Evaluating State Estimation Paradigms for Real-Time 3D Gaussian Splatting

Hojune Kim
Dept. of Aeronautics and Astronautics
Stanford University
Stanford, CA, USA
hojune@stanford.edu

Lars Osterberg
Dept. of Mechanical Engineering
Stanford University
Stanford, CA, USA
larso33@stanford.edu

Nolan Topper
Dept. of Aeronautics and Astronautics
Stanford University
Stanford, CA, USA
ntopper@stanford.edu

***Abstract*—High-fidelity online 3D Gaussian Splatting (3DGS) relies heavily on the quality of robot pose estimates and geometric initialization, yet the exact influence of different SLAM paradigms remains underexplored. In this work, we present a modular, decoupled framework to systematically compare filtering-based (OpenVINS), optimization-based (ORB-SLAM3), and learning-based (VGGT-SLAM) tracking frontends running alongside a unified 3DGS backend. Evaluating on the RPNG AR Table dataset, we find that the feature-based ORB-SLAM3 yields the lowest trajectory error, directly enabling the highest reconstruction fidelity. Crucially, we uncover a counter-intuitive fidelity penalty when utilizing raw RGB-D streams for map seeding; the continuous injection of depth points acts as high-frequency noise that disrupts photometric optimization, causing RGB-only initialization to ultimately yield superior visual quality and faster convergence. Our results demonstrate that online 3DGS benefits more from highly accurate, high-frequency pose tracking and selective geometric priors than from dense, unconstrained depth initialization. We additionally validate a distributed edge-to-server architecture to highlight the feasibility of offloading heavy neural rendering. Code is available at: https://github.com/losterberg3/AA273Project**

***Index Terms*—SLAM, 3D Gaussian Splatting, State Estimation, Robotics, Vision Foundation Models.**

I. Introduction

High-fidelity 3D environment modeling is essential for advanced robotic tasks such as precision landing and semantic scene understanding. While Neural Radiance Fields (NeRFs) [1] introduced photo-realistic implicit representations, their heavy computational cost typically restricts them to offline processing. 3D Gaussian Splatting (3DGS) [2] has emerged as a faster, explicit alternative capable of real-time rendering [3], yet transitioning this technology to online Simultaneous Localization and Mapping (SLAM) remains a significant challenge.

The performance of online 3DGS is inherently sensitive to the quality of robot pose estimates and landmark initialization. In modular "plug-and-play" SLAM architectures, where state estimation is decoupled from the mapping backend, it is often unclear how specific estimator characteristics—such as drift, latency, or noise profiles—affect the final scene quality. This creates a gap in understanding how localization accuracy and geometric initialization affect reconstruction fidelity.

We present a modular evaluation framework for studying how SLAM frontend design affects online 3D Gaussian Splatting. Using a fixed 3DGS backend, we compare filtering-based, optimization-based, and VFM-based tracking frontends in terms of trajectory accuracy, geometric initialization quality, reconstruction fidelity, and computational latency. We further study a distributed edge-to-server deployment setting to quantify when low-power tracking hardware can support high-quality remote digital twinning.

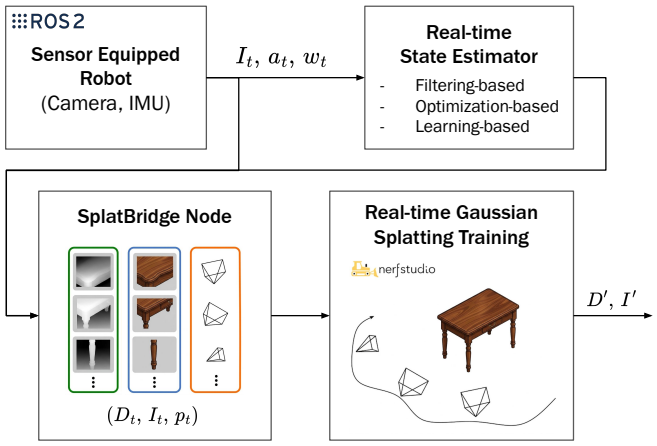

Fig. 1. An overview of the functionality of the SplatBridge for integrating streaming dataset with real-time Gaussian Splatting training.

The main contributions of this work are as follows:

- We present a modular evaluation framework that decouples SLAM frontends from an online 3D Gaussian Splatting backend.
- We compare filtering-based, optimization-based, and VFM-based tracking frontends under a unified reconstruction pipeline.
- We analyze how pose accuracy, geometric prior density, and computational latency jointly affect 3DGS reconstruction quality.
- We demonstrate a distributed edge-to-server deployment workflow for real-time tracking with offloaded dense mapping.

