

GS-LIVO: Real-Time LiDAR, Inertial, and Visual Multisensor Fused Odometry With Gaussian Mapping

Sheng Hong^{1b}, Chunran Zheng^{1b}, *Member, IEEE*, Yishu Shen^{2b}, Changze Li^{2b}, *Student Member, IEEE*, Fu Zhang^{1b}, *Senior Member, IEEE*, Tong Qin^{1b}, *Member, IEEE*, and Shaojie Shen^{1b}

Abstract—In recent years, 3-D Gaussian splatting (3D-GS) has emerged as a novel scene representation approach. However, existing vision-only 3D-GS methods often rely on hand-crafted heuristics for point-cloud densification and face challenges in handling occlusions and high graphics processing unit (GPU) memory and computation consumption. Light detection and ranging (LiDAR)-inertial-visual sensor configuration has demonstrated superior performance in precise localization and dense mapping by leveraging complementary sensing characteristics: rich texture information from cameras, precise geometric measurements from LiDAR, and high-frequency motion data from inertial measurement unit. Inspired by this, we propose a novel real-time Gaussian-based simultaneous localization and mapping system. Our map system comprises a global Gaussian map and a sliding window of Gaussians, along with an iterative error state Kalman filter (IESKF)-based real-time odometry utilizing Gaussian maps. The structure of the global Gaussian map consists of hash-indexed voxels organized in a recursive octree. This hierarchical structure effectively covers sparse spatial volumes while adapting to different levels of detail and scales in the environment. The Gaussian map is efficiently initialized through multisensor fusion and optimized with photometric gradients. Our system incrementally maintains a sliding window of Gaussians with minimal graphics memory usage, significantly reducing GPU computation and memory consumption by only optimizing the map within the sliding window, enabling real-time optimization. Moreover, we implement a tightly coupled multisensor fusion odometry with an IESKF, which leverages real-time updating and rendering of the Gaussian map to achieve competitive localization accuracy. Our system represents the first real-time Gaussian-based SLAM framework deployable on resource-constrained embedded systems (all implemented in C++/CUDA for efficiency), demonstrated on the *NVIDIA Jetson Orin NX* platform. The framework achieves real-time performance

while maintaining robust multisensor fusion capabilities. All implementation algorithms, hardware designs, and CAD models and demo video of our GPU-accelerated system will be publicly available.

Index Terms—Gaussian splatting, multisensor fusion, odometry.

NOMENCLATURE

Notation	Explanation.
$W(\cdot)$	Vector expressed in the world frame.
$C(\cdot)$	Vector expressed in the camera frame.
$I(\cdot)$	Vector expressed in the IMU frame.
$L(\cdot)$	Vector expressed in the LiDAR frame.
$SE(3)$	Special Euclidean group in 3-D spaces.
${}^L T_I \in SE(3)$	LiDAR-to-IMU extrinsic.
${}^I T_C \in SE(3)$	IMU-to-camera extrinsic.
${}^W T_I \in SE(3)$	IMU pose w.r.t. world.
${}^W T_C \in SE(3)$	Camera pose w.r.t. world.
$\mathcal{N}({}^W \mathbf{p}_i, \Sigma_{3D})$	3-D Gaussian defined in world coordinates.
$\mathcal{N}(\mathbf{q}_i, \Sigma_{2D})$	2-D Gaussian after projection to image plane.
${}^W \mathbf{p}_i, \mathbf{q}_i$	Gaussian mean in 3-D / 2-D.
Σ_{3D}, Σ_{2D}	Gaussian covariance in 3-D / 2-D.
${}^W \mathbf{R}_i$	Gaussian orientation (rotation) matrix in 3-D.
${}^W \mathbf{n}_i$	Normal vector of the objects surface.
$S(\cdot)$	Scaling matrix indicating the spatial extent along axes.
σ_i	Opacity of the i th Gaussian.
$\hat{\mathbf{I}}(\cdot), \mathbf{I}(\cdot)$	Rendered image / Captured image.
$\mathbf{v}_s$	Root voxel length in the spatial structure.
$\pi(\cdot)$	Projection model of the pinhole camera.
$\mathbf{J}_\pi$	Jacobian matrix of the projection model.
${}^W T_C^*$	Optimized camera pose after optimization.
$\theta_{k-1}, \theta_{k-1}^*$	Gaussian parameters before/after optimization.

I. INTRODUCTION

IN RECENT years, advances in simultaneous localization and mapping (SLAM) have led to a variety of explicit map representations, including dense colored point clouds, sparse patch-based structures [7], [10], and even mesh-based [11], [12], [13], [14], [15], [16] or surfel-based [17], [18], [19], [20], [21] reconstructions. These forms, often integrated with feature-based or direct methods, support efficient, real-time

Received 10 April 2025; accepted 8 May 2025. Date of publication 24 June 2025; date of current version 14 July 2025. This article was recommended for publication by Associate Editor A. Valada and Editor J. Civera upon evaluation of the reviewers' comments. (Sheng Hong and Chunran Zheng Contributed equally to this work.) (Corresponding author: Tong Qin.)

Sheng Hong and Shaojie Shen are with the Department of Electronic Computer Engineering, The Hong Kong University of Science and Technology, Hong Kong.

Chunran Zheng and Fu Zhang are with the Department of Mechanical Engineering, The University of Hong Kong, Hong Kong.

Yishu Shen, Changze Li, and Tong Qin are with the Global Institute of Future Technology, Shanghai Jiao Tong University, Shanghai 200240, China (e-mail: qintong@sjtu.edu.cn).

Data is available on-line at <https://github.com/HKUST-Aerial-Robotics/GS-LIVO>.

This article has supplementary downloadable material available at <https://doi.org/10.1109/TRO.2025.3582809>, provided by the authors.

Digital Object Identifier 10.1109/TRO.2025.3582809

operation across platforms such as aerial robotics and mobile robots [2], [3], [4], [5], [7], [22], [23]. Many state-of-the-art SLAM systems leverage these classical map constructs due to their well-established pipelines and robust performance in pose estimation tasks. However, while such handcrafted, explicit representations have matured substantially, certain limitations remain. They typically rely on abundant geometric features and high-frame-rate inputs to ensure stable tracking. Moreover, these methods often struggle to provide photorealistic reconstructions and are generally confined to explaining only the observed parts of a scene. This shortfall poses challenges in applications that require the prediction or synthesis of new viewpoints, such as immersive augmented reality, high-quality 3-D modeling, and scenarios where unseen regions must be inferred for robust decision making. Recently, breakthroughs in novel view synthesis have introduced neural representations capable of photorealistic rendering from arbitrary viewpoints. Implicit models such as neural radiance fields (NeRF) [24] and explicit structures such as 3D Gaussian splatting (3D-GS) [25], [26] not only enrich the fidelity of the reconstructed environment but also open the door to more advanced SLAM paradigms. A wave of NeRF-based SLAM approaches [27], [28], [29], [30], [31], [32] and 3D-GS-based methods [33], [34], [35], [36], [37] seek to integrate these high-fidelity representations into the SLAM pipeline, aiming to exploit the richer photometric and geometric cues for more accurate localization and mapping. Intuitively, as higher quality maps provide better spatial and appearance cues, they should enhance the accuracy and robustness of pose estimation, thereby reinforcing the reciprocal relationship between mapping and localization within SLAM. Despite their promise, current neural-based SLAM systems face a key bottleneck: maintaining truly real-time map updates. Although some approaches achieve real-time odometry, their map optimization and refinement processes often lag behind, relying on separate, slower threads [28], [33], [35]. This mismatch between fast pose estimation and slower map updating reduces the system's adaptability, particularly in dynamic or large-scale environments where continuously refreshed, high-fidelity maps are essential. As a result, the practical deployment of these high-quality representations in robotics—where rapid environmental interpretation is paramount—remains limited.

Motivated by these issues, this work aims to develop a light detection and ranging (LiDAR)-inertial-visual odometry (LIVO) system that tightly integrates LiDAR and camera measurements using a novel Gaussian map representation. The objectives of this work are twofold: achieving high-precision localization through tightly coupled multisensor fusion within a Gaussian map, and significantly improving the efficiency of Gaussian map updates, even in large-scale (as exemplified by the aerial scene shown in Fig. 1) and complex environments. This approach addresses key bottlenecks that hinder the practical deployment of current Gaussian-SLAM systems.

In summary, our key contributions include the following.

- 1) We introduce a global Gaussian map representation structured as a spatial hash-indexed octree. This hierarchical structure enables efficient global indexing,

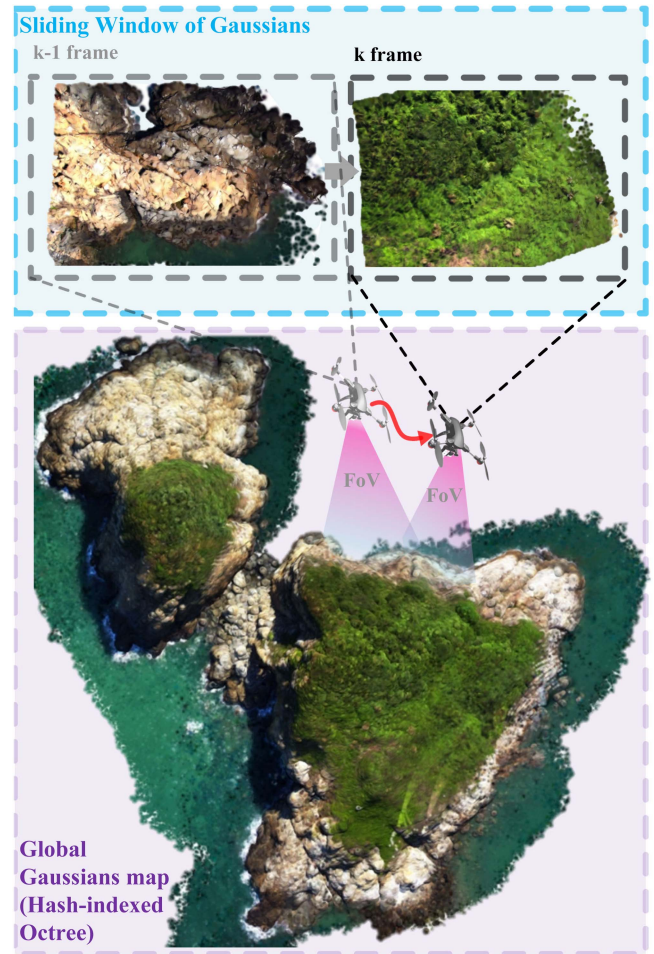


Fig. 1. Components of GS-LIVO for large-scale scenarios: Real-time odometry and Gaussian mapping on aerial datasets [9].

covisibility checks, and inherently supports varying levels of detail (LoD) to accommodate large-scale scenes (see Section II-A).

- 2) We propose a fast initialization strategy that fuses LiDAR and visual data to rapidly generate a well-structured global Gaussian map with high-fidelity rendering (see Section II-B)
- 3) We propose a sliding window of Gaussians to incremental method scheduling the Gaussians, which can minimize the cost in maintaining the map and computation burden optimize of Gaussians and reduce GPU memory burden (see Section II-C)
- 4) We propose a novel visual measurement model leveraging the photorealistic rendering capabilities of our Gaussian map, tightly integrating LiDAR-inertial measurements using an iterative error state Kalman filter (IESKF) with sequential updates (see Section II-D).

Extensive benchmark and real-world experiments (see Section III) show that our method significantly reduces memory usage and accelerates Gaussian map optimization, while achieving competitive odometry accuracy and maintaining high rendering quality across both indoor and outdoor datasets.

A. NeRF-Based SLAM

Recent advancements in SLAM technology have seen significant developments in the use of NeRFs for implicit map representations. Implicit mapping and positioning (iMAP) [27] pioneered the application of implicit maps in SLAM systems, marking a notable milestone despite not outperforming traditional visual odometry methods. Building on iMAP's foundational ideas, neural implicit scalable encoding (NICE)-SLAM [28] introduced an open-source solution with enhanced scalability. It employs multiscale multilayer perceptrons to model geometric structures at various scales, utilizing error-guided probability for pixel-based frame sampling, which significantly improves efficiency. Furthermore, instant neural graphics primitives (Instant-NGP) [38] addresses the computational intensity of NeRF networks by introducing innovative position encoding, reducing network size, and enhancing overall performance through effective multiresolution hashing.

