorealistic rendering quality is achieved after approximately 105 iterations, with a total runtime of about 3 s. In addition, we conduct a detailed time analysis on the sequences used in the ablation study, evaluating the time consumption per frame or iteration for

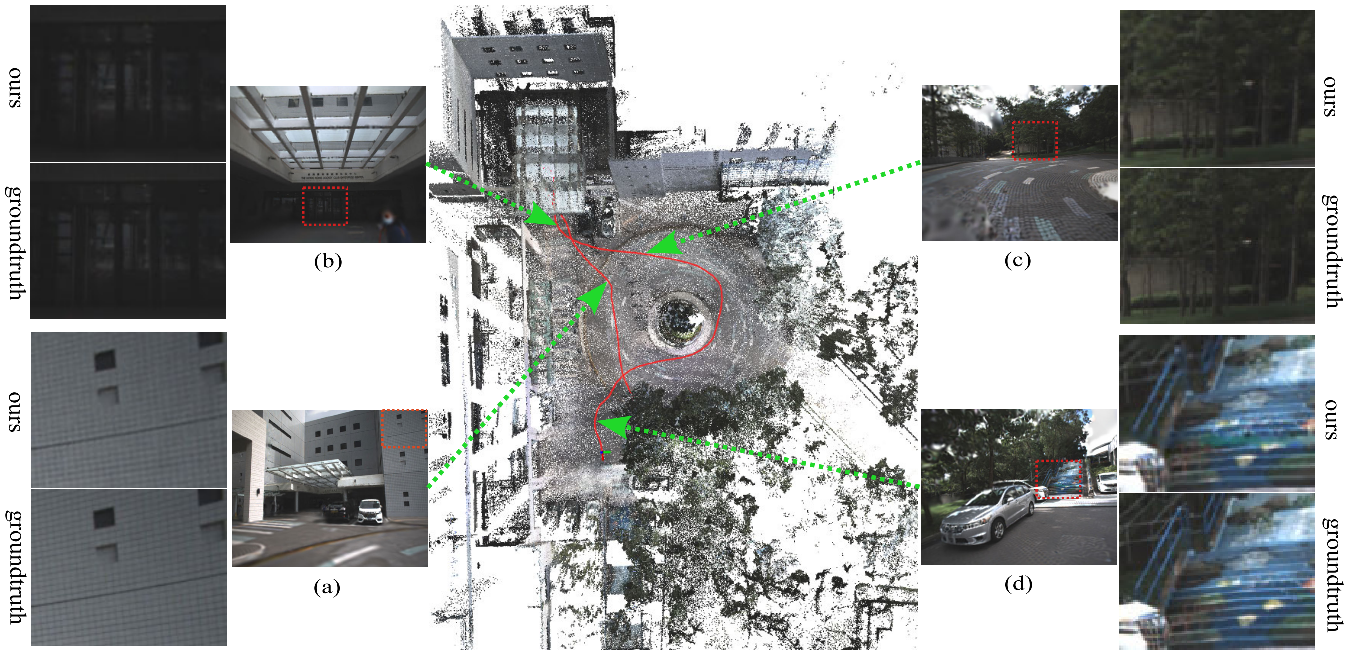

Fig. 3. Photorealistic mapping results of our LVI-GS on the hkust_campus_00(m2) sequence [32]. The red line represents the running trajectory, while (a)–(d) shows four selected rendered images at various positions.