II. Related Work

A. 3D Scene Representations and Online Training

The domain of 3D scene representation has shifted from explicit geometric primitives, such as voxel grids and TSDF fusion [4], to implicit Neural Radiance Fields (NeRF) [1]. While NeRFs offer high-fidelity continuous modeling, their reliance on costly volumetric ray-marching typically restricts them to offline processing. To bridge this gap, recent works have focused on online adaptation: architectures like *iMAP* [5] and *NICE-SLAM* [6] utilize sparse feature grids to accelerate convergence, while frameworks like *NeRFBridge* [7] demonstrate robust system-level integration with robotic middleware (ROS) to enable real-time operation. Despite these advances, implicit MLP-based methods often suffer from limited update rates or "catastrophic forgetting" during trajectory loop closures [8], creating a bottleneck for high-rate maneuvering. These limitations motivate explicit scene representations that can be updated at higher rates under streaming observations.

To overcome these latency issues, 3D Gaussian Splatting (3DGS) [2] was developed, utilizing explicit anisotropic Gaussians for rapid differentiable rasterization. Very recent efforts have begun to adapt 3DGS for SLAM, with systems like *SplaTAM* [9] employing silhouette-guided densification and *GS-SLAM* [10] proposing adaptive pruning strategies. However, these methods typically assume a specific, tightly-coupled tracking frontend and do not systematically evaluate how different state estimation paradigms—filtering, optimization, or learning—influence the convergence and fidelity of the Gaussian map.

B. Filtering-based State Estimation

Filtering-based state estimation applies Recursive Bayesian Estimation to the SLAM problem, updating the probability distribution of the camera pose and map features sequentially with every new image. Unlike optimization approaches that reprocess past keyframes, filtering methods rely on the Markov assumption, maintaining a state vector and a covariance matrix that summarizes the entire history of observations.

A common paradigm for high-rate visual-inertial estimation is the Extended Kalman Filter (EKF), particularly in sliding-window forms such as MSCKF. While early works like *MonoSLAM* [11] modeled the camera and features in a single state vector, they suffered from cubic computational complexity $O(N^3)$. To address this, modern frameworks often utilize the Multi-State Constraint Kalman Filter (MSCKF) architecture, which maintains a sliding window of previous camera poses to perform feature triangulation without adding landmarks to the state vector. We select *OpenVINS* [12] as our baseline to represent the filtering paradigm.

C. Optimization-based SLAM

Optimization-based SLAM approaches formulate localization as a non-linear least squares problem, typically solved via Factor Graphs. Unlike strictly recursive filters, which marginalize past states (fading memory), optimization methods like *PTAM* [13] and its successors maintain a history of keyframes, allowing for re-linearization and global consistency.

Feature-based methods, exemplified by *ORB-SLAM3* [14], extract descriptors (ORB, SIFT) and minimize reprojection error, offering robustness to lighting changes and wide-baseline loop closure. Conversely, direct methods like *LSD-SLAM* [15] and *DSO* [16] operate directly on pixel intensities to optimize geometry. We employ *ORB-SLAM3* in its visual-inertial mode as our optimization baseline due to its strong localization performance and its ability to provide a refined sparse point cloud. This allows us to test if geometric anchors can mitigate the "blurring" artifacts often seen in online neural rendering.

D. Vision Foundation Models and Learning-based SLAM

The integration of deep learning into SLAM has evolved from replacing specific modules, such as *SuperPoint* [17], to end-to-end pose estimation. *Droid-SLAM* [18] represented a breakthrough by utilizing recurrent iterative updates inspired by optical flow. More recently, Vision Foundation Models (VFMs) leveraging Transformer architectures [19] have emerged, with models like *TartanVO* [20] demonstrating that large-scale pre-training enables motion estimation to generalize across diverse domains.

In this work, we use *VGGT-SLAM* [21] as a representative learned dense SLAM frontend. This architecture builds upon the geometric priors established by *DUSt3R* [22] and *StreamVGGT* [23], utilizing Vision Transformers (ViT) [24] to perform joint optimization of camera poses and dense scene geometry. Unlike classical sparse methods, *VGGT-SLAM* provides an early dense geometric prior, which has potential to fix the "cold-start" problem with 3DGS. Future research may further justify the computational cost of transformer-based SLAM by leveraging its dense geometric priors to enable high-fidelity digital twin reconstruction.

III. Methodology

We use a modular framework to evaluate how state estimation affects dense 3D reconstruction. Our system decouples the *Tracking Frontend*, which provides sensor poses and geometric priors, from the *Mapping Backend*, which optimizes a 3D Gaussian Splatting representation.