Joint coordinate and sparse parametric encodings for neural real-time SLAM (Co-SLAM) [29] leverages map representations to learn spatial geometric structures across different frequencies, thereby enhancing both mapping and localization precision. Efficient dense SLAM (ESLAM) [30], similar to NICE-SLAM, focuses on a prior-free training model and uses signed distance function maps and carefully designed loss functions to reduce memory consumption from cubic to quadratic growth using axis-aligned planes. Uncertainty learning for dense neural SLAM (UncLeSLAM) [31] builds upon NICE-SLAM by modeling pixel uncertainty through Laplacian uncertainty, which, although slightly increasing computational time and memory usage, significantly improves accuracy via self-supervised Laplacian uncertainty modeling.

Finally, Orbeez-SLAM [32] integrates the traditional oriented fast and rotated BRIEF (ORB)-SLAM [39] with a NeRF map from Instant-NGP, demonstrating the successful fusion of classical and modern implicit map representations. More recent works, such as H₂-Mapping and H₃-Mapping [40], [41], further push the limits of real-time capability and high-quality reconstruction on edge devices by introducing hierarchical hybrid representations, improved initialization schemes, and advanced keyframe selection strategies. Similarly, Swift-Mapping [42] employs a neural implicit octomap structure to achieve efficient neural representation of large, dynamic urban scenes, enabling online updates and significantly accelerating reconstruction speeds.

These contributions collectively showcase the potential of NeRF-based SLAM in enhancing the robustness, efficiency, and accuracy of 3-D scene reconstruction and localization. However, while NeRF-based methods effectively reduce memory consumption through implicit representations, they often require substantial optimization time, making it challenging to achieve high-fidelity, high-frame-rate mapping updates.

B. 3D-GS in SLAM

3D-GS [25] is an explicit representation method for scene modeling and rendering, which utilizes 3-D Gaussian to depict the geometric structure and appearance of a scene. This approach

excels in novel view synthesis and real-time rendering, significantly reducing parameter complexity compared to traditional representations such as meshes or voxels. To further enhance the efficiency and scalability of 3-D Gaussian representations, some studies have focused on compression techniques. For instance, Motion-GS [43] and RTG-SLAM [44] achieve large-scale 3-D reconstruction by employing compressed Gaussian maps, providing effective representations more suitable for real-time applications. LoD techniques are also utilized to manage the complexity of rendering large 3-D scenes. Octree-GS [45] organizes data into a hierarchical structure, supporting multiresolution anchors that ensure stable rendering speeds from different viewpoints. The level of Gaussians [46] employs a tree-based hierarchical structure, dynamically selecting appropriate detail levels based on the observer's distance and viewpoint, effectively allocating resources while preserving details in close-up views. These LoD techniques efficiently address computational bottlenecks encountered when rendering extensive and complex 3-D scenes, making them highly suitable for real-time and large-scale applications.

Moreover, to improve the initialization results of 3D Gaussian Splatting (3D-GS) systems, several works based on LiDAR point clouds have been proposed. LiDAR-inertial-visual (LIV)-GaussMap [37] and Gaussian-LIC (LiDAR-Inertial-Camera) [47] are among the earliest methods that utilize LiDAR for initializing the structure of maps, providing prior pose estimation, and further optimizing the map using photometric gradients. LiV-GS (LiDAR-inertial-Visual Gaussian Splatting) [36] further incorporates constraints such as normal vector loss to optimize the map. LetsGo [48] introduces a handheld polar coordinate scanner for capturing RGB-D data of large parking environments, combined with LiDAR-assisted Gaussian primitives, achieving high-quality large-scale garage modeling and rendering. LI-GS (LiDAR-Inertial Gaussian Splatting) [49], on the other hand, enhances geometric accuracy in large-scale scenes by converting LiDAR data into plane-constrained multimodal Gaussian mixture models (GMMs). This method uses GMMs during both the initialization and optimization stages to ensure sufficient and continuous supervision over the entire scene, thereby mitigating the risk of overfitting.

In recent years, 3D-GS has found widespread application as a mapping representation method in the field of SLAM [33], [34], [35], [50], [51], [52]. 3D-GS has demonstrated significant improvements in real-time performance, map rendering, iterative updates, and partial support for dense RGB-D SLAM. For example, PhotoSLAM [51] uniquely combines explicit geometric and photometric features, integrating the characteristics of Oriented FAST and Rotated BRIEF (ORB)-SLAM [39] with traditional techniques such as Gaussian pyramid representation and superpixels, emphasizing photorealistic map construction. Mono-GS [52] integrates 3-D Gaussian representation with a real-time differentiable splatting pipeline, introducing depth loss and manual pose Jacobian to improve pose estimation accuracy. GS-SLAM [34] leverages 3-D Gaussian representation to balance efficiency and accuracy, achieving scene reconstruction and robust pose tracking through adaptive expansion strategies, with a notable speedup compared to neural implicit methods.

SplaTAM [33] adopts a contour-guided optimization approach, dynamically expanding the map capacity based on rendered contours and input depth, ensuring efficient and accurate map updates in dynamic environments. Multi-Modal 3D Gaussian Splatting MM3D-GS SLAM [35] proposes a multimodal SLAM framework that utilizes visual, inertial, and depth measurement data, achieving efficient scene reconstruction and real-time rendering through 3-D Gaussian representation.

However, most existing 3D-GS-based SLAM solutions focus on real-time odometry rather than maintaining high-frequency, real-time map updates, causing map construction to lag behind sensor acquisition. Such low update rates critically limit a robot's ability to swiftly interpret its surroundings and navigate dynamic or large-scale environments. In contrast, our approach achieves consistently high-frequency map updates—over 10-Hz indoors on a single CPU thread and around 3-Hz outdoors—thereby setting a new benchmark for 3D-GS-based SLAM and significantly enhancing its practical utility in real-world robotic applications.

C. LIV Multisensor Fusion in SLAM

In the field of SLAM, recent advancements have significantly propelled the development of multisensor fusion, particularly in the integration of LiDAR, inertial measurement units (IMUs), and visual sensors. One pioneering work in this domain is LIC-Fusion (LiDAR-Inertial-Camera) [53], which effectively combines LiDAR, IMU, and visual data to enhance the overall performance of SLAM systems. By leveraging the complementary characteristics of these sensor modalities, LIC-Fusion establishes a robust foundation for subsequent research.

Building upon this groundwork, CamVox [54] further refines the integration process, providing a more resilient and efficient solution for real-time SLAM. This system capitalizes on the unique strengths of each sensor type, achieving superior accuracy and reliability through an optimized fusion strategy that minimizes redundancy while maximizing information gain.

Subsequently, LVI-SAM [55] introduces a tightly coupled approach that integrates a visual-inertial system with a LiDAR-inertial system. Utilizing factor graphs and sliding window optimization, LVI-SAM significantly enhances the robustness and precision of the SLAM system, making it well-suited for challenging environments where high accuracy and reliability are critical. The framework's ability to handle large-scale mapping tasks under dynamic conditions highlights its versatility and effectiveness.

R^2 LIVE [4] advances the state-of-the-art by employing an error-state iterated Kalman filter to reduce feature reprojection errors. This technique ensures real-time performance and robustness, even in highly dynamic and complex scenarios. By addressing the challenges associated with sensor synchronization and drift, R^2 LIVE demonstrates improved consistency and stability in the estimated trajectories.

R^3 LIVE [5] represents another significant step forward, introducing a novel method based on photometric error for constructing dense point cloud maps. Unlike traditional feature-based approaches that heavily rely on prominent visual features, R^3 LIVE reduces computational cost by selecting points with larger gradients. This method reduces feature extraction

computations but is prone to local optima due to point-based photometric errors and requires maintaining a dense colored global map continuously.

FAST-LIVO [7], [8], compared with the colored point-based map maintained by R^3 LIVE, employs a unified surfel-based map composed of adaptive-sized planes that integrate both LiDAR and visual map points. Sparse visual points are attached with image patches, enabling patch-based photometric errors with superior convergence properties. Inspired by its map structure, we design our own global Gaussian map. However, our Gaussian map features a higher density of map points to support dense view synthesis. To maintain efficiency and prevent excessive memory usage, we adopt LoD techniques and a Gaussian map sliding mechanism.

II. METHODOLOGY

The system overview of GS-LIVO is illustrated in Fig. 2. The hardware configuration integrates synchronized LiDAR, IMU, and camera, with precise temporal alignment ensured via an emulated pulse-per-second signal [7], [8], [56]. The software framework comprises four key modules:

- 1) a global Gaussian map organized with a spatial hash-indexed octree that effectively covers sparse spatial volumes while adapting to various environmental details and scales (see Section II-A);
- 2) rapid initialization and online optimization of Gaussians based on LiDAR and visual information with photometric gradients (see Section II-B);
- 3) incremental maintenance of sliding windows of local Gaussians for real-time optimization with minimal graphical memory usage (see Section II-C);
- 4) pose optimization using an IESKF with sequential updates (see Section II-D).

Our system represents a real-time SLAM framework that seamlessly integrates LiDAR, inertial, and visual sensors to achieve competitive localization accuracy. For clarity of presentation, we first define the notations used throughout this section in the Nomenclature, which summarizes the coordinate transformations, Gaussian attributes, spatial structures, and optimization parameters.

A. Global Gaussian Map: Hash-Indexed Octree

Fig. 3 illustrates the structure of the global map and the sliding strategy for Gaussian within our system. The mapping system consists of two main components: the global Gaussian map and the sliding windows of Gaussians. The global Gaussian map utilizes a hash-indexed octree structure that employs spatial hashing to efficiently cover the sparse volumes of the scene. This structure can recursively subdivide regions based on environmental complexity, enabling a more detailed and fine-grained map representation. The indexing of root voxels is determined based on spatial hash keys, calculated as follows:

$$\text{HashKey} = \left\lfloor \frac{w \mathbf{p}_i}{\mathbf{v}_s} \right\rfloor \quad (1)$$

where $\mathbf{v}_s$ is the length of the root voxel. This approach allows for efficient management and indexing of sparsely distributed

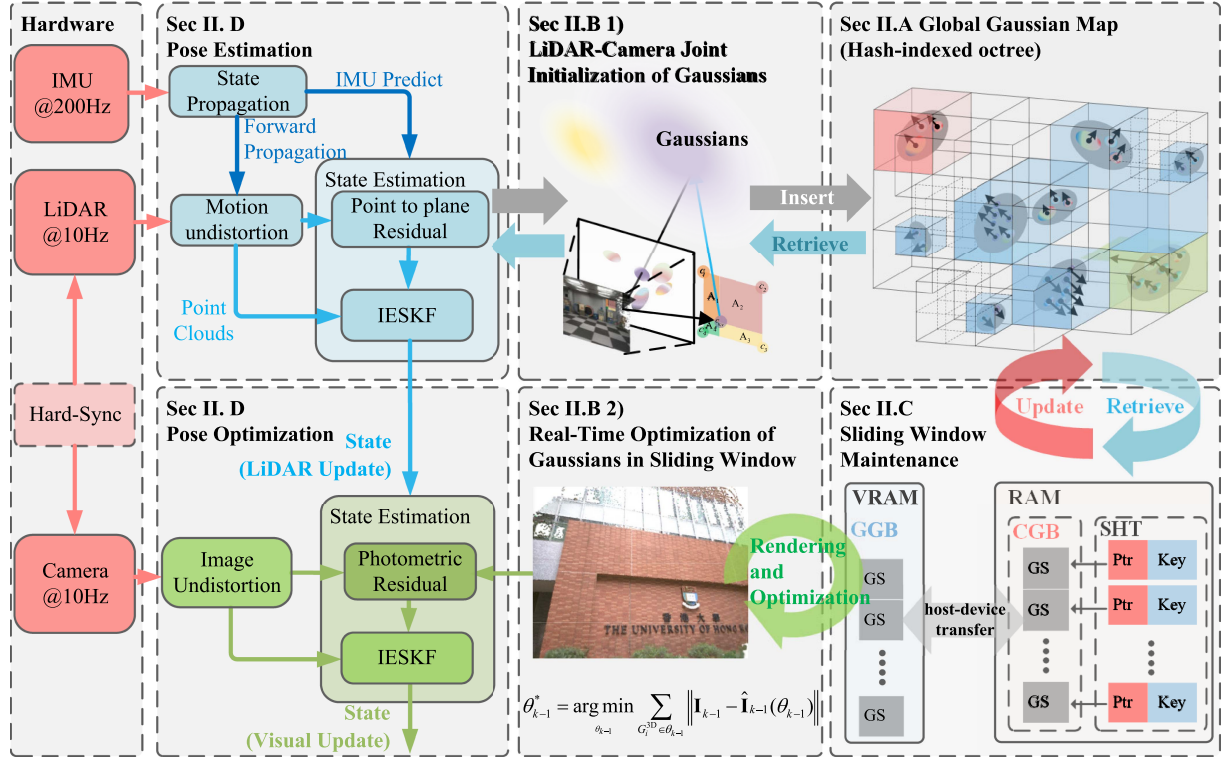


Fig. 2. System overview of GS-LIVO: A real-time LIVO system with Gaussian splatting-based mapping. The pipeline performs joint initialization and optimization of Gaussians using multisensor data, managed through a hash-indexed octree structure and sliding window mechanism.

data points in large-scale environments. The floor function $\lfloor \cdot \rfloor$ denotes rounding down to the nearest integer, ensuring that voxels are properly aligned in the discrete spatial grid.

