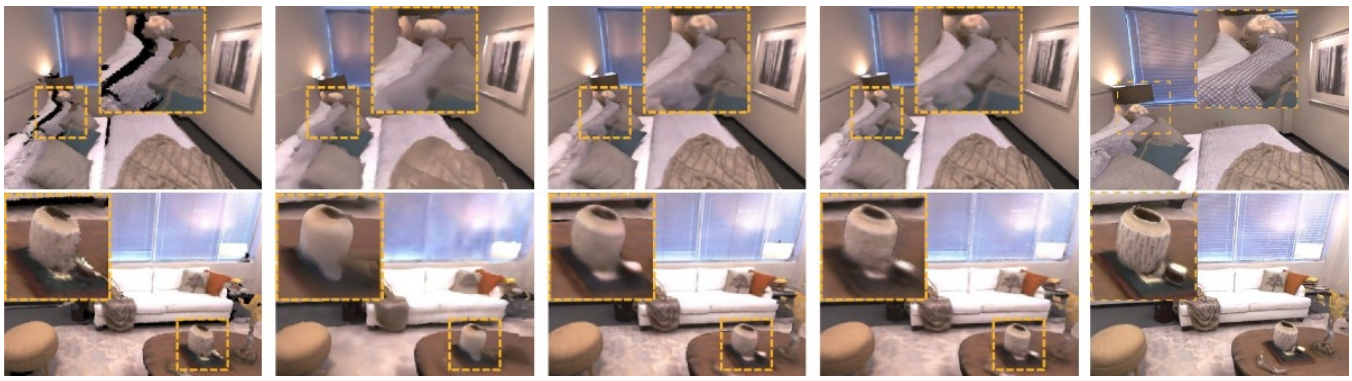
# Geometry-Aware Online Mapping for 3D Gaussian Splatting SLAM

Thai Luu[1] Quan Tran[2] Hieu Phan[1] Tuan Dang*[1]



**(a) BundleFusion [1]** **(b) Nice-SLAM [2]** **(c) ESLAM [3]** **(d) Co-SLAM [4]** **(e) Ours**

**Fig. 1: Qualitative comparison of diverse SLAM systems on the Replica dataset.** As highlighted in the zoomed-in regions, existing methods often suffer from blurry artifacts or incomplete geometry. In contrast, our approach preserves finer details and high-frequency textures (e.g., the patterns on the vase and bedsheets), resulting in superior photorealism.

***Abstract*— Recent 3D Gaussian Splatting (3DGS) has enabled efficient photorealistic view synthesis and is rapidly being adopted in simultaneous localization and mapping (SLAM) systems for online mapping. In these systems, a Gaussian map must be expanded and refined incrementally while tracking runs in real time, so initialization and density control directly determine where limited computation and iterations are spent. This contrasts with offline 3DGS reconstruction, where such heuristics can be amortized over long optimization schedules. However, most 3DGS-SLAM pipelines inherit initialization and density-control heuristics from offline reconstruction, which can become brittle under the strict per-keyframe optimization budgets and incremental map growth of online SLAM. In this work, we revisit these heuristics in a decoupled 3DGS-SLAM setting and propose three geometry-aware methods that operate in the mapping thread: transmittance-preserving densification, camera-aware scale initialization from depth and intrinsics, and error-guided densification that focuses new primitives on high-residual regions. Our results show consistent improvements in rendering quality with negligible overhead, highlighting the coupling between photometric residuals and pose uncertainty in online SLAM. We will open-source our code to the community to foster growth and validate reproducibility.**

## I. Introduction

Simultaneous Localization and Mapping (SLAM) is a core capability for robots and augmented reality (AR) devices: it provides geometric and camera-trajectory information, enabling navigation, interaction, and scene understanding [5], [6], [7], [8]. Recently, the community has also pursued *photorealistic* online mapping, where the map supports high-fidelity view synthesis in addition to localization. A major catalyst is 3DGS [9], which represents a scene as anisotropic Gaussians and renders them efficiently via differentiable rasterization, achieving high-quality real-time novel-view synthesis. This has quickly led to 3DGS-based SLAM systems such as Photo-SLAM [10], MonoGS [11], SplaTAM [12], and GS-SLAM [13].

Despite this momentum, most 3DGS-SLAM systems inherit key *initialization* and *density-control* heuristics from the original offline 3DGS pipeline [9]. However, recent studies highlight that these original heuristics lead to uncontrolled primitive growth and systematic opacity biases even in offline scenarios [14]. Consequently, they are fundamentally ill-suited for the strict optimization budgets, real-time computational constraints, and incremental map growth inherent to online SLAM [11], [12]. First, adaptive density control relies on *cloning* Gaussians in under-reconstructed regions based on view-space gradients [9]. However, repeated cloning systematically biases alpha compositing and causes opacity drift [14]. Specifically, because the original pipeline simply duplicates a primitive while preserving its original opacity, the resulting overlapping Gaussians artificially over-occlude the background during rendering [14]. This double-counting introduces a systematic bias that inflates the photometric contribution of newly cloned Gaussians, leading to localized opacity drift and hindering accurate optimization of the underlying scene structures. Second, purely gradient-triggered densification is *passive*: with only hundreds of mapping iterations per keyframe in online SLAM (versus tens of thousands of iterations typical for offline reconstruction), persistently under-modeled regions may never accumulate gradients strong enough to attract new primitives. These effects are

[1]Cognitive Robotics Lab, Department of Electrical Engineering and Computer Science, University of Arkansas, USA.

[2]Center for AI Research, VinUniversity, Ha Noi, Vietnam.

*Corresponding author: tuand@uark.edu

subtle but compounding, and they disproportionately affect online mapping quality.