A. 3D Gaussian Splatting Formulation

The environment is represented by a set of 3D Gaussians $\mathcal{M} = \{\mathcal{G}_i\}_{i=1}^{N}$. Each primitive $\mathcal{G}_i$ is parameterized by a mean $\boldsymbol{\mu}_i \in \mathbb{R}^3$, covariance $\boldsymbol{\Sigma}_i$, opacity α_i, and spherical harmonic coefficients $\mathbf{c}_i$. To enforce physical validity, the covariance is factorized into scaling $\mathbf{S}$ and rotation $\mathbf{R}$:

$$\boldsymbol{\Sigma} = \mathbf{R}\mathbf{S}\mathbf{S}^\top\mathbf{R}^\top \tag{1}$$

Novel views are synthesized via differentiable rasterization. The color $C(\mathbf{u})$ of a pixel $\mathbf{u}$ is computed by alpha-blending K ordered Gaussians overlapping the ray. Let the contributing Gaussians be depth-sorted along the ray.

$$C(\mathbf{u}) = \sum_{i \in \mathcal{K}} c_i \alpha_i \prod_{j=1}^{i-1}(1-\alpha_j) \tag{2}$$

The mapping backend optimizes parameters $\Theta_{\mathcal{M}}$ by minimizing the loss between the rendered image $\hat{I}$ and ground truth I given the estimated pose $\hat{\mathbf{T}}_t$:

$$\mathcal{L}(\Theta_{\mathcal{M}}, \hat{\mathbf{T}}_t) = (1-\lambda)\|\hat{I} - I\|_1 + \lambda \mathcal{L}_{SSIM}(\hat{I}, I) \quad (3)$$

B. State Estimation Baselines

We evaluate three distinct tracking paradigms. To ensure consistent notation, let t denote the current time step, $\mathbf{T}_t \in SE(3)$ the camera pose, and $\mathcal{P}_t$ the set of map landmarks.

1) Filtering: OpenVINS (MSCKF): We adopt the filtering formulation used in OpenVINS, which is based on the Multi-State Constraint Kalman Filter (MSCKF) for visual-inertial odometry. Unlike MonoSLAM-style EKF formulations, the filter does not maintain all observed landmarks directly in the state [11]. Instead, it estimates the current IMU state together with a sliding window of cloned past camera/IMU poses:

$$\mathbf{x}_t = (\mathbf{x}_{I,t}, \mathbf{x}_{C,1}, \ldots, \mathbf{x}_{C,m})^\top \quad (4)$$

where $\mathbf{x}_{I,t}$ contains the current inertial navigation state (typically orientation, position, velocity, and IMU biases), and $\mathbf{x}_{C,j}$ are the cloned poses in the sliding window. The filter therefore maintains the posterior

$$p(\mathbf{x}_t \mid \mathbf{Z}_{1:t}) \quad (5)$$

over the active navigation state and pose history, rather than over an explicit map of all visual features.

When a feature track i has been observed across multiple cloned views, OpenVINS first linearizes the stacked reprojection residual

$$\mathbf{r}_i \approx \mathbf{H}_{x,i}\,\tilde{\mathbf{x}}_t + \mathbf{H}_{f,i}\,\tilde{\mathbf{p}}_i + \mathbf{n}_i \quad (6)$$

where $\tilde{\mathbf{p}}_i$ is the feature perturbation. In the MSCKF update, the feature state is eliminated by nullspace projection, yielding a constraint that depends only on the filter state:

$$\mathbf{r}_{o,i} \approx \mathbf{H}_{o,i}\,\tilde{\mathbf{x}}_t + \mathbf{n}_{o,i}. \quad (7)$$

These multi-view constraints are then incorporated through the standard Kalman correction

$$\hat{\mathbf{x}}_{t|t} = \hat{\mathbf{x}}_{t|t-1} + \mathbf{K}_t \mathbf{r}_{o,i}, \quad (8)$$

with covariance updated accordingly. This provides a strictly causal, uncertainty-aware state estimate while avoiding the cost of explicitly augmenting the filter with every tracked 3D landmark.

2) Optimization: ORB-SLAM3: We use *ORB-SLAM3* [14] as the optimization-based baseline. Unlike strictly recursive filtering methods, *ORB-SLAM3* maintains a set of keyframes and map points and refines them through repeated nonlinear optimization. Its frontend extracts sparse ORB features and establishes correspondences across views, while its backend improves both camera poses and landmark positions through local bundle adjustment and loop-closure correction.

Let $\mathcal{K}$ denote the set of active keyframes and $\mathcal{X} = \{\mathbf{x}_j\}$ the set of reconstructed 3D map points. For an observation of landmark $\mathbf{x}_j$ in keyframe i, the reprojection residual is

$$\mathbf{r}_{ij} = \mathbf{u}_{ij} - \Pi(\mathbf{T}_i, \mathbf{x}_j)\,, \quad (9)$$

where $\mathbf{u}_{ij}$ is the measured image location of feature j in frame i, $\mathbf{T}_i \in SE(3)$ is the camera pose of keyframe i, and $\Pi(\cdot)$ denotes the camera projection model.