In our LIV system, we use the LiDAR data from the current frame to compute the spatial key, thereby efficiently identifying the root voxels in the global Gaussian map that correspond to the current field-of-view (FoV). This method facilitates the identification of covisibility associations within the spatial environment, determining which areas are observable from the current viewpoint, thus providing accurate map information for subsequent processing. However, due to the noncontiguous nature of hash-indexed storage, direct GPU processing of Gaussian parameters for parallel optimization is constrained. A noncontiguous memory layout leads to inefficiencies in GPU access and processing, introducing unnecessary latency.

To address this issue, we have designed a specialized sliding window of Gaussian specifically for maintaining the Gaussian within the FoV, with a contiguous memory layout to ensure efficient GPU processing of Gaussian parameters. Specifically, the entire Gaussian of the large-scale environment is stored in random access memory (RAM) using a noncontiguous hash-octree structure in the global Gaussian map, while only the Gaussian located within the FoV are stored in a contiguous RAM region, with a duplicate kept in video random access memory (VRAM). When optimization of Gaussian parameters is required, these parameters are transferred from RAM to VRAM, allowing the GPU to perform parallel optimization efficiently. Upon completion of the optimization process, the updated Gaussian is transferred back from VRAM to RAM to maintain consistency across the global map.

Unlike the typically limited and nonscalable graphics memory, RAM provides greater capacity and can be easily expanded via swap space, enabling the handling of larger and more complex scenes. This design not only enhances the efficiency of GPU processing but also ensures the stability and reliability of the system when dealing with large-scale datasets, as shown in Fig. 3.

B. Initialization and Optimization of Gaussians

When a new LiDAR and camera frame is received, voxel-based map down-sampling on dense LiDAR point is implemented to mitigate GPU memory consumption. Unlike others such as [33], [34], [52] that use boundaries for selection, we employ the leaf nodes voxels of the octree, to sample the object's surface in 3-D space. By selecting LiDAR points within each voxel, the scene is represented more efficiently.

If the leaf voxel is not filled to capacity, new Gaussians will be roughly initialized with LiDAR and camera, as shown in Fig. 2, and inserted into the leaf voxel.

1) *LiDAR-Camera Joint Initialization of Gaussians*: In this step, the structural parameters are initialized with LiDAR. Specifically, the scaling matrix $\mathbf{S}$ for the Gaussians is initialized based on the level of the hash-voxel, expressed as follows:

$$\mathbf{S}_i(\mathbf{s}) = \begin{pmatrix} s_\delta & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & s_z \end{pmatrix}. \quad (2)$$

The scaling parameters s_y and s_z are set based on the size of the leaf node voxel, ensuring the Gaussian structure aligns

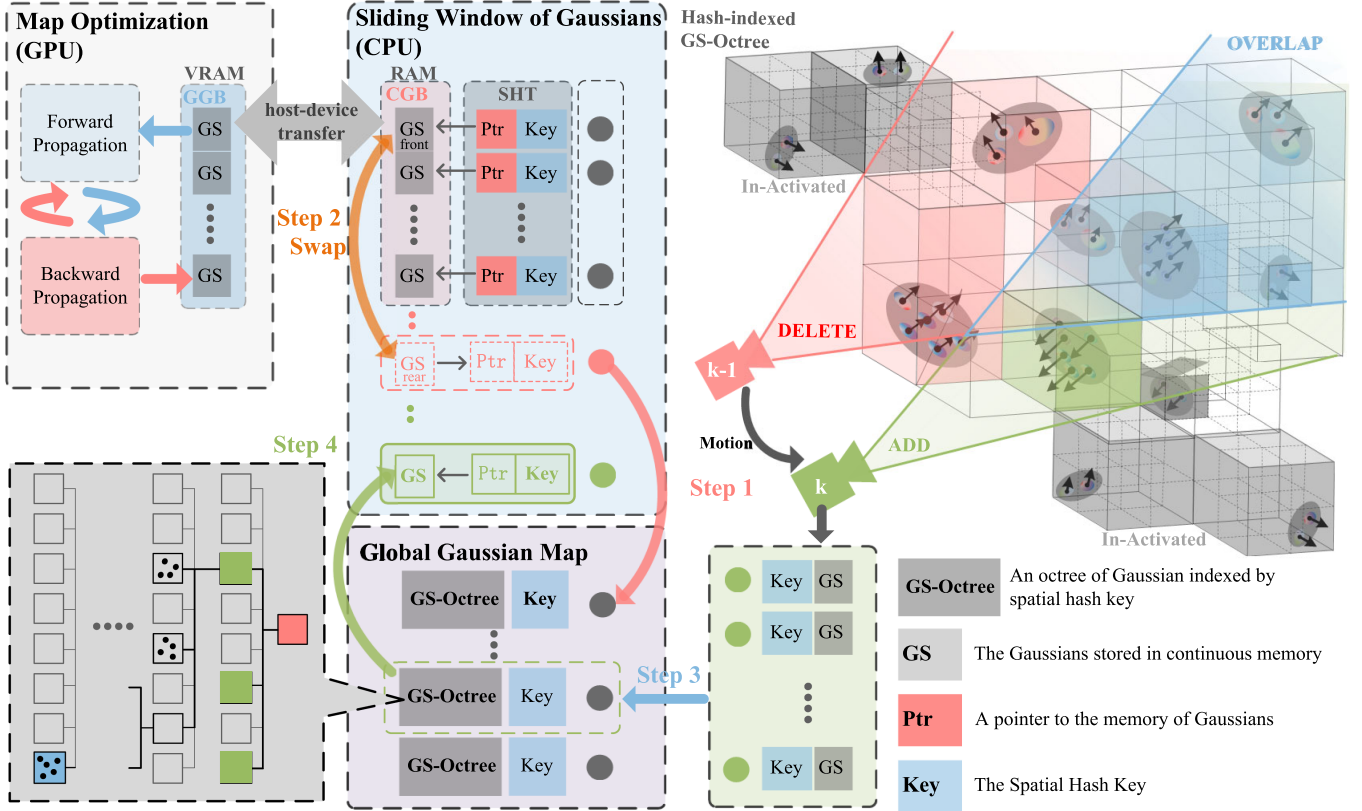


Fig. 3. Overview of the procedures for incrementally updating the sliding window of Gaussians (detailed in Section II-C).

with the object surface in 3-D. The parameter s_δ is a small value used to define a thin slice along the plane, making the Gaussian behave like a 2-D planar surface attached to the object's surface.

The rotation matrix of the Gaussian is initialized using the normal vector $^W \mathbf{n}_i$ of the surface, which is derived from the LiDAR-inertial SLAM system [57] as

$$^W \mathbf{R}_i = \begin{pmatrix} \frac{\mathbf{e}_x \times ^W \mathbf{n}_i}{\|\mathbf{e}_x \times ^W \mathbf{n}_i\|} & ^W \mathbf{n}_i \times \left(\frac{\mathbf{e}_x \times ^W \mathbf{n}_i}{\|\mathbf{e}_x \times ^W \mathbf{n}_i\|} \right) & ^W \mathbf{n}_i \end{pmatrix} \quad (3)$$

where $\mathbf{e}_x$ is the normal vector on the x-axis.

Finally, the covariance matrix of the Gaussian is constructed as follows:

$$\Sigma_{3Di} = (^W \mathbf{R}_i \mathbf{S}_i) (^W \mathbf{R}_i \mathbf{S}_i)^T \quad (4)$$

During the rasterization procedure, the influence of α_i is determined using the product of the 2-D Gaussian $\mathcal{N}(\mathbf{q}_i, \Sigma_{2Di})$ and opacity σ_i . The formation of the 2-D Gaussian involves splatting a 3-D Gaussian $\mathcal{N}(^W \mathbf{p}_i, \Sigma_{3Di})$ onto the screen space, illustrated as follows:

$$\mathbf{q}_i = \pi(^C \mathbf{T}_W ^W \mathbf{p}_i) \quad (5)$$

$$\Sigma_{2Di} = (\mathbf{J}_\pi ^C \mathbf{R}_W) \Sigma_{3Di} (\mathbf{J}_\pi ^C \mathbf{R}_W)^T \quad (6)$$

where π denotes the projection transformation. The linear approximation of the projective transformation π is denoted by the Jacobian $\mathbf{J}_\pi$. In addition, $^C \mathbf{T}_W \in \mathbf{SE}(3)$ represents the transformation from the world frame to the camera frame, with $^C \mathbf{R}_W$ denoting its rotational component.

2) *Real-Time Optimization of Gaussians in Sliding Window:* The zero-order spherical harmonic coefficient [25], representing the constant component of the spherical harmonic functions, is initialized using bilinear interpolation on the image captured by the camera, while the higher order coefficients are initialized to zero.

As illustrated by the following equation, bilinear interpolation is utilized to compute the color of projected nonintegral pixels

$$\mathbf{c}(\mathbf{q}_i) = \sum_{j=1}^4 \mathbf{c}_j \cdot A_j. \quad (7)$$

The computed value in $\mathbf{c}(\mathbf{q}_i)$ represents a weighted average of the colors of its neighboring pixels (within a 2x2 pixel patch). The weighting factor A_i corresponds to the area of the pixel location projected onto the neighboring integer pixels (as shown in Fig. 2 in Section II-B1).

After initialization, the Gaussian will be further optimized with photometric gradients. The optimization process is outlined as follows:

First, the Gaussians render an image $\hat{\mathbf{I}}_k$ with alpha blending, in accordance with

$$\hat{\mathbf{I}}_k(\mathbf{q}_i) = \sum_{i=1}^M \left[c_i \sigma_i G_i^{2D}(\mathbf{q}_i) \prod_{j=1}^{i-1} (1 - \sigma_j G_j^{2D}(\mathbf{q}_j)) \right] \quad (8)$$

where $G_i^{2D}(\mathbf{q}_i, \Sigma_{2Di})$ signifies the 2-D Gaussian derived from $G_i^{3D}(^C \mathbf{p}_i, \Sigma_{3Di})$ by applying a pose transform and local affine

transformation [58] illustrated in (6). The parameter $\sigma_i \in [0, 1]$ represents the opacity related to the Gaussians, and M indicates the number of Gaussians influencing the pixel.

For the map optimization process, we refine the Gaussian parameters in the sliding window by minimizing the photometric loss, a method adapted from [25].

$$\theta_{k-1}^* = \arg \min_{\theta_{k-1}} \sum_{G_i^{3D} \in \theta_{k-1}} \left\| \mathbf{I}_{k-1} - \hat{\mathbf{I}}_{k-1}(\mathbf{W} \mathbf{T}_C; \theta_{k-1}) \right\| \quad (9)$$

where θ_{k-1} denotes the structure parameters and spherical harmonic coefficients of the Gaussian elements within the current FoV. By minimizing this photometric loss, we iteratively adjust θ_{k-1} such that the rendered image $\hat{\mathbf{I}}_{k-1}$ better matches the observed image $\mathbf{I}_{k-1}$. We employ the Adam optimizer to efficiently solve this problem, updating the Gaussian parameters to achieve a more accurate and visually consistent map representation.

C. Maintenance of Gaussian Sliding Window

To improve memory efficiency and optimize computational speed, we limit the optimization scope to the Gaussian within the sliding window of Gaussians.

By limiting the optimization scope to the Gaussian within the sliding window of Gaussian, we can significantly enhance the optimization speed and reduce memory consumption. This targeted approach not only streamlines computational processes but also minimizes the usage of video memory, leading to more efficient performance. In addition, this restriction helps avoid the aliasing effects associated with tile-based depth sorting in the original implementation of 3D-GS. In the original 3D-GS, tile-based depth sorting can inadvertently cause contamination of the background by foreground points, leading to visual artifacts and inaccuracies in depth representation. By confining the optimization to the Gaussian sliding window, we mitigate these aliasing issues, ensuring a cleaner separation between foreground and background elements and improving the overall quality of the depth sorting process (shown in Fig. 13).