In this paper, we revisit these heuristics under the online SLAM regime and propose geometry-aware methods that operate *only* in the mapping thread. As shown in Fig. 1, our method significantly increases the quality of existing SLAM benchmarks by accurately reconstructing high-frequency textures and fine geometric details. We build on the decoupled design of Photo-SLAM [10], where ORB-SLAM3 [8] performs real-time tracking while a parallel thread optimizes a Gaussian map for photorealistic rendering. We observe that under strict iteration budgets, rendering quality is dominated by (1) *where* new primitives are inserted, (2) *how* their initial scale reflects camera geometry rather than point density, and (3) whether densification preserves the intended alpha-compositing behavior. We identify three offline 3DGS heuristics that are brittle in online SLAM and propose three simple, geometry-aware fixes that modify mapping only and introduce no learned components. Our contributions are :

- We introduce **Transmittance-Preserving Densification** to preserve composite opacity under cloning, preventing systematic opacity inflation under alpha compositing during online densification.
- We introduce **Camera-aware Scale initialization** to initialize Gaussian size from the metric pixel footprint at the observed depth (using camera intrinsics and depth).
- We introduce **Error-guided Densification** to allocate new primitives to persistently high-residual image regions by sampling an error map and spawning Gaussians via depth-backed back-projection.

## II. Related Work

**Neural rendering for dense SLAM.** NeRF-based dense SLAM methods jointly optimize camera poses and a neural scene representation online [15], [16], [2], [4], [3]. While these implicit fields enable high-quality view synthesis, ray-based volume rendering and per-frame backpropagation remain expensive, so practical systems often trade off resolution by sampling only a subset of pixels/rays during optimization [16], [2], [4], [3]. The 3DGS instead represents the scene with explicit anisotropic Gaussians and optimizes them through differentiable splatting, enabling efficient high-resolution rendering and gradients [9]. Recent 3DGS-SLAM systems differ mainly in how tightly tracking is coupled to Gaussian optimization: MonoGS [11], SplaTAM [12], and GS-SLAM [13] directly track against the Gaussian map (with different robustness and speed–accuracy strategies), whereas Photo-SLAM [10] follows a decoupled design with feature-based tracking running in parallel with an asynchronous photorealistic Gaussian mapping thread. We adopt this decoupled backbone to isolate mapping-side improvements under real-time tracking constraints.

**Online Gaussian map growth: initialization, densification, and evaluation.** The original 3DGS reconstruction pipeline initializes primitives from sparse points (with k-NN-style scale heuristics) and expands representation capacity through Adaptive Density Control (ADC), which clones/splits Gaussians based on view-space gradient statistics while pruning low-opacity primitives [9]. However, recent analyses show that naive cloning can bias alpha compositing by increasing opacity in repeatedly densified regions, and that purely gradient-threshold densification may fail to allocate capacity to persistently under-modeled high-frequency regions; error-driven density control and opacity-aware corrections can mitigate these issues [14]. ConeGS [17] further explores error-guided primitive insertion and initializes sizes from the pixel viewing footprint (inspired by mip-NeRF-style cones/frustums [18]), [19], but is studied primarily in offline reconstruction settings with auxiliary depth proxies. The ConeGS model is inspired by the iNGP architecture [20], adapting its efficiency to the 3D Gaussian Splatting (3D GS) framework. In online SLAM, map growth is incremental and optimization per keyframe is tightly budgeted, making initialization and densification decisions disproportionately important. Our work targets online RGB-D 3DGS-SLAM in this regime: keeping the decoupled SLAM backbone unchanged, we introduce geometry-aware, mapping-only methods that stabilize densification under alpha compositing, initialize scales from metric pixel footprints using depth and intrinsics, and allocate new primitives to persistently high-residual regions under short-horizon optimization. Finally, because rendering metrics on training keyframes can be overly optimistic due to overfitting, we follow prior recommendations and report rendering quality along the full trajectory rather than keyframe-only train-view evaluation [12].

## III. Methodology

### A. System Overview

Fig. 2 illustrates our decoupled 3DGS-SLAM pipeline. Building upon the Photo-SLAM framework [10], our system operates via two parallel threads. The real-time tracking frontend (ORB-SLAM3, RGB-D mode) processes incoming frames to estimate camera poses $\mathbf{T}_{wc}^{t}$ and select keyframes. Concurrently, the asynchronous mapping backend optimizes a 3DGS map by rendering from keyframe poses and minimizing a photometric reconstruction loss. While the tracking stack, rendering model, and mapping objective remain unchanged from the baseline, we redesign the density-control and initialization mechanisms to operate effectively under a tight online computational budget. Our mapping-only pipeline integrates three key modules directly into the optimization loop. The following sections detail the mathematical formulation of the baseline and our proposed modifications.

### B. Problem Formulation

We build on the decoupled architecture of Photo-SLAM [10], where a real-time tracking thread (ORB-SLAM3 [8]) estimates camera poses, and a parallel mapping thread maintains a 3D Gaussian map optimized through differentiable splatting. Let $t \in \mathcal{K}$ index keyframes selected by the tracking frontend. We denote the RGB observation by