The optimization problem can be written in the standard bundle-adjustment form

$$\min_{\{\mathbf{T}_i\},\{\mathbf{x}_j\}} \sum_{(i,j)\in\mathcal{O}} \rho\Big(\|\mathbf{u}_{ij} - \Pi(\mathbf{T}_i, \mathbf{x}_j)\|^2_{\Sigma_{ij}^{-1}}\Big)\,, \quad (10)$$

where $\mathcal{O}$ is the set of valid landmark observations, Σ_{ij} is the associated measurement covariance, and $\rho(\cdot)$ is a robust loss function that reduces the influence of outliers. In practice, *ORB-SLAM3* alternates between tracking, local mapping, and loop-closure optimization to maintain both short-term accuracy and long-term global consistency.

For our study, *ORB-SLAM3* provides two outputs of interest to the 3DGS backend: (i) a refined camera trajectory with reduced drift, and (ii) a sparse geometric landmark set that can be used as an initialization prior.

3) VFM-based SLAM: VGGT-SLAM: To represent the learning-based paradigm, we use *VGGT-SLAM* [21], a vision foundation model-based SLAM frontend that leverages large-scale learned visual priors to infer camera motion and scene geometry. In contrast to classical feature-based pipelines, which rely on hand-crafted keypoints and explicit data association, VGGT-SLAM predicts dense geometric information directly from image observations, enabling reconstruction even in regions with weak texture or ambiguous sparse correspondences.

Given an input image sequence $\{I_t\}$, the learned model produces an estimated camera pose $\hat{\mathbf{T}}_t$ together with a dense geometric representation, which we write abstractly as

$$(\hat{\mathbf{T}}_t, D_t, \mathbf{X}_t) = f_\theta(I_{1:t}), \quad (11)$$

where D_t denotes a dense depth prediction, $\mathbf{X}_t$ denotes dense correspondences or geometric features inferred by the network, and f_θ is the pretrained transformer-based model. Depending on the implementation, these quantities may be refined jointly over a window of frames to improve temporal consistency and reduce drift.

The refinement of the estimated camera poses $\hat{\mathbf{T}}_t$ is performed by optimizing over the $SL(4)$ manifold. Unlike standard $SE(3)$ optimization which is restricted to rigid body transformations, the use of the $SL(4)$ group—representing the set of 4×4 real matrices with unit determinant—allows the system to directly optimize projective transformations and homographies inferred from the dense transformer features. This formulation provides a mathematically principled way to minimize alignment errors between dense feature maps while preserving the projective consistency of the underlying scene geometry.

Rather than modeling the method through sparse landmark reprojection alone, we treat VGGT-SLAM as a frontend that produces a dense geometric prior for downstream mapping. The key distinction from OpenVINS and ORB-SLAM3 is therefore not only the pose estimate itself, but also the density

and spatial coverage of the initialization supplied to the 3DGS backend.

Within our evaluation framework, VGGT-SLAM serves as a learned tracking baseline. Its principal advantage is the ability to provide robustness on weakly featured regions, and to estimate pose purely on RGB images without calibration or IMU measurements. Its principal drawback is higher computational cost and lower update frequency relative to classical filtering and optimization-based pipelines.

C. Evaluation Framework

The system is benchmarked on the RPNG AR Table dataset [25], utilizing the RGB image streams for estimation while using Vicon ground truth for validation. We assess performance using:

- **Localization:** Absolute Trajectory Error (ATE) RMSE:

$$ATE = \sqrt{\frac{1}{N}\sum_{i=1}^{N} \|\text{trans}(\mathbf{T}_{gt,i}^{-1}\hat{\mathbf{T}}_i)\|^2} \tag{12}$$

- **Reconstruction:** Peak Signal-to-Noise Ratio (PSNR) of the rendered view $\hat{I}$ vs. observed image I:

$$PSNR = 10 \cdot \log_{10}\left(\frac{\text{MAX}_I^2}{\text{MSE}(\hat{I}, I)}\right) \tag{13}$$

- **Efficiency:** We measure Pose Estimation Frequency to assess real-time tracking feasibility, and the 3DGS Convergence Step to quantify backend optimization speed. To objectively identify the convergence step, we apply an exponentially weighted moving average to smooth the logged loss values. Convergence is defined as the earliest step where the relative change in the smoothed loss over a fixed lookback window remains strictly below a defined tolerance threshold for a sustained patience duration, marking the onset of the stable optimization plateau.

IV. System Architecture

To evaluate the influence of different state estimation paradigms on 3D Gaussian Splatting, we developed a modular, decoupled architecture. This design allows for the independent execution of the *Tracking Frontend* and the *Mapping Backend*, ensuring that computational bottlenecks in the neural rendering process do not induce artificial drift in the state estimator.