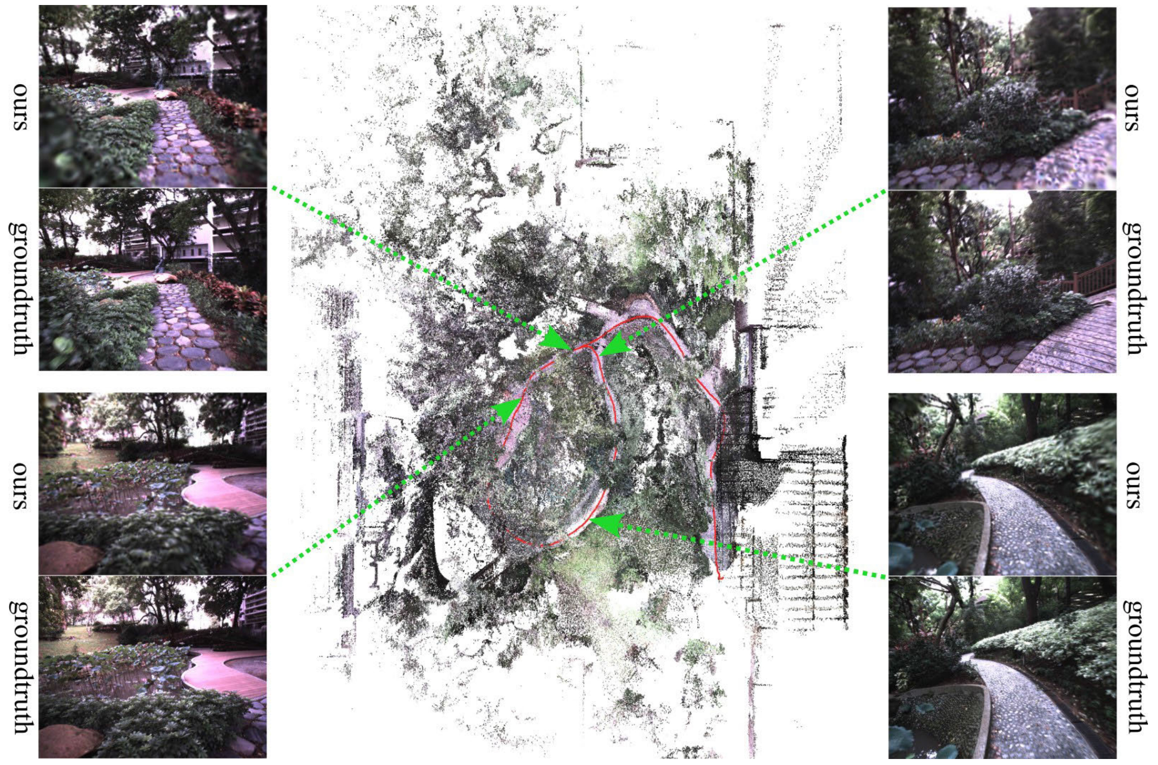

Fig. 4. Photorealistic mapping results of our LVI-GS in the unstructured environment of the hku_park_00 sequence [32]. The red line represents the estimated trajectory, and the rendering results at different locations are shown in the corners.

each component of our system, as summarized in Table VI. The results indicate that our algorithm achieves real-time performance on our high-performance GPU-equipped device.

D. Discussions and Limitations

Our method demonstrates robust performance. However, mapping quality can be further enhanced in challenging scenarios, such as poor lighting conditions or dynamic environments. The test results in Section V-A, which involve sequences with dynamic objects, indicate potential improvements through the incorporation of advanced image processing techniques. Specifically, future work can focus on more effectively identifying and managing dynamic objects to further improve 3-D scene reconstruction. This adaptation would