Rebuilding the Gaussian sliding window from the global Gaussian map for each frame naively requires extensive memory copying, which results in significant computational overhead. However, consecutive frames typically share a large portion of the scene, making much of this effort redundant. To capitalize on the substantial overlap between frames, we introduce an incremental update strategy for the Gaussian sliding window. This method significantly reduces unnecessary memory transfers, enhances real-time performance, and scales more effectively to large and complex environments.

As illustrated in Fig. 3, maintaining the Gaussian sliding window involves the following key components.

- 1) *Spatial hash table (SHT)*: A hash-based indexing structure that maps spatial coordinates to memory pointers in CPU memory. This ensures fast lookups and efficient organization of Gaussian parameters.
- 2) *CPU Gaussian buffer (CGB)*: A contiguous memory region in CPU Memory for storing Gaussian parameters of the currently active voxels. This compact layout enables swift data transfers to the GPU.

- 3) *GPU Gaussian buffer (GGB)*: An allocated memory block on the GPU that facilitates parallel processing and fast rendering by providing direct access to Gaussian data.

The incremental maintenance of the Gaussian sliding window involves a four-step process.

- 1) *Step 1 (Update to global map)*: Identify Gaussian voxels from the previous frame's sliding window of Gaussians that remain within the current FoV (*OVERLAP*). Voxels that fall outside the FoV will have their optimized parameters copied back to the global Gaussian map for persistent storage, and they will be marked as *DELETE*.
- 2) *Step 2 (Deletion and compaction)*: Swap the Gaussian parameters of voxels marked as *DELETE* with those at the rear of the sliding window sequence. After relocated all deletable voxels, remove them from the rear, thus preserving memory continuity.
- 3) *Step 3 (Overlap and addition)*: Using the spatial hash keys derived from the current LiDAR frame, we identify overlapping voxels (*OVERLAP*) with those from the sliding window of previous frame and determine new areas that need to be integrated into the sliding window of the current FoV (*ADD*).
- 4) *Step 4 (Appending new leaf voxels)*: Append all voxels (*ADD*) to the rear of the CGB and update the SHT accordingly. Subsequently, transfer the Gaussian data from the CGB (host memory) to GGB (device memory) directly, ensuring the Gaussians in the sliding window can be further optimized and rendered immediately by GPU.

This methodology significantly reduces redundant memory operations by incrementally updating only the visibility-changed voxels instead of reloading the entire sliding window. In addition, limiting optimization to Gaussians within the current FoV decreases computational consumption, thereby enhancing real-time performance. Furthermore, leveraging the scalability and ample capacity of CPU memory enables the system to handle larger and more complex environments with improved robustness and efficiency.

D. State Estimation

Building upon the characteristics of Gaussian rendering, we have redesigned the visual update pipeline of FAST-LIVO2 [8]. Instead of warping patches from the current frame to the reference frame to compute the photometric error, we uniformly compute the photometric loss on the current frame by comparing the image rendered from the Gaussian map with the actual image. The convergence of this optimization is guaranteed by the smooth and differentiable nature of Gaussian rendering, as demonstrated in MonoGS [52].

As shown in Fig. 2, our odometry system tightly integrates LiDAR and image measurements using an IESKF with sequential updates, which is modified from FAST-LIVO2 [8].

The LiDAR-inertial pose estimation is similar to the approach in [57] and [59], which utilizes a size-adaptive voxel method to extract planar features from the scene.

Our visual module is based on a semidense method, which is the same as [7] and [8]. However, unlike Zheng et al. [8],

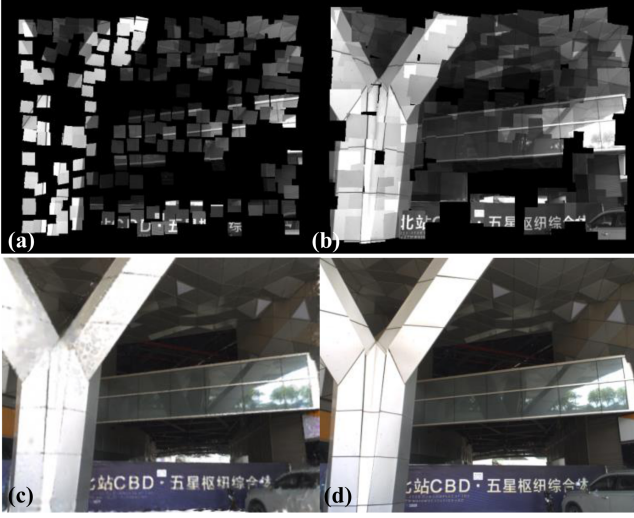


Fig. 4. Comparison of map representation delicacy with patch based method. (a) and (b) Warping transformation results using patch sizes of 32 and 64, respectively. (c) Gaussian rendering results. (d) Reference (ground truth) image.

we utilize a dense Gaussian map instead of a sparse visual map. First, we employed the optimized Gaussian maps in current FoV to render a novel view with the LiDAR-inertial updated pose.

For the visual measurement model, we minimize the following photometric residual:

$${}^W\mathbf{T}_C^* = \arg \min_{\mathbf{T}(\xi)} \sum_{G_i^{3D} \in \theta_{k-1}} \left\| \mathbf{I}_k - \hat{\mathbf{I}}_k({}^W\mathbf{T}_C; \theta_{k-1}) \right\| \quad (10)$$

where θ_{k-1} represents the set of G_i^{3D} used to construct photometric errors, ξ denotes the Lie algebra of the camera pose ${}^W\mathbf{T}_C$ to be optimized. By minimizing (10), we iteratively optimize ξ to make the rendered image $\hat{\mathbf{I}}_k$ from Gaussian map best match the observed image $\mathbf{I}_k$.

As shown in Fig. 4, the Gaussian map is rendered at the estimated camera pose of the current frame. For comparison, we warp the patch from the reference frame to the current frame using various patch size settings (to emulate the different levels of pyramid used in the FAST-LIVO2 method). With increasing patch size (and level of pyramid), it is readily observable that there appear to be seams between patches. However, our Gaussian map representation is capable of delivering not only seamless rendering but also rendering non-Lambertian surfaces with a photorealistic quality, which highlights the advantages of our approach based on the Gaussian method.

We derive the Jacobian from photometric loss of Gaussian rendering to the IESKF-based estimator's pose update of IMU pose.

The Jacobian that relates photometric loss to camera pose is derived similarly to MonoGS [52], as demonstrated in the following equations:

$$\frac{\partial \mathbf{q}_i}{\partial {}^W\mathbf{R}_I} = \frac{\partial \mathbf{q}_i}{\partial {}^C\mathbf{p}_i} \frac{\partial {}^C\mathbf{p}_i}{\partial {}^C\mathbf{R}_W} \frac{\partial {}^C\mathbf{R}_W}{\partial {}^W\mathbf{R}_I} \quad (11)$$

$$\frac{\partial \mathbf{q}_i}{\partial {}^W\mathbf{t}_I} = \frac{\partial \mathbf{q}_i}{\partial {}^C\mathbf{p}_i} \frac{\partial {}^C\mathbf{p}_i}{\partial {}^C\mathbf{t}_W} \frac{\partial {}^C\mathbf{t}_W}{\partial {}^W\mathbf{t}_I} + \frac{\partial \mathbf{q}_i}{\partial {}^C\mathbf{p}_i} \frac{\partial {}^C\mathbf{p}_i}{\partial {}^C\mathbf{R}_W} \frac{\partial {}^C\mathbf{R}_W}{\partial {}^W\mathbf{t}_I} \quad (12)$$

$$\frac{\partial \Sigma_{2D}}{\partial {}^W\mathbf{R}_I} = \frac{\partial \Sigma_{2D}}{\partial \mathbf{J}_\pi} \frac{\partial \mathbf{J}_\pi}{\partial {}^C\mathbf{p}_i} \frac{\partial {}^C\mathbf{p}_i}{\partial {}^C\mathbf{R}_W} \frac{\partial {}^C\mathbf{R}_W}{\partial {}^W\mathbf{R}_I} + \frac{\partial \Sigma_{2D}}{\partial {}^C\mathbf{R}_W} \frac{\partial {}^C\mathbf{R}_W}{\partial {}^W\mathbf{R}_I} \quad (13)$$

$$\begin{aligned} \frac{\partial \Sigma_{2D}}{\partial {}^W\mathbf{t}_I} &= \frac{\partial \Sigma_{2D}}{\partial \mathbf{J}_\pi} \frac{\partial \mathbf{J}_\pi}{\partial {}^C\mathbf{p}_i} \frac{\partial {}^C\mathbf{p}_i}{\partial {}^C\mathbf{t}_W} \frac{\partial {}^C\mathbf{t}_W}{\partial {}^W\mathbf{t}_I} \\ &+ \frac{\partial \Sigma_{2D}}{\partial \mathbf{J}_\pi} \frac{\partial \mathbf{J}_\pi}{\partial {}^C\mathbf{p}_i} \frac{\partial {}^C\mathbf{p}_i}{\partial {}^C\mathbf{R}_W} \frac{\partial {}^C\mathbf{R}_W}{\partial {}^W\mathbf{t}_I}. \end{aligned} \quad (14)$$

This update is systematically extended from the camera pose to the IMU pose, integrated within the IESKF framework, as demonstrated in the following equations:

$$\frac{\partial {}^C\mathbf{R}_W}{\partial {}^W\mathbf{R}_I} = -{}^I\mathbf{R}_C^T \quad (15)$$

$$\frac{\partial {}^C\mathbf{t}_W}{\partial {}^W\mathbf{t}_I} = -{}^W\mathbf{R}_C^T \quad (16)$$

$$\frac{\partial {}^C\mathbf{R}_W}{\partial {}^W\mathbf{t}_I} = -{}^C\mathbf{t}_I^{\wedge C} \mathbf{R}_W. \quad (17)$$

It is important to highlight that the majority of Gaussian splatting-based SLAM methods primarily depend on optimizers to compute the camera pose updates. Typically, these approaches do not assess the covariance of the updated pose; however, the pose and its covariance can be further propagated to the next sensor update such as IMU and LiDAR, and can help forming a tightly coupled IESKF system.

III. EXPERIMENT

To provide a comprehensive assessment of the proposed system, we conducted experiments on distinct computing platforms, including a high-performance desktop and an embedded device. We first performed a comparative study against several state-of-the-art SLAM algorithms on a desktop computer (Intel i9-13900KF CPU, 128-GB RAM, and NVIDIA RTX-4090 GPU). The results demonstrate that our method achieves competitive accuracy and efficiency relative to existing approaches. Subsequently, we deployed the system on an on-board computing platform, the NVIDIA Jetson Orin NX.¹ Despite the limited computational resources on the onboard PC, our algorithm consistently maintains real-time performance, highlighting its suitability for robotic platforms.

A. Dataset Preparation

In our research, we utilized multiple datasets, including public and self-collected. Among the public datasets, we selected the FAST-LIVO2 dataset [8], specifically the “CBD03” and “HKU01” sequences, which depict extensive large-scale university scenes. In addition, we employed the “HKairport01”

¹ Equipped with an 8-core CPU and 1024 CUDA cores, featuring 16-GB LPDDR5 memory. For more details, see <https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-orin-nx/>

and “HKisland03” sequences from the MARS-LVIG (Multi-sensor Aerial Robotic System - LiDAR Visual Inertial Gaussian) dataset [9], which provide data collected in vast natural environments of mountains and seas using aerial robotic vehicles. The MARS-LVIG dataset is distinguished by its inclusion of a D-RTK system (DJI’s Differential Real-Time Kinematic GNSS system), which provides precise ground truth for odometry.

To complement these, we collected three proprietary sequences (“Playground01,” “Playground02,” and “landmark01”) in small-scale indoor environments using a motion capture system as ground truth. This setup allowed us to evaluate the accuracy of our algorithm under controlled conditions. To ensure data quality, we meticulously calibrated the camera’s intrinsics using a checkerboard [22], while the extrinsics between the LiDAR and camera were calibrated following the procedure in [60]. Furthermore, the temporal and spatial alignment between the optical tracker and odometry were rigorously calibrated according to the methodology in [61]. These efforts ensured that both the public and proprietary datasets were of high quality, with precise calibration and synchronization to support robust evaluation. We also utilized the Oxford Spires dataset [62], which provides high-precision ground truth trajectories (1–2 cm accuracy) through LiDAR-TLS map registration. From this dataset, we selected the sequence of “Radcliffe01” for evaluation.

B. Comparative Experiments

In this section, we conduct comprehensive assessments of our system by thoroughly analyzing two critical aspects: the mapping quality through Gaussian rendering performance and the precision of the odometry.