A. Modular Software Pipeline

Our system is built upon the Robot Operating System (ROS 2), utilizing a high-performance publisher-subscriber model to facilitate inter-process communication. The architecture is logically divided into three primary layers:

- **Sensing Layer:** This layer interfaces with the RPNG AR Table Dataset [25]. It streams synchronized 640×480 RGB and Depth images from the Intel RealSense D455 at 30Hz and BMI055 IMU measurements at 400Hz.
- **Estimation Layer (Frontend):** This layer hosts the three candidate paradigms: the EKF-based *OpenVINS*, the factor-graph optimized *ORB-SLAM3*, and the Transformer-based *VGGT-SLAM*. Each estimator processes the raw image stream and publishes a high-frequency transform message $\hat{\mathbf{T}}_t$.
- **Optimization Layer (Backend):** The 3DGS engine subscribes to RGB, Depth images and the pose estimates. It maintains a global Gaussian map, performing adaptive densification and pruning while simultaneously optimizing the spherical harmonic coefficients and anisotropic covariance of the primitives.

B. Hardware Decoupling and Data Logging

Due to the high VRAM requirements of 3DGS training and the GPU-intensive nature of Vision Foundation Models, we adopt a hardware-isolated approach. During the online phase, the state estimator runs on an isolated runtime environment. To ensure scientific reproducibility and strictly evaluate the "real-time" nature of the tracking, all metadata—including camera intrinsics, estimated 6-DOF poses, and timestamped image frames—are serialized into a high-bandwidth *.db3* ROS 2 bag.

This logging strategy allows for a synchronized replay evaluation. By replaying the serialized state estimates on a high-performance workstation, we simulate the real-time arrival of data while maintaining a repeatable environment for benchmarking PSNR and ATE. This ensures that the mapping backend receives the exact trajectory produced by the tracker without the timing jitter associated with running multiple heavy-weight neural networks on a single shared GPU.

C. Initialization and Gaussian Seeding

To evaluate the influence of state estimation accuracy on mapping quality, we test the 3DGS engine across two distinct configurations: RGB-only and RGB-D. In the RGB-only case, the system is initialized without a geometric prior, instead seeding the global map with a sparse point cloud generated from random noise within the initial camera frustum. This forces the backend to rely entirely on the photometric loss and the provided trajectory $\hat{\mathbf{T}}_t$ to recover spatial structure. Conversely, in the RGB-D case, we utilize the depth image from the Intel RealSense D455 to back-project a point cloud for the first frame, providing an initial geometric seed for the Gaussian primitives. This dual-track approach allows us to quantify the fidelity penalty of operating without active depth sensing and isolates whether certain tracking paradigms are more resilient to the lack of an initial geometric anchor.

D. Hardware Diversity and Resource Constraints

A unique aspect of our experimental setup is the heterogeneous computing environment, which reflects a realistic robotics deployment where state estimation occurs on low-power edge devices while mapping is offloaded to high-performance servers. Notably, the filtering frontend (*OpenVINS*) was executed on a resource-constrained Apple M1 Silicon mobile processor. This allows us to quantify the

precision-power trade-off—specifically, the trajectory drift incurred when using low-power hardware for real-time tracking—relative to the high-performance desktop CPUs used for the optimization and VFM-based frontends.

V. Implementation Details

The experimental validation is conducted across a distributed computing setup. The hardware and software specifications, including the specific mobile and desktop processors used for each module, are detailed in Table I.

TABLE I
System Implementation and Multi-Computer Specifications

Component	Specification
Operating System	Ubuntu 22.04.3 LTS / macOS (Linux VM)
Middleware	ROS 2 Humble Hawksbill
GPU (VGGT)	NVIDIA GeForce RTX 4090 (24GB VRAM)
GPU (SplatBridge)	NVIDIA GeForce RTX 3090 (24GB VRAM)
CPU (VGGT/ORB)	Intel Core i7-14700 @ 5.4 GHz
CPU (OpenVINS)	Apple M1 Silicon @ 3.2 GHz
CPU (SplatBridge)	Intel Core i9-13900K @ 5.8 GHz
CUDA Version	11.8 / 12.1
3DGS Backend	Nerfstudio / SplatBridge
Dataset Format	RPNG / ROS 2 Bag (MCAP)

VI. Results and Discussion

We evaluate our framework on the RPNG AR Table Dataset, leveraging the heterogeneous computing environment described in Section IV. Our analysis focuses on the relationship between frontend design, initialization density, and the resulting 3DGS map fidelity.

A. Comparative Tracking and Localization

The three frontends exhibit clear differences in localization accuracy and update rate. To quantify global consistency, we computed the RMSE of the translational APE following a Sim(3) Umeyama alignment. These metrics were visualized alongside error plots to distinguish between linear and angular inaccuracies.

TABLE II
Localization Metrics on RPNG AR Table Dataset

Metric	OpenVINS	ORB-SLAM3	VGGT-SLAM
ATE RMSE	0.047 m	**0.014 m**	0.051 m
Frequency (Hz)	**30 Hz**	**30 Hz**	4 Hz

1) Filtering Performance (Edge Tier): The EKF-based *OpenVINS* maintained a stable trajectory despite the resource-constrained 3.2 GHz mobile processor on the 2020 MacBook Pro. As seen in Table II, it achieved a positional RMSE of 0.047m at 30Hz, suggesting that optimized C++ implementations can provide reliable VIO on dual-core ARM silicon, though it exhibited minor drift over the longer 80s sequences.