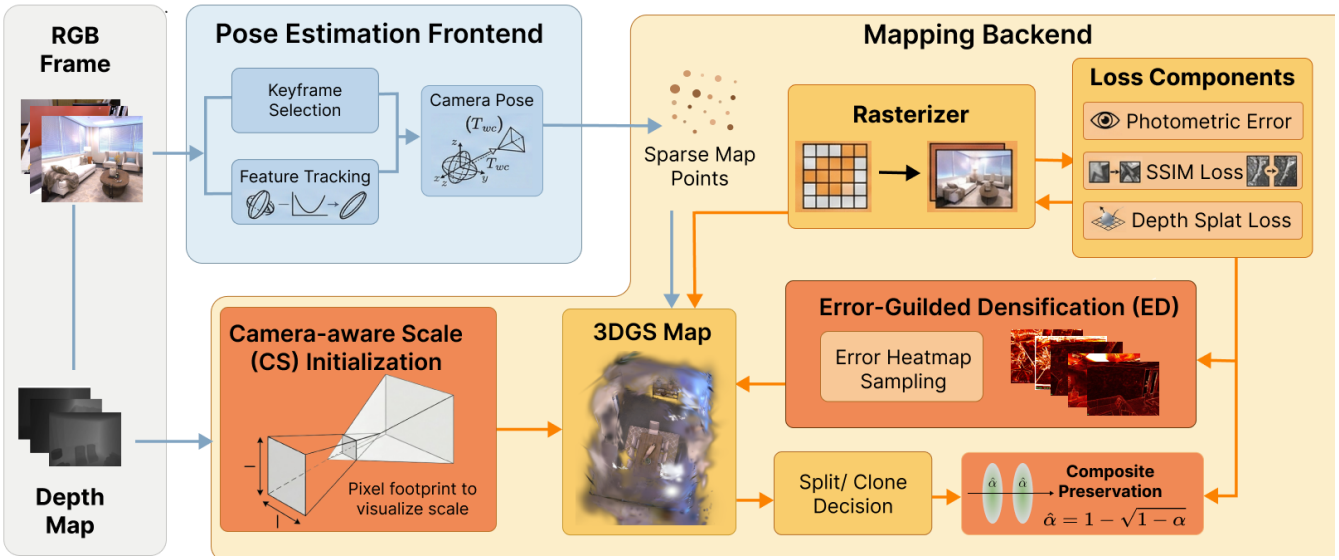



**Fig. 2: System overview.** Our pipeline follows Photo-SLAM's decoupled design: ORB-SLAM3 estimates poses and selects keyframes, while an asynchronous backend optimizes a 3D Gaussian Splatting map via differentiable rasterization. mapping modules: Camera-aware Scale initialization, Transmittance-Preserving Densification, and Error-guided Densification; the tracking frontend remains unchanged.

$I_t$, camera intrinsics by $\mathbf{K}$, and the world-from-camera pose by $\mathbf{T}_{wc}^t \in SE(3)$ (with $\mathbf{T}_{cw}^t = (\mathbf{T}_{wc}^t)^{-1}$).

Following standard 3DGS [9], the map is a set of Gaussian primitives

$$\mathcal{G} = \{g_i\}_{i=1}^{M}, \tag{1}$$

where each primitive $i$ is parameterized by a 3D mean $\boldsymbol{\mu}_i$, an anisotropic covariance $\boldsymbol{\Sigma}_i$, an opacity $\alpha_i$, and appearance coefficients (e.g., spherical harmonics). Given $\mathcal{G}$, $\mathbf{K}$, and a keyframe pose $\mathbf{T}_{cw}^t$, the differentiable splatting renderer $\mathcal{R}$ produces a rendered image:

$$\hat{I}_t = \mathcal{R}\big(\mathcal{G}; \mathbf{T}_{cw}^t, \mathbf{K}\big)\,. \tag{2}$$

Mapping then minimizes a photometric reconstruction objective over keyframes

$$\mathcal{L}_{\text{photo}}(\mathcal{G}) = \sum_{t\in\mathcal{K}} \left\| \hat{I}_t - I_t \right\|_1, \tag{3}$$

interleaved with *adaptive density control* that clones/splits Gaussians based on view-space position gradients and prunes low-opacity primitives [9]. Abstractly, one mapping iteration can be written as

$$\mathcal{G}^{(k+\frac{1}{2})} = \mathcal{G}^{(k)} - \eta \nabla_{\mathcal{G}} \mathcal{L}_{\text{photo}}(\mathcal{G}^{(k)}), \tag{4}$$

$$\mathcal{G}^{(k+1)} = \mathcal{D}_{\star}\Big(\mathcal{G}^{(k+\frac{1}{2})}\Big)\,, \tag{5}$$

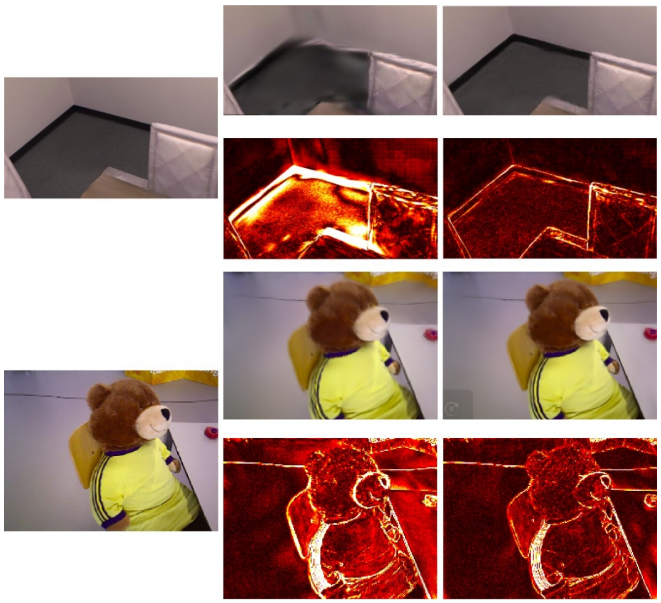
where $\mathcal{D}_{\star}$ denotes the density-control operator.

In Photo-SLAM [10], the decoupled design means that tracking provides $\{\mathbf{T}_{wc}^t\}_{t\in\mathcal{K}}$ in real time, and the mapping $G$ using Eqs. (3)–(5) with poses treated as fixed inputs. In offline 3DGS, initialization and density-control heuristics are amortized over long optimization schedules. In online SLAM, mapping is instead constrained by limited wall-clock compute. In the Photo-SLAM implementation, each mapping iteration performs a single optimization step on one selected active keyframe; hence, the effective number of updates received by each keyframe is stochastic and run-dependent, rather than a fixed per-keyframe budget. As a result, early design choices disproportionately affect where representational capacity is allocated and how optimization gradients evolve over time. In particular, (i) neighbor-distance scale initialization ties Gaussian size to local point density rather than scene geometry, (ii) cloning operations can introduce systematic opacity inflation under alpha compositing, and (iii) purely gradient-triggered densification may fail to populate persistently under-modeled regions within a tight iteration budget.

We observe that, under a tight online mapping budget, rendering quality is governed less by learning dynamics and more by three "non-learning" design choices: *where* new Gaussians are inserted, *how* they are initially scaled in metric space, and whether density-control operations preserve the intended alpha-compositing behavior. Accordingly, we introduce three contributions that operate in the mapping thread. Formally, we keep the tracking frontend (ORB-SLAM3), the renderer $\mathcal{R}(\cdot)$, and the objective $\mathcal{L}_{\text{photo}}(\cdot)$ unchanged. We simply replace the baseline density-control operator $\mathcal{D}_{\text{base}}$ in Eq. (5) with our proposed operator $\mathcal{D}_{\text{ours}}$. Both operators act on the space of Gaussian maps $\mathcal{G}$ where $\mathcal{D}_{\text{ours}}$ differs from the baseline through the three mapping-side mechanisms introduced in Sec. III-C, Sec. III-D, and Sec. III-E.