For Gaussian-based scene reconstruction, we compare our approach with several LiDAR-assisted methods including S3Gaussian [63] and LetsGo [48]. With respect to odometry precision, our experiments cover both recent multisensor fusion SLAM systems with traditional representations [5], [7], [55] and state-of-the-art SLAM frameworks utilizing Gaussian-based maps [33], [52].

1) *Evaluation of Mapping Quality:* For parameter settings, we employ different configurations for indoor and outdoor scenarios. In indoor environments, we use a fine root voxel size of 0.03 m with a maximum subdivision level of 2 to capture detailed features. For large-scale outdoor aerial environments, we adopt a coarser root voxel size of 1.0 m while maintaining the same subdivision level. For fair comparison, we run each method for 15 000 iterations (equivalent to ten iterations over 1500 frames) to ensure thorough optimization convergence.

As shown in Table I, we first compare our approach with LiDAR-integrated Gaussian reconstruction methods such as S3Gaussian [63] and LetsGo [48]. While LetsGo achieves adaptive LoD through distance-based voxel sizing, our fixed-level octree structure demonstrates comparable rendering quality with reduced computational overhead. The efficiency gains stem from our sliding-window strategy that enables real-time sliding-window updates, in contrast to LetsGo’s offline processing approach.

Furthermore, we evaluated our system against RGB-D SLAM methods that use Gaussian representations, specifically

TABLE I
COMPARATIVE EVALUATION FOR RENDERING

Sequence	Method	PSNR/dB $\uparrow$	Dur./s $\downarrow$	Mem./GB $\downarrow$
Indoor Datasets				
HKU01 [7]	3D-GS [1]	26.22	2128.6	13.8
	M2Mapping [64]	25.06	2276.9	12.5
	SplaTAM [33]	24.06	292.2	2.6
	MonoGS [52]	23.51	258.0	3.1
	S3GS [63]	$\times$	$\times$	$\times$
	LetsGo [48]	24.51	3231.3	18.1
	GS-LIVO (Ours)	25.34	82.5	2.2
CBD03 [7]	3D-GS [1]	29.54	1873.8	12.5
	M2Mapping [64]	27.65	1978.9	13.2
	SplaTAM [33]	26.85	265.2	4.8
	MonoGS [52]	27.10	278.4	4.6
	S3GS [63]	24.92	3450.7	15.2
	LetsGo [48]	25.51	3573.6	20.4
	GS-LIVO (Ours)	27.52	88.4	2.2
Playground01	3D-GS [1]	29.75	763.4	8.8
	M2Mapping [64]	27.32	786.8	13.1
	SplaTAM [33]	22.01	292.2	3.4
	MonoGS [52]	21.15	281.0	3.1
	S3GS [63]	20.50	2903.2	9.7
	LetsGo [48]	25.20	3210.3	10.2
	GS-LIVO (Ours)	24.09	48.5	1.5
Playground02	3D-GS [1]	26.54	873.8	7.5
	M2Mapping [64]	25.12	957.7	12.1
	SplaTAM [33]	24.45	278.4	3.8
	MonoGS [52]	25.93	265.2	4.6
	S3GS [63]	22.24	2957.3	19.1
	LetsGo [48]	23.12	2967.5	18.2
	GS-LIVO (Ours)	25.52	63.4	1.2
Outdoor Datasets				
HKisland03 [9]	3D-GS [1]	17.52	3494.1	21.6
	SplaTAM [33]	12.60	790.0	10.5
	MonoGS [52]	14.22	743.7	12.3
	S3GS [63]	$\times$	$\times$	$\times$
	LetsGo [48]	18.32	2803.3	17.6
	GS-LIVO (Ours)	15.32	82.8	3.2
HKairport01 [9]	3D-GS [1]	16.98	3919.8	22.8
	SplaTAM [33]	12.39	915.2	11.0
	MonoGS [52]	13.87	789.6	13.4
	S3GS [63]	$\times$	$\times$	$\times$
	LetsGo [48]	17.32	3103.3	18.1
	GS-LIVO (Ours)	15.18	93.2	3.1

SplaTAM [33] and MonoGS [52]. For comparison purposes, we converted LiDAR measurements to depth maps. These methods employ different strategies for map management—SplaTAM uses silhouette-based selection, while MonoGS relies on covisibility-based keyframe selection. Our approach directly downsamples Gaussians in 3-D voxel space, enabling more efficient extraction of structural features. Moreover, the integration of IMU measurements provides motion priors that enhance robustness against rapid movements and vibrations compared to pure RGB-D methods. For implicit representation, M2Mapping [64] also shows competitive performance in rendering, particularly in medium-scale scenes, but requires similarly long processing times.

Fig. 5 presents three distinct real-world scenes: 1) HKU campus, 2) UAV playground, and 3) a well-known landmark. For each scene (organized by columns), we show three types of visualizations: the rendering from the camera view (top row), the Gaussian visualization from a roaming perspective (middle row), and the underlying Gaussian structure (bottom row).

In the first scene (1), we demonstrate high rendering fidelity with clear “HKU” text visible on the building signs. In the second



Fig. 5. (a)–(c) Mapping results of three distinct real-world scenes. Top row: The rendering results from camera poses. Middle row: The rendering results from roaming perspectives. Bottom row: The shapes of scene Gaussians.

scene (2), the checkerboard patterns in the playground illustrate a precise geometry reconstruction. The third scene (3) highlights fine rendering details, such as crisp inscriptions.

The first scene demonstrates high rendering fidelity with clear HKU text on the building signs. The second scene showcases precise geometry reconstruction, evidenced by sharp checkerboard patterns on the playground. In the third scene, the inscriptions are rendered with crisp detail. In the bottom row for each scene, the Gaussians naturally extend along surface orientations, demonstrating how our method effectively captures the scene geometry through joint LiDAR visual optimization.

2) *Evaluation of Localization*: Fig. 6 presents a comprehensive performance analysis of various SLAM systems. In indoor environments [see Fig. 6(a)], Our method achieves comparable accuracy with traditional LIV-based SLAM methods while being significantly more efficient than other GS-based approaches. For outdoor scenarios [see Fig. 6(b)], our method demonstrates superior accuracy with an RMSE of 0.042 m, R^3 LIVE (1.465 m), and

LVI-SAM (4.665 m). While the processing time of GS-LIVO is slightly higher than some traditional methods, it maintains real-time performance while providing enhanced mapping capabilities through its Gaussian splatting representation.

For consistent evaluation across different datasets, we configure the experimental parameters as: image resolution of 640×480 , octree configuration of (0.06 m, two layers) for indoor and (0.5 m, two layers) for outdoor environments, and a sliding window size of 100 000 Gaussians for incremental map updates.

As demonstrated in Table II, GS-LIVO demonstrates competitive localization accuracy, showing notable improvements over traditional methods such as R^3 LIVE, while maintaining performance comparable to FAST-LIVO with slightly lower precision. In terms of computational efficiency, our system maintains processing times below 90 ms in both indoor and outdoor environments. This difference in processing time can be attributed to our distinct mapping approach: while FAST-LIVO [8] achieves

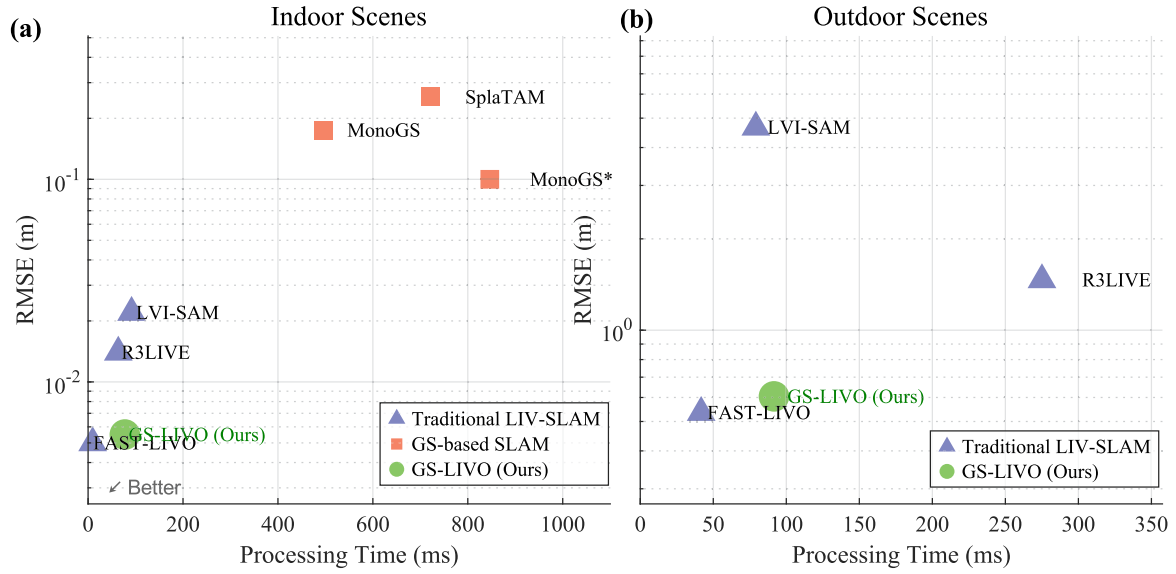


Fig. 6. Performance comparison of different SLAM systems in terms of accuracy (RMSE) and computational efficiency (processing time). (a) Indoor scenes. (b) Outdoor scenes.

TABLE II
COMPARATIVE EVALUATION OF LIV-BASED SLAM SYSTEMS ACROSS DATASETS (PINK: BEST; ORANGE: SECOND-BEST)

Method	Sequence	RMSE/m↓	Dur./ms↓
Outdoor Datasets			
FAST-LIVO [7]	<i>HKisland03</i>	0.51	38.9
	<i>HKairport01</i>	0.56	44.5
	<i>Radcliffe01</i>	0.63	49.8
R ³ LIVE [5]	<i>HKisland03</i>	1.71	283.3
	<i>HKairport01</i>	1.22	266.8
	<i>Radcliffe01</i>	1.85	292.4
LVI-SAM [55]	<i>HKisland03</i>	4.12	73.5
	<i>HKairport01</i>	5.21	84.8
	<i>Radcliffe01</i>	6.15	83.6
GS-LIVO (Ours)	<i>HKisland03</i>	0.58	82.8
	<i>HKairport01</i>	0.63	93.2
	<i>Radcliffe01</i>	0.75	94.8
Indoor Datasets			
FAST-LIVO [7]	<i>Playground01</i>	0.005	10.5
	<i>Playground02</i>	0.005	8.75
R ³ LIVE [5]	<i>Playground01</i>	0.014	60.6
	<i>Playground02</i>	0.014	66.6
LVI-SAM [55]	<i>Playground01</i>	0.023	96.6
	<i>Playground02</i>	0.021	86.6
GS-LIVO (Ours)	<i>Playground01</i>	0.006	48.5
	<i>Playground02</i>	0.005	63.4

TABLE III
COMPARATIVE EVALUATION OF GS-BASED SLAM SYSTEMS ACROSS DATASETS (PINK: BEST; ORANGE: SECOND-BEST)

Method	Sequence	RMSE/m↓	Dur./ms↑	Mem./GB↓
Indoor Datasets				
SplaTAM [33]	<i>Playground01</i>	0.28	612.8	12.5
	<i>Playground02</i>	0.23	831.6	21.0
MonoGS* [52]	<i>Playground01</i>	0.09	841.5	19.6
	<i>Playground02</i>	0.11	851.2	17.2
MonoGS [52]	<i>Playground01</i>	0.18	541.5	21.0
	<i>Playground02</i>	0.17	451.2	18.7
GS-LIVO (Ours)	<i>Playground01</i>	0.006	48.5	1.2
	<i>Playground02</i>	0.005	63.4	1.5
Outdoor Datasets				
SplaTAM [33]	<i>HKisland03</i>	×	×	×
	<i>HKairport01</i>	×	×	×
	<i>Radcliffe01</i>	×	×	×
MonoGS* [52]	<i>HKisland03</i>	×	×	×
	<i>HKairport01</i>	×	×	×
	<i>Radcliffe01</i>	×	×	×
MonoGS [52]	<i>HKisland03</i>	×	×	×
	<i>HKairport01</i>	×	×	×
	<i>Radcliffe01</i>	×	×	×
GS-LIVO (Ours)	<i>HKisland03</i>	0.58	82.8	8.0
	<i>HKairport01</i>	0.63	93.2	9.5
	<i>Radcliffe01</i>	0.75	94.8	9.7

efficient pose optimization through sparse visual submap warping, our system simultaneously optimizes camera poses while maintaining a photometrically accurate dense Gaussian map, requiring more computational resources for real-time dense map updates. This design choice enables us to maintain not just a hand-crafted sparse map for odometry but a photorealistic dense representation that supports high-fidelity scene reconstruction.