Fig. 2. Position and Orientation Error Plots

2) Optimization Performance (Desktop Tier): The feature-based *ORB-SLAM3* was the most accurate among the evaluated frontends on the tested sequences. Leveraging the workstation i7 CPU for continuous bundle adjustment, ORB-SLAM3 achieved an RMSE of $\mathbf{0.014}$m. The position error suggests that ORB-SLAM3 maintained sub-5cm error across 100% of the trajectory.

3) VFM Performance (Workstation Tier): The transformer-based *VGGT-SLAM* demonstrated competitive performance (Position RMSE of 0.051m) but was limited by its inference latency. While executing on the workstation tiers, the model only achieved an update rate of 4 Hz, which is insufficient for high-speed robotic maneuvering but useful for dense mapping initialization.

B. Impact of Geometric Initialization

To isolate the mapping backend's dependency on geometric priors, we evaluated the resulting 3DGS maps initialized via purely stochastic means against those seeded with raw depth sensor data. For our empirical analysis, the convergence step reported in our metrics is calculated using an exponential smoothing factor of $\alpha = 0.05$, a lookback window of 3000 steps, a relative change tolerance of 3%, and a sustained patience duration of 250 steps.

TABLE III
Reconstruction Quality Under Different Initialization Strategies

Input	State Estimator	PSNR (dB)	Converged Step
RGB	OpenVINS	19.80	11740
	VGGT-SLAM	18.10	**6750**
	ORB-SLAM3	**23.68**	10130
RGB+Depth	OpenVINS	18.69	16850
	VGGT-SLAM	18.53	16410
	ORB-SLAM3	20.75	16370

1) RGB-Only Initialization: When forced to rely entirely on the state estimator's poses and photometric loss, the feature-based trajectory from ORB-SLAM3 yielded the highest re-

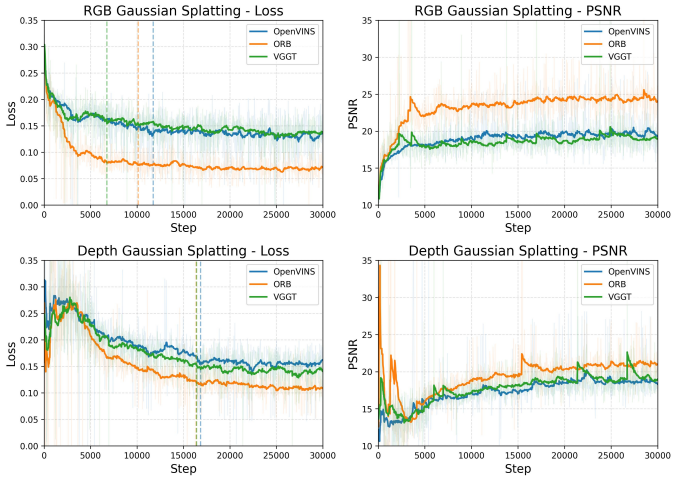

Fig. 3. Gaussian Splatting Loss and PSNR for Each Estimation Method. Vertical dotted lines indicate the automatically calculated convergence step for each tracking paradigm.

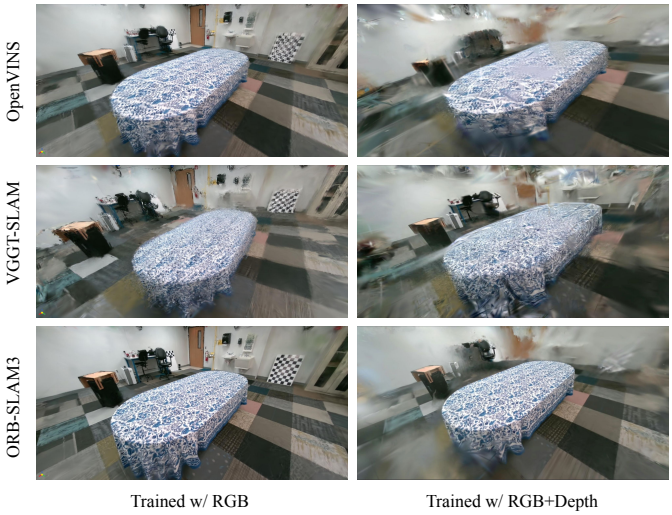

Fig. 4. Representative 3DGS renderings initialized from OpenVINS (top), VGGT-SLAM (middle), and ORB-SLAM3 (bottom). The left column displays models trained strictly with RGB inputs, while the right column shows models initialized with RGB+Depth prior.