### C. Error-guided Densification (ED)

Standard 3DGS performs adaptive density control by cloning/splitting Gaussians whose *average view-space positional gradient magnitude* exceeds a threshold [9]. This signal is effective in offline reconstruction, where long optimization schedules allow photometric errors to propagate



**a) Ground truth** **b) Photo-SLAM** **c) Ours**

**Fig. 3: Effect of Error-guided Densification under the same online mapping budget.** For each example, we show the observed keyframe (left), the baseline gradient-based densification (middle), and ours with ED (right). The residual heatmaps is brighter indicates larger error), where ED reduces holes and over-smoothed regions.

into gradients and eventually trigger densification near under-modeled regions. In online decoupled SLAM, however, each keyframe receives only a limited number of mapping updates (Sec. IV-A), so regions with insufficient primitive coverage may not *exhibit sufficiently large* view-space positional gradients early enough, leading to persistent holes or over-smoothed surfaces as shown in Fig. 3.

**Error-guided insertion.** We therefore add an auxiliary densification operator that directly spawns new Gaussians at high-residual pixels. Let keyframe $t$ provide an RGB observation $\mathbf{I}_t$ and a depth map $\mathbf{D}_t$ (RGB-D setting), with intrinsics $\mathbf{K}$ and a pose estimate $\mathbf{T}_{cw}^t \in SE(3)$ (world $\rightarrow$ camera) from the tracking frontend; we denote its inverse by $\mathbf{T}_{wc}^t = (\mathbf{T}_{cw}^t)^{-1}$. Rendering the current Gaussian map $\mathcal{G}$ at pose $\mathbf{T}_{cw}^t$ yields $\hat{\mathbf{I}}_t = \mathcal{R}(\mathcal{G}; \mathbf{T}_{cw}^t, \mathbf{K})$. Every $K_{\text{ED}}$ mapping iterations, we compute the per-pixel photometric residual

$$e_t(u,v) = \left\| \hat{\mathbf{I}}_t(u,v) - \mathbf{I}_t(u,v) \right\|_1 . \tag{6}$$

$$\mathcal{V}_t = \{(u,v) \mid e_t(u,v) > \tau,\ \mathbf{D}_t(u,v) \text{ valid},\ g_t(u,v) < \delta\}, \tag{7}$$

where $g_t(u,v)$ is a depth-edge score used to suppress depth discontinuities, so that we avoid inserting Gaussians on depth edges. In practice we use a simple finite-difference filter, $g_t(u,v) = \max\{|\mathbf{D}_t(u{+}1,v) - \mathbf{D}_t(u,v)|,\ |\mathbf{D}_t(u,v{+}1) - \mathbf{D}_t(u,v)|\}$. We sample $N$ pixels $(u_i, v_i) \sim \text{Cat}(\pi_t)$ from candidate set $\mathcal{V}_t$ by importance sampling on the residual heatmap:

$$\pi_t(u,v) \propto e_t(u,v), \quad (u,v) \in \mathcal{V}_t. \tag{8}$$

For each sampled pixel, we back-project using depth $d_i = \mathbf{D}_t(u_i, v_i)$:

$$\tilde{\mathbf{p}}_{c,i} = [\mathbf{p}_{c,i}^\top,\ 1]^\top, \qquad \tilde{\boldsymbol{\mu}}_i = \mathbf{T}_{wc}^t \tilde{\mathbf{p}}_{c,i}, \qquad \boldsymbol{\mu}_i = \tilde{\boldsymbol{\mu}}_{i,1:3}. \tag{9}$$

and spawn a new Gaussian with mean $\boldsymbol{\mu}_i$, color initialized from $\mathbf{I}_t(u_i, v_i)$, and a conservative opacity $\alpha_0$. We initialize its covariance isotropically as $\boldsymbol{\Sigma}_i = s_{\text{cam}}(d_i)^2 \mathbf{I}$ using the camera-aware scale rule in Sec. III-D, so that newly inserted primitives roughly match the pixel footprint at depth $d_i$.

$\mathbf{p}_{c,i} \in \mathbb{R}^3$ is the 3D point in camera coordinates obtained by back-projecting sampled pixel $(u_i, v_i)$ using the sensor depth $d_i = \mathbf{D}_t(u_i, v_i)$:

$$\mathbf{p}_{c,i} = d_i \cdot \mathbf{K}^{-1} \begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} = \begin{bmatrix} d_i(u_i - c_x)/f_x \\ d_i(v_i - c_y)/f_y \\ d_i \end{bmatrix}$$

Step-by-step: 1. $\mathbf{p}_{c,i}$: 3D point in **camera frame** from depth back-projection 2. $\tilde{\mathbf{p}}_{c,i}$: homogeneous form of $\mathbf{p}_{c,i}$ (append 1) 3. $\tilde{\boldsymbol{\mu}}_i = \mathbf{T}_{wc}^t \tilde{\mathbf{p}}_{c,i}$: transform to **world frame** (4-vector) 4. $\boldsymbol{\mu}_i = \tilde{\boldsymbol{\mu}}_{i,\,1:3}$: extract the first 3 components as the new Gaussian's world-space center

ED complements (rather than replaces) the standard gradient-based clone/split operator by injecting primitives in image-space error regions that may not yet be supported by nearby Gaussians under tight online budgets. All ED hyperparameters $(K_{\text{ED}}, N, \tau, \delta, \alpha_0)$ are fixed across scenes and reported in Sec. IV-A.

### *D. Camera-aware Scale (CS) Initialization*

3DGS initializes each Gaussian's covariance isotropically using a k-NN heuristic (e.g., the mean distance to its three nearest neighbors in an initial point cloud) [9]. This ties the initial scale to local point density, which is reasonable when the point cloud is dense and uniformly sampled. In incremental SLAM, however, new points arrive sparsely and unevenly due to feature tracking and keyframe selection, so point density can reflect texture and pipeline stochasticity rather than scene geometry [21]. Under a short online optimization budget, inaccurate initial scales are harder to correct and can manifest as considerable splats (blurry surfaces) or undersized primitives (holes), as illustrated in Fig. 4.