As presented in Table III, we conducted a comprehensive comparative analysis between our proposed algorithm and state-of-the-art Gaussian-based SLAM systems, specifically SplaTAM [33] and MonoGS [52]. Although MonoGS originally

supports RGB-D and RGB as input, MonoGS [52] market with * in Table III denotes the version enhanced with LiDAR-projected depth image, while another refers to its monocular version.

Unlike these Gaussian-based SLAM system that rely solely on depth loss for pose estimation or a constant velocity motion model, our approach integrates LiDAR-inertial odometry pose as a prior, resulting in a significant improvement in accuracy. This enhances the algorithm's capability to manage the challenging motion conditions frequently faced by robotic systems.

Our system effectively handles large-scale outdoor environments, as demonstrated in Fig. 7, maintaining both real-time performance and high accuracy with a trajectory root mean

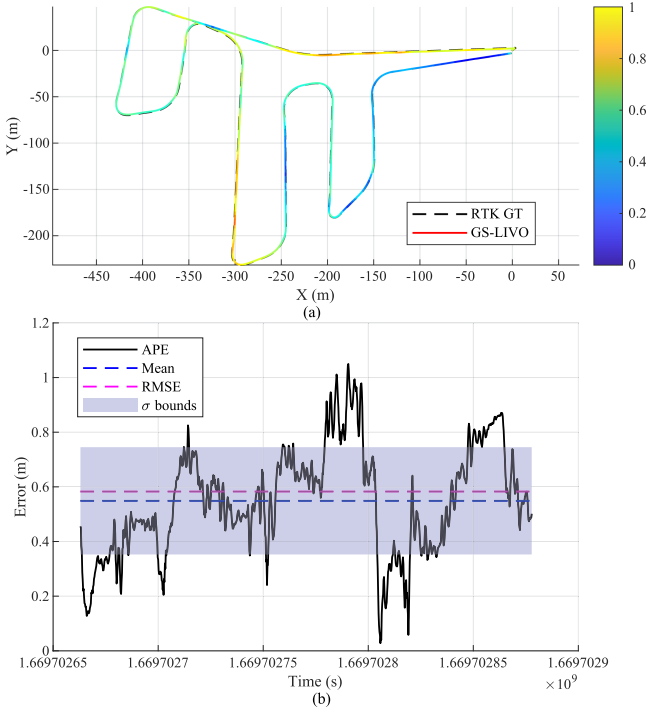


Fig. 7. Trajectory and error analysis for the sequence of HKisland03 from [9], using RTK as ground truth. (a) Estimated trajectory color-coded by positional deviation. (b) Temporal evolution of absolute position error (APE) with mean and standard deviation bands. (a) Trajectory comparison with error distribution. (b) Absolute pose error analysis.

square error (RMSE) of 0.58 m, substantially outperforming traditional methods such as R^3 LIVE [5] and LVI-SAM [55]. In particular, our computational efficiency remains competitive with traditional approaches. While R^3 LIVE's computation time grows with map size due to increasing ikdtree indexing overhead for colored point clouds, and LVI-SAM incurs significant computational cost from its indirect method despite using a sparse map, our system maintains near real-time performance through efficient sliding window management of Gaussian points, even while maintaining a photorealistic map. In the next section, a detailed analysis of system performance is followed, including processing time and GPU memory consumption.

C. Ablation Study of Sliding Window

In our ablation study, we compared the VRAM usage and the optimization duration with and without the sliding window for Gaussian, as well as the processing time required for the map maintaining process both in indoor and outdoor sequence.

1) *Memory Consumption*: The implementation of sliding window for Gaussians shows significant advantages in both indoor (shown in Fig. 8) and outdoor (shown in Fig. 9) environments. Our strategy of storing activated Gaussians of the current FoV in GPU memory while maintaining the global map through an octree structure in CPU memory achieves an optimal balance between mapping quality and computational efficiency. As shown in Figs. 8(a) and 9(a), this approach maintains high peak signal-to-noise ratio (PSNR) values comparable to full

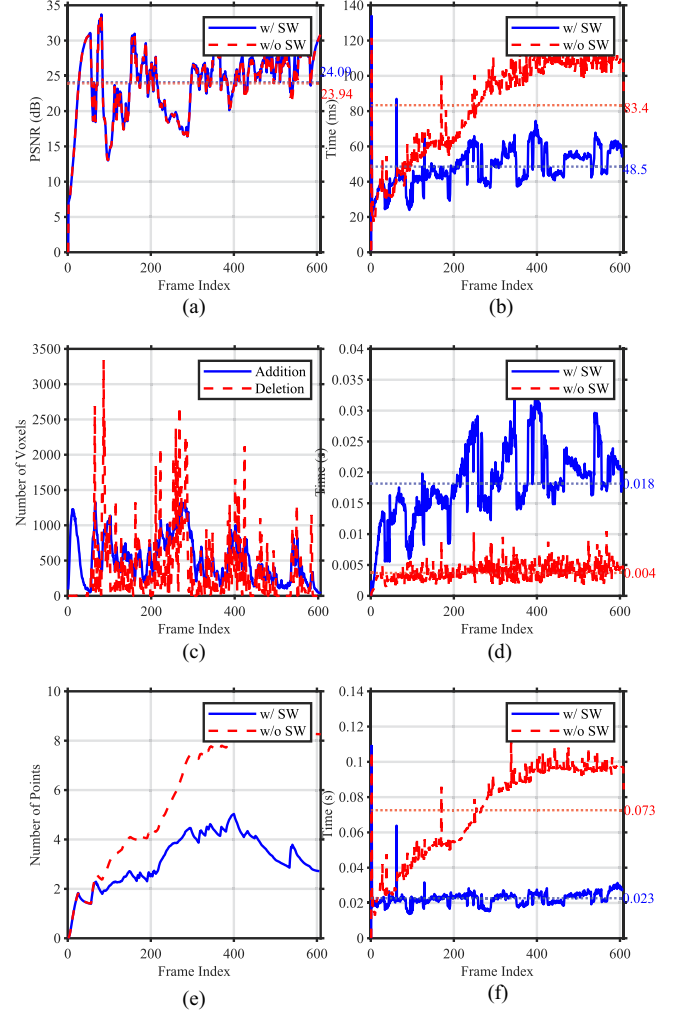


Fig. 8. Performance analysis of the sliding window approach in indoor environments (sequence of Playground01.bag). (a) PSNR comparison. (b) Total runtime analysis. (c) Voxel operation in sliding window. (d) Maintenance time of sliding window. (e) Gaussians in sliding window. (f) Optimization runtime.

GPU implementations while significantly reducing memory consumption [see Figs. 8(e) and 9(e)]. This efficient map management enables our system to process large-scale environments and complex scenes where traditional 3D-GS approaches would face GPU memory constraints, demonstrating the practical scalability of our method.

2) *Time Consumption*: As shown in Figs. 8(b) and 9(b), with our sliding window strategy, the total processing time—including both window maintenance and Gaussian optimization—remains consistently below 100 ms in both indoor and outdoor environments, enabling real-time updates at 10 Hz. In contrast, approaches without sliding window optimization show steadily increasing computation times as the map grows, which hinders real-time performance in large-scale scenarios. Besides, Fig. 10 illustrates the statistical relationship between the number of Gaussians in the map, the optimization time, and the map maintenance time.

The breakdown of processing time, illustrated in Fig. 11, demonstrates that our sliding window strategy achieves efficient

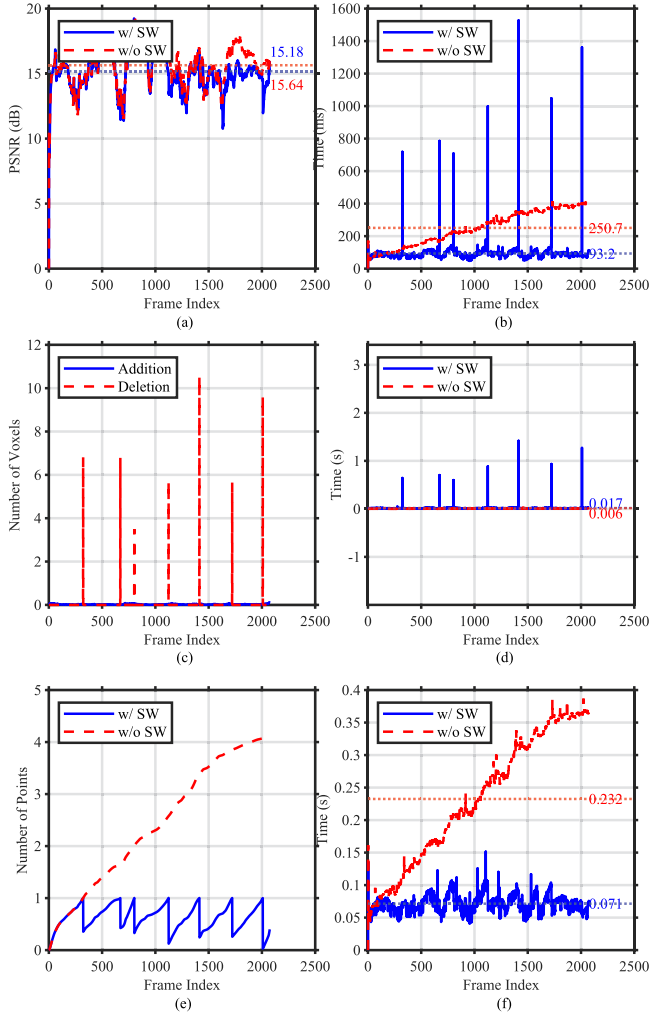


Fig. 9. Performance analysis of the sliding window approach in outdoor environments (sequence of HKisland03.bag). (a) PSNR comparison. (b) Total runtime analysis. (c) Operation in sliding window. (d) Maintenance time of sliding window. (e) Gaussians in sliding window. (f) Optimization runtime.

response times across different components. The processing overhead averages 23 ms for indoor scenes and 71 ms for outdoor environments, which can be further optimized by adjusting the LoD parameters based on available computing resources. This adaptability makes our approach suitable for various platforms and scenarios.

Importantly, as shown in Figs. 8 and 9, our sliding-window approach maintains high mapping quality while significantly reducing computational overhead. The system consistently achieves PSNR values around 25 dB, with only temporary reductions during viewpoint changes. Through iterative optimization within the sliding window, PSNR quickly recovers to values between 25 and 30 dB, demonstrating that our efficiency gains do not come at the cost of mapping quality.

D. Experiment on Embedded System

To validate the efficiency of our algorithm, we deployed GS-LIVO on a mobile platform equipped with NVIDIA Jetson Orin NX [see Fig. 12(e)], configured with root voxel size of 0.5 m,

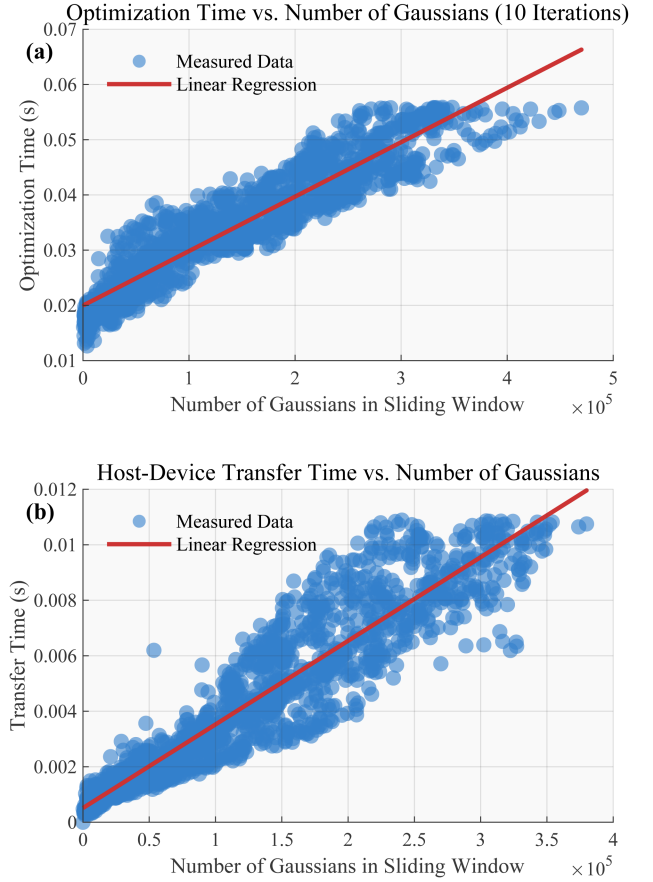


Fig. 10. Relationship of the number of Gaussians with two performance metrics: optimization time (top panel) and GPU-CPU transfer time (bottom panel).