construction fidelity. As detailed in Table III, ORB-SLAM3 achieved a peak PSNR of 23.68 dB at its calculated convergence step. The loss curves in Fig. 3 corroborate this, demonstrating that ORB-SLAM3 consistently maintained the lowest loss throughout the optimization process. The vertical dotted lines superimposed on these plots explicitly mark the convergence step for each respective method, visualizing the point at which the map structure stabilized. Conversely, while the transformer-based VGGT-SLAM produced the lowest overall PSNR (18.10 dB), it exhibited the fastest initial convergence, stabilizing at step 6750. We hypothesize that this rapid, yet suboptimal, convergence is a direct artifact of VGGT-SLAM's lower update frequency (4 Hz). Because the mapping backend receives fewer tracking poses, the 3DGS optimizer is subjected to fewer multi-view constraints, causing it to quickly settle into a local minimum that lacks the geometric refinement needed for higher-fidelity reconstruction.

2) The Depth-Seeding Penalty: Contrary to the standard intuition that an explicit geometric prior accelerates convergence, the introduction of depth-guided seeding resulted in a distinct "fidelity penalty" across all tracking frontends. To contextualize this behavior, it is important to note that the 80s dataset streaming concludes at approximately step 5000, after which we extend the training iterations strictly to observe the final convergence of the global map.

Table III shows a universal decrease in converged PSNR and a notable rightward shift in the required optimization steps (denoted by the dotted lines in Fig. 3) when depth is introduced. This penalty occurs because, during the initial streaming phase (steps 0 to 5000), back-projecting the raw depth map continually introduces novel points to seed new Gaussians. Because these new primitives are instantiated without a well-defined prior covariance, they act as high-frequency noise that actively disrupts the photometric optimization. As seen in the depth-based loss curves, the loss struggles to descend while the dataset stream is active. Once the stream concludes at step 5000 and the injection of noisy depth points ceases, the 3DGS engine must expend substantial computational iterations attempting to correct and prune these poorly initialized primitives, ultimately leading to delayed convergence and degraded final visual quality.

3) Qualitative Evaluation: The quantitative metrics are directly reflected in the qualitative renderings presented in Fig. 4. The maps optimized strictly using RGB inputs (left column) display sharper structural boundaries and finer preservation of high-frequency details, such as the checkerboard calibration target and the patterned tablecloth. In contrast, the renderings resulting from RGB+Depth initialization (right column) exhibit noticeable cloudiness, floaters, and blurring artifacts. Among the estimators, the map generated using ORB-SLAM3's trajectory (bottom row) provided the clearest visual reconstruction, aligning with its superior tracking accuracy and sub-5cm positional error.

C. Edge-to-Server Offloading

Our hardware-isolated architecture also enabled an edge-to-server offloading workflow. By logging the low-power *Apple M1* trajectory to a ROS bag and replaying it through *SplatBridge* on the *RTX 3090/4090* workstation, we observed reconstruction quality comparable to that of a desktop-only mapping configuration. These results suggest that edge devices can focus on low-latency localization while the more computationally intensive 3DGS optimization is offloaded to higher-performance hardware.

VII. Conclusion and Future Work

In this paper, we presented a modular, decoupled framework to systematically evaluate the impact of state estimation paradigms on the fidelity of online 3D Gaussian Splatting. By isolating the tracking frontend from the mapping backend, we

demonstrated that the underlying trajectory's accuracy and the mapping engine's initialization strategy are critical drivers of final scene reconstruction quality.

Our comparative analysis revealed that the optimization-based *ORB-SLAM3* provided the most accurate camera poses, directly enabling the highest reconstruction fidelity (23.68 dB PSNR) when relying purely on photometric optimization. We also demonstrated the feasibility of a distributed computing workflow, showing that filtering-based methods like *Open-VINS* can maintain robust, high-frequency localization on low-power edge silicon while the demanding 3DGS optimization is successfully offloaded to a remote high-performance server. While the VFM-based *VGGT-SLAM* offered dense geometric tracking, its high inference latency restricted the update rate, leading the 3DGS optimizer to prematurely settle into suboptimal local minima.

Most notably, our experiments uncovered a significant fidelity penalty associated with raw depth-guided initialization. Contrary to the assumption that explicit geometric priors universally improve neural rendering, the continuous injection of un-covarianced depth points during the online streaming phase acted as high-frequency noise. This disrupted the photometric loss landscape, forcing the 3DGS engine to expend substantial computational effort correcting poorly initialized primitives rather than refining global scene structure.

Future work will focus on mitigating this initialization penalty. A highly promising avenue is to transition away from naively seeding the entire raw depth map and instead selectively initialize new Gaussians using only the novel, triangulated landmarks actively tracked and validated by the state estimator. By leveraging these sparse but highly reliable geometric priors, the system can avoid flooding the optimization landscape with unconstrained noise. Additionally, we plan to explore dynamic, uncertainty-aware covariance scaling for these newly seeded points and investigate adaptive pruning strategies that can more aggressively remove volatile primitives during the active streaming phase. Resolving these bottlenecks will be crucial for fully harnessing geometric priors in high-speed, online 3DGS applications.