We replace density-based scale initialization with a camera-aware rule derived from the metric footprint of a pixel at depth $d$. For a pinhole camera with focal lengths $(f_x, f_y)$ (in pixels), the back-projection of pixel $(u,v)$ with depth $d$ is $\mathbf{p}_c = d\,\mathbf{K}^{-1}[u,v,1]^\top$. A one-pixel change in image coordinates corresponds to metric displacements $\Delta_x(d) = d/f_x$ and $\Delta_y(d) = d/f_y$ in camera space. We define an isotropic initial scale proportional to this pixel footprint:

$$s_{\text{cam}}(d) = \lambda \frac{d}{\bar{f}}, \qquad \bar{f} = \frac{1}{2}(f_x + f_y), \tag{10}$$

where $\lambda$ is a global coverage factor. We initialize newly inserted Gaussians with $\boldsymbol{\Sigma}_0 = s_{\text{cam}}(d)^2 \mathbf{I}$ and then optimize

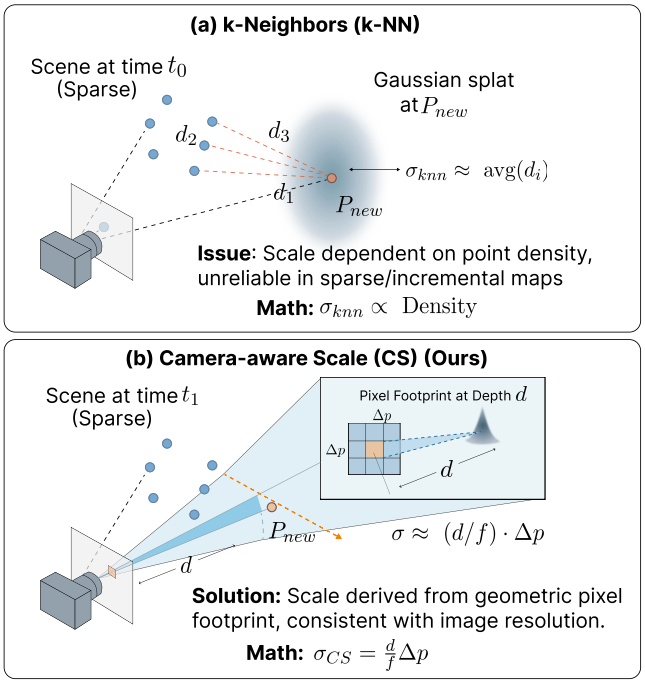



**Fig. 4: Scale initialization in online SLAM.** (a) k-NN scale initialization depends on sparse point density, often producing considerable splats and blurry artifacts. (b) Camera-aware Scale initializes the Gaussian size from the metric pixel footprint at the observed depth, yielding consistent, sharper primitives independent of point density.

anisotropic covariance as in standard 3DGS. In our RGB-D setting, $d$ is directly available from the sensor depth at insertion time (e.g., for ED in Sec. III-C).

Eq. (10) follows the same pixel-footprint intuition used by mip-NeRF-style conical frustums [18] and recent error-guided Gaussian insertion strategies [17], but we apply it in an online RGB-D SLAM backend and use directly observed depth to set the initial scale at insertion time (rather than relying on an auxiliary depth predictor). All hyperparameters, including $\lambda$, are reported in Sec. IV-A.

### *E. Transmittance-Preserving Densification (TPD)*

Adaptive density control (ADC) in 3DGS increases representation capacity by *cloning* Gaussians that exhibit large positional gradients [9]. In the standard implementation, the cloned Gaussian inherits the parent's opacity. Recent analysis shows that this choice introduces a systematic bias under the alpha-compositing used by Gaussian splatting, effectively *inflating opacity* in regions that undergo repeated cloning and reducing the contribution of farther primitives [14].

**Opacity inflation under cloning.** Consider the setting used in prior work [14]: rendering the *center pixel* of a splat, where the primitive induces an alpha coefficient $\alpha_i \in (0, 1)$. Let $T_{\text{pre}}$ denote the accumulated transmittance from all primitives in front of $g_i$ along this pixel ray. Before cloning, the residual transmittance passed to all farther primitives is $T_{\text{pre}}(1 - \alpha_i)$. If we naïvely clone $g_i$ into two overlapping copies with the same opacity, alpha compositing yields residual transmittance $T_{\text{pre}}(1 - \alpha_i)^2$, which is strictly smaller for $\alpha_i \in (0, 1)$ and thus over-weights the cloned region.

**Transmittance-preserving opacity correction.** To prevent systematic opacity inflation [22], when $g_i$ is cloned we overwrite the opacity of *both* the parent and the clone with a corrected value $\hat{\alpha}_i$ such that the composite transmittance is preserved in the worst-case overlap scenario:

$$T_{\text{pre}}(1 - \hat{\alpha}_i)^2 = T_{\text{pre}}(1 - \alpha_i).$$

Solving gives the closed form

$$\hat{\alpha}_i = 1 - \sqrt{1 - \alpha_i}. \tag{11}$$

This correction is exact for the center-pixel overlap analysis, and, as argued in [14], it reduces the cloning-induced bias for general pixels even though it cannot eliminate it completely for all screen-space overlap configurations. We apply Eq. (11) only to *clone* operations; split offspring are displaced and resized, so the perfect-overlap assumption does not hold.

## IV. Evaluations


**(a) Photo-SLAM [10]** **(b) Our Rendering**

**Fig. 5:** Comparison over representative views between (a) the Photo-SLAM baseline [10] and (b) our geometry-aware online mapping; PSNR (dB) is overlaid on the renderings.