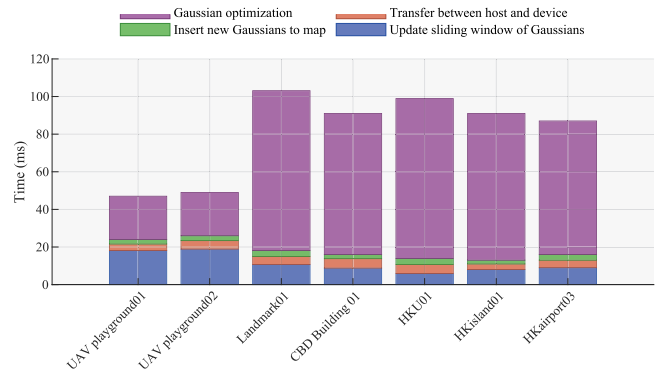


Fig. 11. Time consumption analysis of the mapping process.

two subdivision layers, image resolution of 256×216 , and a sliding window size of 20 000 Gaussians.

The system maintains real-time performance on the embedded platform (ORIN NX 16 G), with optimization taking 15.3 ms [see Fig. 12(b)], map maintenance 18.9 ms [see Fig. 12(c)], and total pipeline takes 48.3 ms while achieving a PSNR of 23.52 dB [see Fig. 12(a)].

To further demonstrate the real-time and high-precision performance of our proposed odometry, we integrated GS-LIVO into a complete autonomous navigation system. The Gaussian

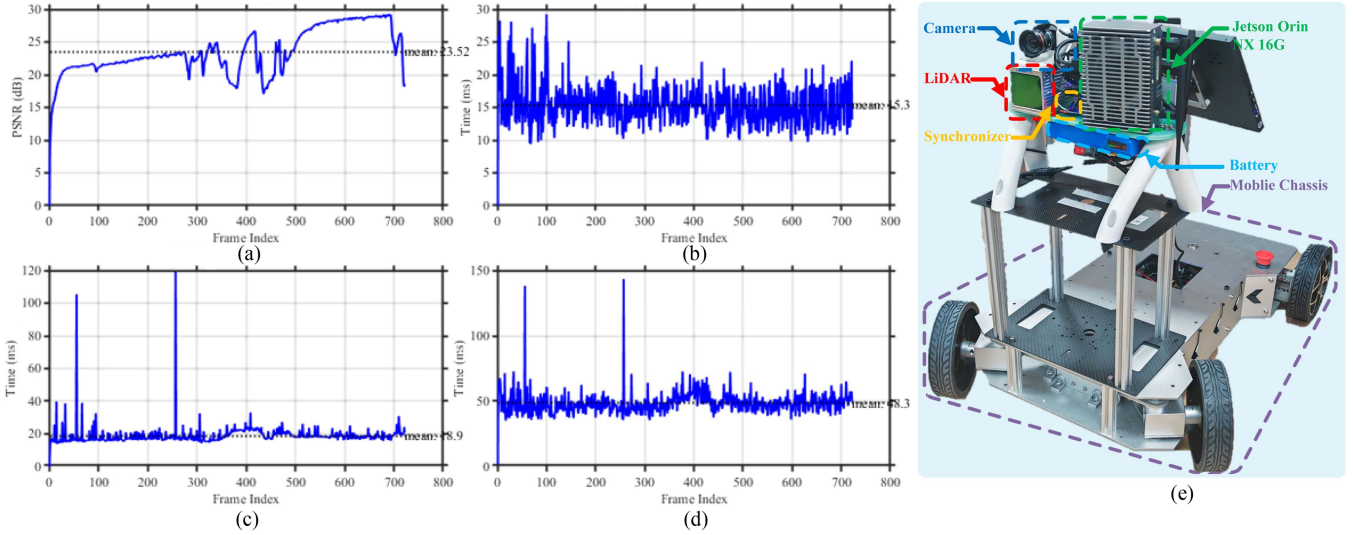


Fig. 12. Performance evaluation of GS-LIVO on the embedded platform: (a)–(d) System metrics including PSNR and processing time analysis. (e) Our sensor suite integrated with Jetson Orin NX mounted on a mobile chassis. (a) PSNR. (b) Optimization runtime. (c) Map maintenance time. (d) Total runtime. (e) Our sensor suit mounted on a mobile chassis.



Fig. 13. Comparison of aliasing artifacts in scenes with occlusions.

map is processed to generate 2-D occupancy grids for path planning, while the odometry provides real-time localization for trajectory tracking. The integrated system successfully demonstrates autonomous navigation using standard planning and control algorithms (A* for global planning and LQR for trajectory tracking). Demonstration results are shown in the supplementary video.

To the best of our knowledge, this is the first real-time Gaussian-based SLAM system with online map updates deployed on an ARM-based embedded platform.

IV. CONCLUSION

In this article, we presented GS-LIVO, a novel real-time SLAM system that integrates traditional LIVO with novel map representation of 3D-GS. By replacing conventional colored point clouds and sparse patch maps with Gaussian-based scene representation, our system achieves both accurate localization and high-fidelity mapping. Our key contributions include the following:

- 1) a spatial hash-indexed octree structure for efficient global Gaussian map management;
- 2) LiDAR-visual joint initialization for high-fidelity mapping;

- 3) an incremental sliding window strategy for real-time map optimization;
- 4) a tightly coupled multisensor fusion framework using the IESKF.

Our system addressed the challenge of real-time map updates by leveraging multisensor fusion and tightly coupled odometry, improving both localization accuracy and map update frequency. GS-LIVO was the first Gaussian-based SLAM system successfully deployed on the NVIDIA Jetson Orin NX platform, showcasing its practical application for autonomous robotic navigation. This deployment was a key step in demonstrating the system's real-time capabilities and its applicability to real-world environments, validating the practical value of the approach. Extensive experiments confirmed that GS-LIVO outperformed existing methods, achieving superior performance in both indoor and outdoor settings, reducing memory consumption, and optimizing processing time while maintaining high-quality rendering.

However, our system still struggled with indoor–outdoor transition situation. Solutions such as size-adaptive voxels, which dynamically adjust the subdivision levels of the octree according to the scene, would be promising research directions to address these challenging scenarios.

REFERENCES

- [1] G. Kopanas, B. Kerbl, A. Guédon, and J. Luiten, “3D Gaussian splatting tutorial,” in *Proc. Int. Conf. 3D Vis. Tutorial*, 2024. [Online]. Available: <https://3dgtutorial.github.io/>
- [2] W. Xu and F. Zhang, “FAST-LIO: A fast, robust LiDAR-inertial odometry package by tightly-coupled iterated Kalman filter,” *IEEE Robot. Automat. Lett.*, vol. 6, no. 2, pp. 3317–3324, Apr. 2021.
- [3] W. Xu, Y. Cai, D. He, J. Lin, and F. Zhang, “FAST-LIO2: Fast direct LiDAR-inertial odometry,” *IEEE Trans. Robot.*, vol. 38, no. 4, pp. 2053–2073, Aug. 2022.
- [4] J. Lin, C. Zheng, W. Xu, and F. Zhang, “R² Live: A robust, real-time, LiDAR-inertial-visual tightly-coupled state estimator and mapping,” *IEEE Robot. Automat. Lett.*, vol. 6, no. 4, pp. 7469–7476, Oct. 2021.