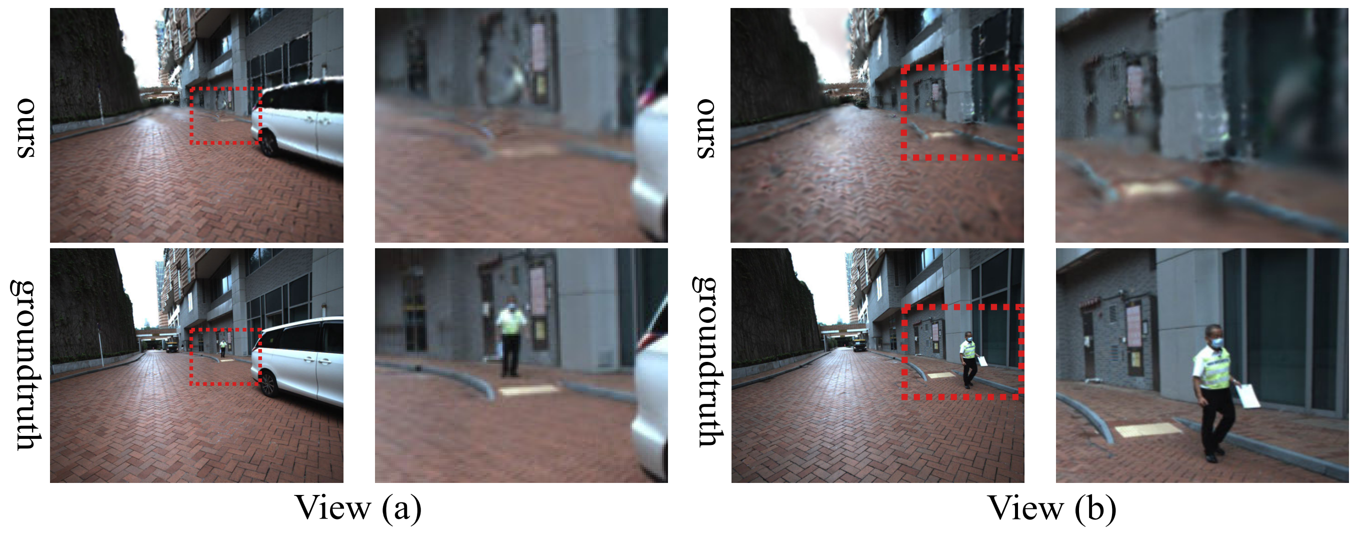

Fig. 5. Rendering result of our LVI-GS in the scene with a dynamic object. (a) Rendering result from viewpoint A. (b) Rendering result from viewpoint B.

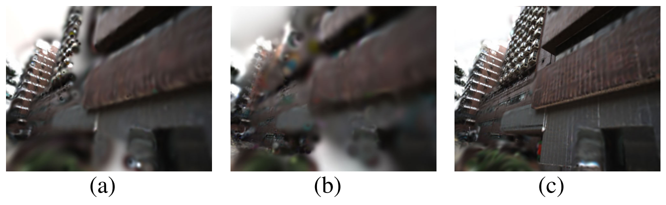

Fig. 6. Rendering results across different control groups. (a) $l = 2$ without depth loss. (b) $l = 3$ with depth loss. (c) $l = 2$ with depth loss.

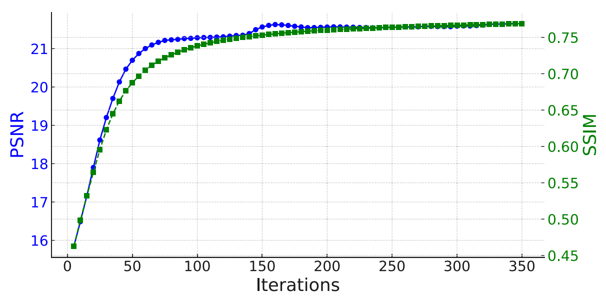

Fig. 7. Training metrics (PSNR and SSIM) over iterations for one keyframe.

TABLE V

IMPACT OF KEYFRAME SELECTION AND KEYFRAME DELAY ON PERFORMANCE

Configuration	PSNR↑	SSIM↑	LPIPS↓	Time (s)
w/ Keyframe selection and delay	**22.36**	**0.747**	**0.269**	100
w/o Keyframe selection	18.87	0.645	0.378	312
w/o Keyframe delay	22.31	0.711	0.270	100

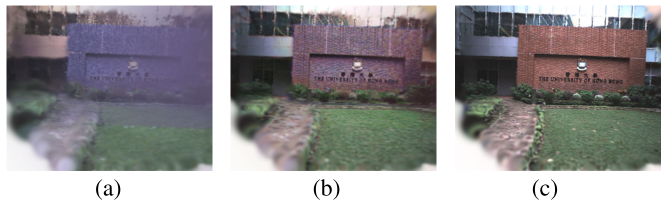

Fig. 8. Rendering images at different training iterations. (a) Iteration = 5. (b) Iteration = 30. (c) Iteration = 105.

TABLE VI

TIME CONSUMPTION OF SEVERAL KEY COMPONENTS OF OUR FRAMEWORK

Component	Time Cost (ms)
Odometry per-frame	9.42
Keyframe selection per-frame	0.098
Gaussian rendering per-frame	0.93
Pyramids processing per-frame	2.2
Optimization per-iteration	23.0

position the algorithm as an even more versatile solution for dynamic environments. In addition, as the map expands, memory consumption naturally increases. To optimize memory efficiency, future research could explore advanced strategies such as quantization and clustering. These approaches could enable the method to scale seamlessly while maintaining high performance in increasingly complex environments.