### *A. Experimental Setup*

We evaluate our mapping-side modifications along two axes: (i) *photorealistic rendering quality* of the final Gaussian map when rendered from the estimated camera trajectory, and (ii) *camera tracking accuracy*. We compare three configurations: (1) the Photo-SLAM baseline [10], (2) baseline +Transmittance-Preserving Densification + Camera-aware Scale initialization (+TPD+CS), and (3) baseline +TPD+CS + Error-guided Densification (+ED). Importantly, TPD/CS/ED modify the *mapping* thread only; tracking (ORB-SLAM3) is unchanged.

**TABLE I: Rendering quality and tracking efficiency.** Evaluated on the full estimated trajectory (average of 3 runs). Our geometry-aware mapping consistently outperforms the baseline in rendering quality (PSNR, SSIM, LPIPS) across both datasets, while strictly maintaining real-time tracking performance (FPS). Best results are in **bold**.

| | Scene | Baseline (Photo-SLAM) | | | | Ours | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | PSNR↑ | SSIM↑ | LPIPS↓ | FPS↑ | PSNR↑ | SSIM↑ | LPIPS↓ | FPS↑ |
| TUM RGB-D | fr1/desk | 20.87 | .743 | .239 | 58.4 | **21.44** | **.761** | **.216** | **79** |
| | fr2/xyz | 22.09 | .765 | .169 | 52.9 | **23.27** | **.775** | **.145** | **72** |
| | fr3/office | **22.74** | **.780** | .154 | 43.7 | 22.70 | .774 | **.152** | **52** |
| | **Average** | 21.90 | .763 | .187 | 51.6 | **22.47** | **.770** | **.171** | **68** |
| Replica | office0 | **38.48** | .964 | .050 | 48.6 | 38.41 | **.968** | **.046** | **58** |
| | office1 | 39.09 | .961 | .047 | 47.3 | **39.61** | **.965** | **.041** | **62** |
| | office2 | 33.03 | .938 | .077 | 44.1 | **33.07** | **.944** | **.073** | **57** |
| | office3 | **33.79** | .938 | .066 | 40.6 | 33.63 | **.942** | **.063** | **54** |
| | office4 | **36.02** | .952 | .054 | 39.9 | 35.76 | **.955** | **.051** | **50** |
| | room0 | 30.72 | .899 | .075 | 39.8 | **31.76** | **.923** | **.066** | **53** |
| | room1 | 33.51 | .934 | .057 | 43.4 | **33.97** | **.943** | **.052** | **56** |
| | room2 | 35.03 | .951 | .043 | 36.2 | **35.49** | **.959** | **.039** | **49** |
| | **Average** | 34.96 | .942 | .059 | 42.5 | **35.21** | **.950** | **.054** | **55** |

**(a) Ground Truth** **(b) Ours**

**Fig. 6: Additional qualitative results on TUM RGB-D.** Top row: *fr3/office*. Bottom row: *fr1/desk*. Left: Ground Truth rendering. Right: Ours, demonstrating a high visual fidelity that closely matches the ground truth under the same online mapping budget.

We follow Photo-SLAM's RGB-D setting and evaluate on two standard RGB-D benchmarks: TUM RGB-D [24] and Replica [23]. All experiments are conducted on an NVIDIA RTX 5090 (32GB) with CUDA 12.8. We use the official Photo-SLAM evaluation toolkit [25] to compute PSNR, SSIM [26], and LPIPS [27]. Crucially, the toolkit renders the saved Gaussian model along the *entire* estimated trajectory by timestamp-associating poses with the dataset RGB stream, rather than evaluating only on training keyframes [25]. This choice is deliberate: prior work has noted that reporting rendering metrics on the same input (training) views can be misleading because high-capacity models may simply overfit; evaluating on held-out/novel views is more informative [12].

**TABLE II:** Hyperparameters used by our method.

| Symbol | Value | Description |
|---|---|---|
| $\lambda$ | 1.0 | CS coverage factor in Eq. (10) |
| $K_{\text{ED}}$ | 100 | ED update period (mapping iterations) |
| $N$ | $0.5\,\lvert\mathcal{P}_{\text{err}}\rvert$ | ED samples per update (50% of error pixels) |
| $\tau$ | 0.02 | residual threshold for ED candidate pixels |
| $\delta$ | 0.5 | depth-edge threshold (finite differences) |
| $\alpha_0$ | 0.1 | initial opacity for ED-inserted Gaussians |

**TABLE III:** Comparison with prior systems on Replica [23], averaged over 8 scenes. Results marked with † are from prior work, while ‡ indicates our results.

| **Method** | **PSNR↑** | **SSIM↑** | **LPIPS↓** | **FPS↑** |
|---|---|---|---|---|
| BundleFusion† [1] | 23.84 | 0.822 | 0.197 | 8.6 |
| Nice-SLAM† [2] | 26.16 | 0.832 | 0.232 | 2.3 |
| ESLAM† [3] | 30.59 | 0.866 | 0.162 | 6.7 |
| Co-SLAM† [4] | 30.25 | 0.864 | 0.175 | 14.6 |
| SplaTAM† [12] | 34.11 | 0.970 | 0.100 | ∼0.5 |
| GS-SLAM† [13] | 34.27 | **0.975** | 0.082 | ∼8.0 |
| Photo-SLAM† [10] | 34.96 | 0.942 | 0.059 | 42.5 |
| **Ours‡** | **35.21** | 0.950 | **0.054** | **55.0** |

While standard SLAM benchmarks do not always provide fully independent hold-out views, rendering on the full trajectory (including non-keyframes) provides a substantially more demanding and honest proxy than keyframe-only evaluation.