REFERENCES

[1] 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," *Communications of the ACM*, vol. 65, no. 1, pp. 99–106, 2021.

[2] B. Kerbl, G. Kopanas, T. Leimkühler, and G. Drettakis, "3d gaussian splatting for real-time radiance field rendering," *ACM Transactions on Graphics*, vol. 42, no. 4, pp. 1–14, 2023.

[3] H. Matsuki, R. Murai, P. H. Kelly, and A. J. Davison, "Gaussian splatting slam," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 18 039–18 048.

[4] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison *et al.*, "Kinectfusion: Real-time 3d reconstruction and interaction using a moving depth camera," in *Proceedings of the 24th annual ACM symposium on User interface software and technology*, 2011, pp. 559–568.

[5] E. Sucar, S. Liu, J. Ortiz, and A. J. Davison, "imap: Implicit mapping and positioning in real-time," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 6229–6238.

[6] Z. Zhu, S. Peng, V. Larsson, W. Xu, H. Bao, Z. Cui, M. Pollefeys, and A. Geiger, "Nice-slam: Neural implicit scalable encoding for slam," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 12 786–12 796.

[7] D. Maggio, J. Yu, J. E. Low, K. Nagami, and M. Schwager, "Nerf-bridge: Bringing neural radiance fields to robotics," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 1–7.

[8] T. Müller, A. Evans, C. Schied, and A. Keller, "Instant neural graphics primitives with a multiresolution hash encoding," *ACM Transactions on Graphics (ToG)*, vol. 41, no. 4, pp. 1–15, 2022.

[9] N. Keetha, J. Karhade, K. M. Jatavallabhula, G. Yang, S. Scherer, D. Ramanan, and J. Luiten, "Splatam: Splat, track & map 3d gaussians for dense rgb-d slam," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 21 357–21 366.

[10] C. Yan, D. Qu, D. Xu, B. Zhao, Z. Wang, D. Dong, and X. Wang, "Gs-slam: Dense visual-slam with 3d gaussian splatting," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 6334–6343.

[11] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse, "Monoslam: Real-time single camera slam," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 6, pp. 1052–1067, 2007.

[12] P. Geneva, K. Eckenhoff, L. Woosik, Y. Yang, and G. Huang, "Open-VINS: A research platform for visual-inertial estimation," *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, 2020.

[13] G. Klein and D. Murray, "Parallel tracking and mapping for small ar workspaces," in *2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*. IEEE, 2007, pp. 225–234.

[14] C. Campos, R. Elvira, J. J. Gómez, J. M. Montiel, and J. D. Tardós, "Orb-slam3: An accurate open-source library for visual, visual–inertial, and multi-map slam," *IEEE Transactions on Robotics*, vol. 37, no. 6, pp. 1874–1890, 2021.

[15] J. Engel, T. Schöps, and D. Cremers, "Lsd-slam: Large-scale direct monocular slam," in *European Conference on Computer Vision*. Springer, 2014, pp. 834–849.

[16] J. Engel, V. Koltun, and D. Cremers, "Direct sparse odometry," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 3, pp. 611–625, 2017.

[17] D. DeTone, T. Malisiewicz, and A. Rabinovich, "Superpoint: Self-supervised interest point detection and description," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2018, pp. 224–236.

[18] Z. Teed and J. Deng, "Droid-slam: Deep visual slam for monocular, stereo, and rgb-d cameras," *Advances in Neural Information Processing Systems*, vol. 34, pp. 16 558–16 569, 2021.

[19] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, 2017, pp. 5998–6008.

[20] W. Wang, Y. Hu, and S. Scherer, "Tartanvo: A generalizable learning-based vo," in *Conference on Robot Learning*. PMLR, 2020, pp. 1761–1772.

[21] D. Maggio, H. Lim, and L. Carlone, "Vggt-slam: Dense rgb slam optimized on the sl(4) manifold," 2025. [Online]. Available: https://arxiv.org/abs/2505.12549

[22] S. Wang, C. Rupprecht, and A. Vedaldi, "Dust3r: Geometric 3d vision made easy," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 20 697–20 709.

[23] D. Zhuo, W. Zheng, J. Guo, Y. Wu, J. Zhou, and J. Lu, "Streaming 4d visual geometry transformer," *arXiv preprint arXiv:2507.11539*, 2025.

[24] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," in *International Conference on Learning Representations*, 2021.

[25] C. Chen, P. Geneva, Y. Peng, W. Lee, and G. Huang, "Monocular visual-inertial odometry with planar regularities," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 11 183–11 189.