VI. CONCLUSION

In this article, we presented LVI-GS, a tightly coupled LiDAR–visual–inertial SLAM system using 3DGS for real-time, high-fidelity scene reconstruction and rendering. Our approach leveraged both LiDAR and image data, enabling the capture of precise geometric structures and detailed visual information, even in challenging outdoor environments. Our system achieved photorealistic mapping quality with notable computational efficiency through effective integration of Gaussian map expansion, keyframe management, thread management and CUDA-based acceleration.

Extensive experiments demonstrated that LVI-GS outperforms the existing state-of-the-art RGB- and RGB-D-based 3DGS SLAM systems, particularly in preserving rendering quality and efficiency across various complex scenarios. Our ablation studies further validated the benefits of pyramid-based training and depth loss for enhancing map representation accuracy.

References

[1] I. Abaspur Kazerouni, L. Fitzgerald, G. Dooly, and D. Toal, "A survey of state-of-the-art on visual SLAM," *Expert Syst. Appl.*, vol. 205, Nov. 2022, Art. no. 117734.

[2] J. Huang, S. Wen, W. Liang, and W. Guan, "VWR-SLAM: Tightly coupled SLAM system based on visible light positioning landmark, wheel odometer, and RGB-D camera," *IEEE Trans. Instrum. Meas.*, vol. 72, pp. 1–12, 2023.

[3] H. Li, B. Tian, H. Shen, and J. Lu, "An intensity-augmented LiDAR-inertial SLAM for solid-state LiDARs in degenerated environments," *IEEE Trans. Instrum. Meas.*, vol. 71, pp. 1–10, 2022.

[4] A. Elfes, "Using occupancy grids for mobile robot perception and navigation," *Computer*, vol. 22, no. 6, pp. 46–57, Jun. 1989.

[5] V. Reijgwart, A. Millane, H. Oleynikova, R. Siegwart, C. Cadena, and J. Nieto, "Voxgraph: Globally consistent, volumetric mapping using signed distance function submaps," *IEEE Robot. Autom. Lett.*, vol. 5, no. 1, pp. 227–234, Jan. 2020.

[6] J. Lin et al., "ImMesh: An immediate LiDAR localization and meshing framework," *IEEE Trans. Robot.*, vol. 39, no. 6, pp. 4312–4331, Dec. 2023.

[7] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, "NeRF: Representing scenes as neural radiance fields for view synthesis," *Commun. ACM*, vol. 65, no. 1, p. 99–106, Dec. 2021.

[8] E. Sucar, S. Liu, J. Ortiz, and A. J. Davison, "IMAP: Implicit mapping and positioning in real-time," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 6229–6238.

[9] Z. Zhu et al., "NICE-SLAM: Neural implicit scalable encoding for SLAM," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2022, pp. 12786–12796.

[10] X. Yang, H. Li, H. Zhai, Y. Ming, Y. Liu, and G. Zhang, "Vox-fusion: Dense tracking and mapping with voxel-based neural implicit representation," in *Proc. IEEE Int. Symp. Mixed Augmented Reality (ISMAR)*, Oct. 2022, pp. 499–507.

[11] H. Wang, J. Wang, and L. Agapito, "Co-SLAM: Joint coordinate and sparse parametric encodings for neural real-time SLAM," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2023, pp. 13293–13302.

[12] M. M. Johari, C. Carta, and F. Fleuret, "ESLAM: Efficient dense SLAM system based on hybrid representation of signed distance fields," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2023, pp. 17408–17419.

[13] B. Fei, J. Xu, R. Zhang, Q. Zhou, W. Yang, and Y. He, "3D Gaussian splatting as new era: A survey," *IEEE Trans. Vis. Comput. Graphics*, early access, May 7, 2024, doi: 10.1109/TVCG.2024.3397828.

[14] B. Kerbl, G. Kopanas, T. Leimkuehler, and G. Drettakis, "3D Gaussian splatting for real-time radiance field rendering," *ACM Trans. Graph.*, vol. 42, no. 4, pp. 1–14, Aug. 2023.

[15] H. Matsuki, R. Murai, P. H. J. Kelly, and A. J. Davison, "Gaussian splatting SLAM," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2024, pp. 18039–18048.

[16] N. Keetha et al., "SplaTAM: Splat, track & map 3D Gaussians for dense RGB-D SLAM," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2024, pp. 21357–21366.