SLAM systems are inherently non-deterministic due to multi-thread scheduling and feature-level stochasticity. We therefore run each configuration three times per scene and report mean±std to ensure fairness. This practice is

**TABLE IV:** Tracking accuracy (ATE RMSE in cm, mean $\pm$ std over 3 runs). TPD and CS are mapping-only; differences from Baseline lie within ORB-SLAM3 run-to-run variance.

| Dataset | Scene | Baseline | Ours |
|---|---|---|---|
| TUM | fr1/desk | $1.53 \pm 0.01$ | $1.53 \pm 0.01$ |
| | fr2/xyz | $0.31 \pm 0.02$ | $0.30 \pm 0.01$ |
| | fr3/office | $0.90 \pm 0.02$ | $0.94 \pm 0.02$ |
| | *Average* | $\mathbf{0.91}$ | $0.92$ |
| Replica | office0 | $0.45 \pm 0.02$ | $0.45 \pm 0.02$ |
| | office1 | $0.44 \pm 0.09$ | $0.37 \pm 0.02$ |
| | office2 | $1.04 \pm 0.17$ | $0.98 \pm 0.10$ |
| | office3 | $0.38 \pm 0.01$ | $0.38 \pm 0.02$ |
| | office4 | $0.50 \pm 0.03$ | $0.53 \pm 0.08$ |
| | room0 | $0.31 \pm 0.03$ | $0.32 \pm 0.01$ |
| | room1 | $0.39 \pm 0.01$ | $0.35 \pm 0.01$ |
| | room2 | $0.20 \pm 0.00$ | $0.21 \pm 0.01$ |
| | *Average* | $0.46$ | $\mathbf{0.45}$ |
| *Overall Average* | | $0.59$ | $\mathbf{0.58}$ |

**TABLE V:** Rendering-quality ablation within our method (averages only), mean of 3 runs. Rendering metrics via Photo-SLAM-eval.

| Dataset | +TPD+CS | | | +TPD+CS+ED | | |
|---|---|---|---|---|---|---|
| | PSNR↑ | SSIM↑ | LPIPS↓ | PSNR↑ | SSIM↑ | LPIPS↓ |
| TUM RGB-D | 21.91 | .771 | .182 | 21.85 | .762 | .184 |
| Replica | 34.72 | .944 | .055 | 34.39 | .931 | .055 |
| **Overall Avg** | 31.22 | .891 | .082 | 30.97 | .874 | .083 |

also consistent with Photo-SLAM's own recommendation to repeat runs to reduce the impact of nondeterminism [10]. Unless otherwise specified, we follow the official Photo-SLAM implementation and keep all baseline tracking and mapping parameters unchanged. For mapping, we use the provided Gaussian-mapper YAML configurations for each dataset, which fix the optimizer and the standard 3DGS density-control schedule (e.g., a densification interval of 100 iterations and a gradient threshold of $0.001$; dataset-specific pruning/densification ranges are kept as in the configs). Our mapping-side modifications introduce only a small set of additional hyperparameters summarized in Table II, and we keep them fixed across all scenes.

OC is parameter-free: when cloning is triggered by the baseline density controller, we apply the closed-form opacity correction in Eq. (11). Camera-aware Scale initialization uses a single coverage factor $\lambda$ in Eq. (10). Error-guided Densification (ED) is executed every $K_{\mathrm{ED}}$ mapping iterations; at each ED step as shown in Table V, we sample $N$ pixels from the residual map using threshold $\tau$, filter depth edges using $\delta$, and initialize spawned Gaussians with opacity $\alpha_0$ (Sec. III-C).

### *B. Rendering Quality*

Table I reports full-trajectory rendering quality over all 11 scenes. Across both the real-world TUM RGB-D and synthetic Replica benchmarks, our geometry-aware mapping (TPD + CS) achieves the best average PSNR, improving from 21.90 to 22.47 dB on TUM and from 34.96 to 35.21 dB on Replica, our geometry-aware online mapping significantly improves rendering quality over the Photo-SLAM [10] baseline, achieving higher PSNR scores across representative views and effectively eliminating artifacts in complex regions (red circles) as shown in Fig. 5. Although PSNR gains are not uniform across all sequences, the overall trend is supported by corresponding improvements in average SSIM and reductions in LPIPS. The largest PSNR gains are observed on more challenging scenes such as fr2/xyz (+1.18 dB) and room0 (+1.04 dB), indicating that the proposed design is particularly beneficial under difficult online mapping conditions.

Furthermore, as demonstrated in Fig. 6, our method is capable of recovering highly intricate details across diverse scenes such as fr3/office and fr1/desk. Notably, in well-observed regions, specific rendered frames can achieve a visual fidelity that closely approximates the ground truth, even under a strict online mapping budget.

**TABLE VI:** Iteration ablation on TUM RGB-D (average of fr1/desk and fr2/xyz). Performance of the baseline versus our proposed mapping modifications under varying per-keyframe iteration budgets.

| Iters / KF | PSNR ↑ | | SSIM ↑ | | LPIPS ↓ | | #Gaussians (K) | |
|---|---|---|---|---|---|---|---|---|
| | Base | Ours | Base | Ours | Base | Ours | Base | Ours |
| 50 | 13.70 | **13.80** | 0.575 | **0.578** | 0.598 | **0.588** | 3.50 | 3.50 |
| 100 | 12.90 | **14.25** | 0.575 | **0.582** | 0.588 | **0.580** | 1.90 | 1.90 |
| 200 | **15.90** | 15.70 | 0.613 | **0.618** | 0.498 | **0.490** | 2.30 | 2.40 |
| 300 | **16.90** | 16.80 | **0.638** | 0.635 | **0.425** | 0.428 | 2.85 | 2.83 |

### *C. Efficiency Analysis*

As shown in Table I, TPD/CS/ED introduces no measurable overhead in tracking throughput (FPS). Furthermore, Table IV confirms that our geometry-aware enhancements do not compromise tracking accuracy. Because our modifications operate strictly within the asynchronous mapping backend, the minor fluctuations in ATE RMSE between the baseline and our method are purely attributable to the inherent run-to-run stochasticity of the ORB-SLAM3 frontend.

Per-keyframe rendering time during mapping remains sub-millisecond across configurations. TPD+CS moderately increases the final number of Gaussians compared to the baseline. Intuitively, changing the initial scale distribution affects subsequent pruning/densification dynamics in adaptive density control. ED adds additional Gaussians in proportion to the number of spawning events and samples, as expected, but does not reduce the tracking FPS.

**Impact of Optimization Budget.** As shown in Table VI, both the baseline and our method (+TPD+CS) converge to comparable rendering quality given a large budget (200–300 iterations per keyframe). However, under computational constraints of online SLAM (e.g., 100 iterations), the baseline suffers a severe performance drop. In contrast, our geometry-aware approach maintains stable map growth and

superior photometric accuracy, demonstrating significantly faster convergence and robustness in resource-constrained settings.