- [5] J. Lin and F. Zhang, "R³ Live: A robust, real-time, RGB-colored, LiDAR-inertial-visual tightly-coupled state estimation and mapping package," in *Proc. Int. Conf. Robot. Automat.*, 2022, pp. 10672–10678.
- [6] J. Lin and F. Zhang, "R³ Live++: A robust, real-time, radiance reconstruction package with a tightly-coupled LiDAR-inertial-visual state estimator," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 46, no. 12, pp. 11168–11185, Dec. 2024.
- [7] 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.*, 2022, pp. 4003–4009.
- [8] C. Zheng et al., "FAST-LIVO2: Fast, direct LiDAR-inertial-visual odometry," *IEEE Trans. Robot.*, vol. 41, pp. 326–346, 2024.
- [9] H. Li et al., "MARS-LVIG dataset: A multi-sensor aerial robots SLAM dataset for LiDAR-visual-inertial-GNSS fusion," *Int. J. Robot. Res.*, vol. 43, 2024, Art. no. 02783649241227968.
- [10] C. Forster, Z. Zhang, M. Gassner, M. Werlberger, and D. Scaramuzza, "SVO: Semidirect visual odometry for monocular and multicamera systems," *IEEE Trans. Robot.*, vol. 33, no. 2, pp. 249–265, Apr. 2017.
- [11] J. Lin et al., "ImMesh: An immediate LiDAR localization and meshing framework," *IEEE Trans. Robot.*, vol. 39, no. 6, pp. 4312–4331, Oct. 2023.
- [12] Y. Jia et al., "CAD-Mesher: A Convenient, Accurate, Dense Mesh-based Mapping Module in SLAM for Dynamic Environments," **IEEE Trans. Multimedia**, 2025, *arXiv:2408.05981*, doi: [10.48550/arXiv.2408.05981](https://doi.org/10.48550/arXiv.2408.05981).
- [13] Y. Zhu, X. Zheng, and J. Zhu, "MeshLOAM: Accepted by IEEE Transactions on Intelligent Vehicles," *IEEE Trans. Intell. Veh.*, to be published, doi: [10.1109/TIV.2024.3409085](https://doi.org/10.1109/TIV.2024.3409085).
- [14] J. Ruan, B. Li, Y. Wang, and Y. Sun, "SLAMesh: Real-time LiDAR simultaneous localization and meshing," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2023, pp. 3546–3552.
- [15] W. Wang, H. Qian, and S. Feng, "Simplified unstructured-mesh based UAV path planning method using Octree overlap detection," *IEEE Trans. Intell. Veh.*, vol. 9, no. 10, pp. 6394–6406, Oct. 2024.
- [16] C. Wang, Y. Zhang, and X. Li, "PMDS-SLAM: Probability mesh enhanced semantic SLAM in dynamic environments," in *Proc. 5th Int. Conf. Control, Robot. Cybern.*, 2020, pp. 40–44.
- [17] H. M. Cho, H. Jo, and E. Kim, "SP-SLAM: Surfel-point simultaneous localization and mapping," *IEEE/ASME Trans. Mechatron.*, vol. 27, no. 5, pp. 2568–2579, Oct. 2021.
- [18] J. Quenzel and S. Behnke, "Real-time multi-adaptive-resolution-surfel 6D LiDAR odometry using continuous-time trajectory optimization," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2021, pp. 5499–5506.
- [19] T.-M. Nguyen, D. Duberg, P. Jensfelt, S. Yuan, and L. Xie, "SLICT: Multi-input multi-scale surfel-based LiDAR-inertial continuous-time odometry and mapping," *IEEE Robot. Automat. Lett.*, vol. 8, no. 4, pp. 2102–2109, Apr. 2023.
- [20] K. Wang, F. Gao, and S. Shen, "Real-time scalable dense surfel mapping," in *Proc. Int. Conf. Robot. Automat.*, 2019, pp. 6919–6925.
- [21] Y. Gao, Y.-P. Cao, and Y. Shan, "SurfelNeRF: Neural surfel radiance fields for online photorealistic reconstruction of indoor scenes," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2023, pp. 108–118.
- [22] W. Gao, K. Wang, W. Ding, F. Gao, T. Qin, and S. Shen, "Autonomous aerial robot using dual-fisheye cameras," *J. Field Robot.*, vol. 37, no. 4, pp. 497–514, 2020.
- [23] T. Qin, P. Li, and S. Shen, "VINS-Mono: A robust and versatile monocular visual-inertial state estimator," *IEEE Trans. Robot.*, vol. 34, no. 4, pp. 1004–1020, Aug. 2018.
- [24] 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, pp. 99–106, 2021.
- [25] B. Kerbl, G. Kopanas, T. Leimkühler, and G. Drettakis, "3D Gaussian splatting for real-time radiance field rendering," *ACM Trans. Graph.*, vol. 42, no. 4, pp. 1–14, 2023.
- [26] B. Kerbl, A. Meuleman, G. Kopanas, M. Wimmer, A. Lanvin, and G. Drettakis, "A hierarchical 3D Gaussian representation for real-time rendering of very large datasets," *ACM Trans. Graph.*, vol. 43, no. 4, pp. 1–15, Jul. 2024. [Online]. Available: <https://repo-sam.inria.fr/fungraph/hierarchical-3d-gaussians/>
- [27] E. Sucar, S. Liu, J. Ortiz, and A. Davison, "iMAP: Implicit mapping and positioning in real-time," in *Proc. Int. Conf. Comput. Vis.*, 2021, pp. 6209–6218.
- [28] Z. Zhu et al., "NICE-SLAM: Neural implicit scalable encoding for SLAM," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2022, pp. 12786–12796.
- [29] 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.*, Jun. 2023, pp. 13293–13302.
- [30] 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.*, Jun. 2023, pp. 17408–17419.
- [31] E. Sandström, K. Ta, L. V. Gool, and M. R. Oswald, "UncLe-SLAM: Uncertainty learning for dense neural SLAM," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. Workshops*, 2023, pp. 4539–4550.
- [32] 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. Automat.*, 2023, pp. 9400–9406.
- [33] N. Keetha et al., "SplaTAM: Splat track & map 3D Gaussians for dense RGB-D SLAM," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2024, pp. 21357–21366.
- [34] C. Yan et al., "GS-SLAM: Dense visual slam with 3D Gaussian splatting," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2024, pp. 19595–19604.
- [35] L. C. Sun et al., "MM3D-GS SLAM: Multi-modal 3D Gaussian splatting for slam using vision, depth, and inertial measurements," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2024, pp. 10159–10166.
- [36] R. Xiao, W. Liu, Y. Chen, and L. Hu, "LIV-GS: LiDAR-vision integration for 3D Gaussian splatting SLAM in outdoor environments," *IEEE Robot. Automat. Lett.*, vol. 10, no. 1, pp. 421–428, Jan. 2025.
- [37] S. Hong, J. He, X. Zheng, C. Zheng, and S. Shen, "LIV-GaussMap: LiDAR-inertial-visual fusion for real-time 3D radiance field map rendering," *IEEE Robot. Automat. Lett.*, vol. 9, no. 11, pp. 9765–9772, Nov. 2024.
- [38] T. Müller, A. Evans, C. Schied, and A. Keller, "Instant neural graphics primitives with a multiresolution hash encoding," *ACM Trans. Graph.*, vol. 41, no. 4, pp. 102–1–102–15, Jul. 2022, doi: [10.1145/3528223](https://doi.org/10.1145/3528223).
- [39] C. Campos, R. Elvira, J. J. Gomez, J. M. M. Montiel, and J. D. Tardos, "ORB-SLAM3: An accurate open-source library for visual, visual-inertial and multi-map SLAM," *IEEE Trans. Robot.*, vol. 37, no. 6, pp. 1874–1890, Dec. 2021.
- [40] C. Jiang et al., "H2-Mapping: Real-time dense mapping using hierarchical hybrid representation," *IEEE Robot. Automat. Lett.*, vol. 8, no. 10, pp. 6787–6794, Oct. 2023.
- [41] C. Jiang, Y. Luo, B. Zhou, and S. Shen, "H3-Mapping: Quasi-heterogeneous feature grids for real-time dense mapping using hierarchical hybrid representation," *IEEE Robot. Automat. Lett.*, vol. 9, no. 10, pp. 9047–9054, Oct. 2024.
- [42] K. Wu, K. Zhang, M. Gao, J. Zhao, Z. Gan, and W. Ding, "Swift-mapping: Online neural implicit dense mapping in urban scenes," in *Proc. AAAI Conf. Artif. Intell.*, 2024, vol. 38, no. 6, pp. 6048–6056.
- [43] X. Guo, P. Han, W. Zhang, and H. Chen, "MotionGS: Compact Gaussian splatting SLAM by motion filter," in *Proc. 7th Int. Conf. Robot., Control Automat. Eng.*, 2024, pp. 685–692.
- [44] Z. Peng et al., "RTG-SLAM: Real-time 3D reconstruction at scale using Gaussian splatting," in *Proc. ACM SIGGRAPH Conf.*, 2024, pp. 1–11.
- [45] K. Ren, et al., "Octree-GS: Towards consistent real-time rendering with LOD-structured 3D Gaussians," *IEEE Trans. Pattern Anal. Mach. Intell.*, to be published in 2025, doi: [10.1109/TPAMI.2025.3568201](https://doi.org/10.1109/TPAMI.2025.3568201)
- [46] Q. Shuai et al., "Real-time view synthesis for large scenes with millions of square meters," *GitHub repository and project page*, 2024. Available at: zju3dv.github.io/LoG_webpage/
- [47] X. Lang et al., "Gaussian-LIC: Real-Time Photo-Realistic SLAM with Gaussian Splatting and LiDAR-Inertial-Camera Fusion," in **Proc. 2025 IEEE Int. Conf. Robot. Automat. (ICRA)**, Available: *arXiv:2404.06926*, doi: [10.48550/arXiv.2404.06926](https://doi.org/10.48550/arXiv.2404.06926)
- [48] 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, Nov. 2024, doi: [10.1145/3687762](https://doi.org/10.1145/3687762).
- [49] C. Jiang, R. Gao, K. Shao, Y. Wang, R. Xiong, and Y. Zhang, "LI-GS: Gaussian splatting with LiDAR incorporated for accurate large-scale reconstruction," *IEEE Robot. Autom. Lett.*, vol. 10, no. 2, pp. 1864–1871, Feb. 2024.
- [50] K. Wu, Z. Zhang, M. Tie, Z. Ai, Z. Gan, and W. Ding, "VINGS-Mono: Visual-inertial Gaussian splatting monocular SLAM in large scenes," 2025, *arXiv:2501.08286*.
- [51] H. Huang, L. Li, C. Hui, 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.*, 2024, pp. 21584–21593.

- [52] H. Matsuki, R. Murai, P. H. J. Kelly, and A. J. Davison, "Gaussian splatting SLAM," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2024, pp. 18039–18048.
- [53] X. Zuo, P. Geneva, W. Lee, Y. Liu, and G. Huang, "LIC-Fusion: LiDAR-inertial-camera odometry," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2019, pp. 5848–5854.
- [54] Y. Zhu, C. Zheng, C. Yuan, X. Huang, and X. Hong, "CamVox: A low-cost and accurate LiDAR-assisted visual SLAM system," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2021, pp. 5049–5055.
- [55] T. Shan, B. Englot, C. Ratti, and D. Rus, "LVI-SAM: Tightly-coupled LiDAR-visual-inertial odometry via smoothing and mapping," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2021, pp. 5692–5698.
- [56] S. Hong, C. Zheng, H. Yin, and S. Shen, "Rollvox: Real-time and high-quality LiDAR colorization with rolling shutter camera," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2023, pp. 7195–7201.
- [57] C. Yuan, W. Xu, X. Liu, X. Hong, and F. Zhang, "Efficient and probabilistic adaptive voxel mapping for accurate online LiDAR odometry," *IEEE Robot. Automat. Lett.*, vol. 7, no. 3, pp. 8518–8525, Jul. 2022.
- [58] M. Zwicker, H. Pfister, J. Van Baar, and M. Gross, "Surface splatting," in *Proc. 28th Annu. Conf. Comput. Graph. Interactive Techn.*, 2001, pp. 371–378.
- [59] B. Jiang and S. Shen, "A LiDAR-inertial odometry with principled uncertainty modeling," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2022, pp. 13292–13299.
- [60] T. Ye, W. Xu, C. Zheng, and Y. Cui, "MFCalib: Single-shot and automatic extrinsic calibration for LiDAR and camera in targetless environments based on multi-feature edge," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2024, pp. 864–871.
- [61] F. Furrer, M. Fehr, T. Novkovic, H. Sommer, I. Gilitschenski, and R. Siegwart, "Evaluation of combined time-offset estimation and hand-eye calibration on robotic datasets," in *Proc. 11th Int. Conf. Field Serv. Robot.*, 2018, pp. 145–159.
- [62] Y. Tao et al., "The Oxford Spires Dataset: Benchmarking Large-Scale LiDAR-Visual Localisation, Reconstruction and Radiance Field Methods," accepted for publication in *Int. J. Robotics Res.*, 2025. Available: [arXiv:2411.10546](https://arxiv.org/abs/2411.10546), doi: [10.48550/arXiv.2411.10546](https://doi.org/10.48550/arXiv.2411.10546)
- [63] N. Huang et al., "S³ Gaussian: Self-supervised street Gaussians for autonomous driving," 2024, *arXiv:2405.20323*.
- [64] J. Liu et al., "Neural Surface Reconstruction and Rendering for LiDAR-Visual Systems," accepted for publication in *Proc. 2025 IEEE Int. Conf. Robot. Automat. (ICRA)*, Available: *arXiv:2409.05310*, doi: [10.48550/arXiv.2409.05310](https://doi.org/10.48550/arXiv.2409.05310)



Sheng Hong received the B.Eng. degree in electrical engineering and automation from the University of Electronic Science and Technology of China, Chengdu, China, in 2015, and the M.Eng. degree in electrical engineering from Xi'an Jiaotong University, Xi'an, China, in 2019. He is currently working toward the Ph.D. degree in robotics and autonomous systems with the Aerial Robotics Group, Hong Kong University of Science and Technology, Hong Kong.

His research interests include simultaneous localization and mapping, multisensor fusion, and 3-D Gaussian splatting.



Chunran Zheng (Member, IEEE) received the B.Eng. degree in automation from Xi'an Jiaotong University, Xi'an, China, in 2020. He is currently working toward the Ph.D. degree in robotics and simultaneous localization and mapping with the Department of Mechanical Engineering, University of Hong Kong, Hong Kong.

His research interests include multisensor calibration, light detection and ranging (LiDAR)-inertial-visual odometry, LiDAR-visual mapping, and 3-D Gaussian splatting.



Yishu Shen is currently working toward the B.Eng. degree in industrial design engineering with Shanghai Jiao Tong University, Shanghai, China.

He is currently working as a Research Intern with the Global Institute of Future Technology, Shanghai Jiao Tong University. His research interests include multisensor fusion and 3-D Gaussian splatting.



Changze Li (Student Member, IEEE) received the B.Eng. degree in control science and engineering from Xidian University, Xi'an, China, in 2019, and the M.S. degree in navigation guidance and control from the Northwestern Polytechnical University, Xi'an, in 2022. He is currently working toward the Ph.D. degree in control science and engineering with Shanghai Jiao Tong University, Shanghai.

He worked as a Research Engineer with the Department of Advanced Driving Solution, Huawei, from 2022 to 2024. His research interests include end-to-end autonomous driving, visual simultaneous localization and mapping, 3-D Gaussian splatting, and deep reinforcement learning.



Fu Zhang (Senior Member, IEEE) received the B.E. degree in automation from the University of Science and Technology of China, Hefei, China, in 2011, and the Ph.D. degree in controls from the University of California, Berkeley, CA, USA, in 2015.

His Ph.D. work focused on self-calibration and control of microrate-integrating gyrosensors. In 2016, he was a Research Assistant Professor in design and control of aerial robotic vehicles, with the Robotics Institute, Hong Kong University of Science and Technology, Hong Kong. From August 2018, he is an

Assistant Professor with the Department of Mechanical Engineering, University of Hong Kong, Hong Kong. His research interests include robotics and controls, with focus on UAV design, navigation, control, and light detection and ranging (LiDAR)-based simultaneous localization and mapping.



Tong Qin (Member, IEEE) received the B.Eng. degree in control science and engineering from Zhejiang University, Hangzhou, China, in 2015, and the Ph.D. degree in electronic and computer engineering from the Department of Electronic and Computer Engineering, the Hong Kong University of Science and Technology, Hong Kong, in 2019.

He worked as a Staff Research Scientist with the Department of Advanced Driving Solution, Huawei, from 2019 to 2023. He is currently working as an Associate Professor with Global Institute of Future

Technology, Shanghai Jiao Tong University, Shanghai, China. His research interests include simultaneous localization and mapping, neural radiance field, machine learning, and autonomous driving.



Shaojie Shen received the B.Eng. degree in electronic engineering from the Hong Kong University of Science and Technology (HKUST), Hong Kong, in 2009, and the M.S. degree in robotics and the Ph.D. degree in electrical and systems engineering from the University of Pennsylvania, Philadelphia, PA, USA, in 2011 and 2014, respectively.

He joined the Department of Electronic and Computer Engineering, HKUST, in 2014, as an Assistant Professor, and was promoted to an Associate Professor in 2020. His research interests include robotics

and aerial robotic vehicles, with focus on state estimation, sensor fusion, computer vision, localization and mapping, and autonomous navigation in complex environments.