[17] C. Yan et al., "GS-SLAM: Dense visual SLAM with 3D Gaussian splatting," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2024, pp. 19595–19604.

[18] H. Huang, L. Li, H. Cheng, and S.-K. Yeung, "Photo-SLAM: Real-time simultaneous localization and photorealistic mapping for monocular, stereo, and RGB-D cameras," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2024, pp. 21584–21593.

[19] S. Hong, J. He, X. Zheng, and C. Zheng, "LIV-GaussMap: LiDAR-inertial-visual fusion for real-time 3D radiance field map rendering," *IEEE Robot. Autom. Lett.*, vol. 9, no. 11, pp. 9765–9772, Nov. 2024.

[20] J. Cui et al., "LetsGo: Large-scale garage modeling and rendering via LiDAR-assisted Gaussian primitives," *ACM Trans. Graph.*, vol. 43, no. 6, pp. 1–18, Dec. 2024.

[21] X. Lang et al., "Gaussian-LIC: Real-time photo-realistic SLAM with Gaussian splatting and LiDAR-inertial-camera fusion," 2024, *arXiv:2404.06926*.

[22] A. Rosinol, J. J. Leonard, and L. Carlone, "NeRF-SLAM: Real-time dense monocular SLAM with neural radiance fields," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2023, pp. 3437–3444.

[23] J. Naumann, B. Xu, S. Leutenegger, and X. Zuo, "NeRF-VO: Real-time sparse visual odometry with neural radiance fields," *IEEE Robot. Autom. Lett.*, vol. 9, no. 8, pp. 7278–7285, Aug. 2024.

[24] S. Ha, J. Yeon, and H. Yu, "RGBD GS-ICP SLAM," in *Proc. Eur. Conf. Comput. Vis.*, Oct. 2024, pp. 180–197.

[25] C. Wu, Y. Duan, X. Zhang, Y. Sheng, J. Ji, and Y. Zhang, "MM-Gaussian: 3D Gaussian-based multi-modal fusion for localization and reconstruction in unbounded scenes," 2024, *arXiv:2404.04026*.

[26] L. C. Sun et al., "MM3DGS SLAM: Multi-modal 3D Gaussian splatting for SLAM using vision, depth, and inertial measurements," 2024, *arXiv:2404.00923*.

[27] Y. Zhang, F. Tosi, S. Mattoccia, and M. Poggi, "GO-SLAM: Global optimization for consistent 3D instant reconstruction," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2023, pp. 3727–3737.

[28] E. Sandström, Y. Li, L. Van Gool, and M. R. Oswald, "Point-SLAM: Dense neural point cloud-based SLAM," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2023, pp. 18387–18398.

[29] C.-M. Chung et al., "Orbeez-SLAM: A real-time monocular visual SLAM with ORB features and NeRF-realized mapping," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2023, pp. 9400–9406.

[30] A. Segal, D. Haehnel, and S. Thrun, "Generalized-ICP," in *Proc. Robot., Sci. Syst.*, Seattle, WA, USA, Jun. 2009, p. 435.

[31] C. Campos, R. Elvira, J. J. G. Rodríguez, J. M. M. Montiel, and J. D. Tardós, "ORB-SLAM3: An accurate open-source library for visual, visual–inertial, and multimap SLAM," *IEEE Trans. Robot.*, vol. 37, no. 6, pp. 1874–1890, Dec. 2021.

[32] J. Lin and F. Zhang, "R3LIVE: A robust, real-time, RGB-colored, LiDAR-Inertial-Visual tightly-coupled state estimation and mapping package," in *Proc. Int. Conf. Robot. Autom. (ICRA)*, May 2022, pp. 10672–10678.

[33] C. Zheng, Q. Zhu, W. Xu, X. Liu, Q. Guo, and F. Zhang, "FAST-LIVO: Fast and tightly-coupled sparse-direct LiDAR-inertial-visual Odometry," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2022, pp. 4003–4009.

[34] V. Yugay, Y. Li, T. Gevers, and M. R. Oswald, "Gaussian-SLAM: Photo-realistic dense SLAM with Gaussian splatting," 2023, *arXiv:2312.10070*.

[35] Z. Teed and J. Deng, "DROID-SLAM: Deep visual SLAM for monocular, stereo, and RGB-D cameras," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 34, 2021, pp. 16558–16569.