### D. Context with Other 3DGS-SLAM Methods

Table III provides context against published Gaussian-SLAM systems. Direct numerical comparison must be treated with caution because evaluation protocols differ substantially across papers—in particular, many works report rendering on training views, whereas our primary results follow a full-trajectory protocol via the Photo-SLAM evaluation toolkit [25], motivated by concerns about overfitting and the limited informativeness of train-view rendering [12]. Finally, our modifications preserve Photo-SLAM's core advantage: a decoupled pipeline that maintains real-time tracking while improving mapping-side rendering quality [10].

## V. Conclusion

We studied online 3DGS mapping in a SLAM setting and found that several widely used *offline* heuristics become brittle when optimization is constrained to a short per-keyframe budget. Building on the decoupled Photo-SLAM architecture [10], we proposed three lightweight, training-free mapping-side modifications: Transmittance-Preserving Densification to prevent opacity drift under cloning, Camera-aware Scale initialization to tie the initial Gaussian size to camera geometry rather than point density, and Error-guided Densification to explicitly allocate new primitives to regions with persistently high residual values. Finally, our results reinforce a methodological point: because SLAM pipelines are non-deterministic, reporting multi-run variance and using more demanding rendering protocols (e.g., full-trajectory evaluation rather than keyframe-only train-view rendering) is important for honest benchmarking in the rapidly growing 3DGS-SLAM literature. Future work could incorporate uncertainty-aware densification and explore principled evaluation protocols that combine novel-view rendering with trajectory reliability.

## References


[1] A. Dai, M. Nießner, M. Zollhöfer, S. Izadi, and C. Theobalt, "Bundlefusion: Real-time globally consistent 3d reconstruction using on-the-fly surface reintegration," *ACM Transactions on Graphics (ToG)*, vol. 36, no. 4, p. 1, 2017.

[2] Z. Zhu, S. Peng, V. Larsson, W. Xu, H. Bao, Z. Cui, M. R. Oswald, and M. Pollefeys, "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.

[3] M. M. Johari, C. Carta, and F. Fleuret, "Eslam: Efficient dense slam system based on hybrid representation of signed distance fields," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2023, pp. 17 408–17 419.

[4] H. Wang, J. Wang, and L. Agapito, "Co-slam: Joint coordinate and sparse parametric encodings for neural real-time slam," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2023, pp. 13 293–13 302.

[5] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. J. Leonard, "Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age," *IEEE Transactions on robotics*, vol. 32, no. 6, pp. 1309–1332, 2017.

[6] T. Taketomi, H. Uchiyama, and S. Ikeda, "Visual slam algorithms: A survey from 2010 to 2016," *IPSJ transactions on computer vision and applications*, vol. 9, no. 1, p. 16, 2017.

[7] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, "Orb-slam: A versatile and accurate monocular slam system," *IEEE transactions on robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.

[8] C. Campos, R. Elvira, J. J. G. Rodríguez, J. M. Montiel, and J. D. Tardós, "Orb-slam3: An accurate open-source library for visual, visual–inertial, and multimap slam," *IEEE transactions on robotics*, vol. 37, no. 6, pp. 1874–1890, 2021.

[9] B. Kerbl, G. Kopanas, T. Leimkühler, G. Drettakis, *et al.*, "3d gaussian splatting for real-time radiance field rendering." *ACM Trans. Graph.*, vol. 42, no. 4, pp. 139–1, 2023.

[10] H. Huang, L. Li, H. Cheng, and S.-K. Yeung, "Photo-slam: Real-time simultaneous localization and photorealistic mapping for monocular stereo and rgb-d cameras," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 21 584–21 593.

[11] 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.

[12] 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.

[13] C. Yan, D. Qu, D. Xu, B. Zhao, Z. Wang, D. Wang, and X. Li, "Gs-slam: Dense visual slam with 3d gaussian splatting," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2024, pp. 19 595–19 604.

[14] S. Rota Bulò, L. Porzi, and P. Kontschieder, "Revising densification in gaussian splatting," in *European Conference on Computer Vision*. Springer, 2024, pp. 347–362.

[15] 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.

[16] 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.

[17] B. Baranowski, S. Esposito, P. Gschoßmann, A. Chen, and A. Geiger, "Conegs: Error-guided densification using pixel cones for improved reconstruction with fewer primitives," *arXiv preprint arXiv:2511.06810*, 2025.

[18] J. T. Barron, B. Mildenhall, M. Tancik, P. Hedman, R. Martin-Brualla, and P. P. Srinivasan, "Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 5855–5864.

[19] S. Kheradmand, D. Rebain, G. Sharma, W. Sun, Y.-C. Tseng, H. Isack, A. Kar, A. Tagliasacchi, and K. M. Yi, "3d gaussian splatting as markov chain monte carlo," *Advances in Neural Information Processing Systems*, vol. 37, pp. 80 965–80 986, 2024.

[20] 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.

[21] Q. Tran and T. Dang, "Triags: Differentiable triangulation-guided geometric consistency for 3d gaussian splatting," *arXiv preprint arXiv:2512.06269*, 2025.

[22] S. Rota Bulò, L. Porzi, and P. Kontschieder, "Revising densification in gaussian splatting," in *European Conference on Computer Vision*. Springer, 2024, pp. 347–362.

[23] J. Straub, T. Whelan, L. Ma, Y. Chen, E. Wijmans, S. Green, J. J. Engel, R. Mur-Artal, C. Ren, S. Verma, *et al.*, "The replica dataset: A digital replica of indoor spaces," *arXiv preprint arXiv:1906.05797*, 2019.

[24] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of rgb-d slam systems," in *2012 IEEE/RSJ international conference on intelligent robots and systems*. IEEE, 2012, pp. 573–580.

[25] H. Huang *et al.*, "Photo-slam evaluation toolkit," https://github.com/HuajianUP/Photo-SLAM-eval, 2024, official evaluation toolkit for Photo-SLAM (CVPR 2024).

[26] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE transactions on image processing*, vol. 13, no. 4, pp. 600–612, 2004.

[27] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang, "The unreasonable effectiveness of deep features as a perceptual metric," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 586–595.