IG-SLAM: Instant Gaussian SLAM

F. Aykut Sarıkamış   A. Aydın Alatan

Center for Image Analysis (OGAM), EEE Department, METU, Turkey

Figure 1. **Qualitative rendering results from Photo-SLAM [11] and IG-SLAM.** We compare the visual quality of the methods on the large-scale EuRoC dataset [3].

Abstract

3D Gaussian Splatting has recently shown promising results as an alternative scene representation in SLAM systems to neural implicit representations. However, current methods either lack dense depth maps to supervise the mapping process or detailed training designs that consider the scale of the environment. To address these drawbacks, we present IG-SLAM, a dense RGB-only SLAM system that employs robust dense SLAM methods for tracking and combines them with Gaussian Splatting. A 3D map of the environment is constructed using accurate pose and dense depth provided by tracking. Additionally, we utilize depth uncertainty in map optimization to improve 3D reconstruction. Our decay strategy in map optimization enhances convergence and allows the system to run at 10 fps in a single process. We demonstrate competitive performance with state-of-the-art RGB-only SLAM systems while achieving faster operation speeds. We present our experiments on the Replica, TUM-RGBD, ScanNet, and EuRoC datasets. The system achieves photo-realistic 3D reconstruction in large-scale sequences, particularly in the EuRoC dataset.

1. Introduction

Dense Simultaneous Localization and Mapping (SLAM) is a fundamental problem in computer vision with numerous applications in robotics, augmented reality, virtual reality, and more. Any SLAM system must operate in real-time and scale to large scenes for all these real-world applications. Additionally, the system must be robust against noisy visual sensor measurements.

The prominent scene representation is a 3D point cloud in traditional Dense SLAM systems. However, point clouds are an impoverished representation of the world. As a sparse representation, the point clouds do not provide water-tight, photo-realistic depictions of the environment. Recently, two promising scene representations have been introduced and studied in the SLAM literature: Neural Radiance Fields (NeRF) [17] and Gaussian Splatting [14].

Earlier dense SLAM studies that equip NeRF as an only-scene representation [31, 49] achieved 3D reconstruction without camera poses in real-time. Several following studies [5, 26, 47] integrate classical SLAM methods such as tracking by feature matching, dense-bundle adjustment, loop closure, and global bundle adjustment. Several performance improvements are made in later studies [12, 36, 45, 48, 50] by incorporating additional data structures along with NeRF [17], by employing off-the-shelf tracking modules [19, 34] and monocular depth estimation [7]. However, NeRF suffers from slow rendering speed [24]; since the real-time operation is crucial for a SLAM system, slow rendering speed puts NeRF into a disadvantageous position as a scene representation.

Later the following studies incorporate Gaussian Splatting as scene representation: Early works [16, 39, 43] adopt Gaussian Splatting as an only-scene representation and simultaneously track and map the environment in real-time. However, utilizing novel view synthesis methods as both

tracking and mapping tools is compelling yet challenging. The difficulty arises because pose and map optimizations are performed jointly. To decouple these two daunting tasks, [11, 27] utilize traditional SLAM methods demonstrating superior performance over only-scene representation methods in terms of reconstruction. However, these studies either lack dense depth supervision or a high frame rate.

Purposely, we introduce IG-SLAM, a deep-learning-based dense SLAM system that achieves photo-realistic 3D reconstruction in real-time. The proposed system features robust pose estimation, refined dense depth maps, and Gaussian Splatting representation. The proposed system frequently performs global dense bundle adjustment to reduce drift. Since the pose and depth maps optimized by a dense SLAM system are often noisy, we utilize depth uncertainty to make the mapping process robust to noise. Our efficient mapping algorithm is optimized specifically to work with dense depth maps enabling our system to operate at high frame rates. We perform extensive experiments on various indoor RGB sequences, demonstrating the robustness, fast operation speed, and scalability of our method. In summary, we make the following contributions:

- We present IG-SLAM, an efficient dense RGB SLAM system that performs at high frame rates, offering scalability and robustness even in challenging conditions.
- A novel 3D reconstruction algorithm that accounts for depth uncertainty, making the 3D reconstruction robust to noise.
- A training procedure to make dense depth supervision for the mapping process as efficient as possible.

2. Related Work

2.1. Dense Visual SLAM

Pioneering dense SLAM algorithms, DTAM [21] and KinectFusion [20], show that dense SLAM can be performed in real-time despite its computational complexity. DTAM aims to produce dense depth maps associated with the keyframes, known as the view-centric approach. Later research adopted a similar approach but with a crucial distinction. While these traditional approaches generally decouple the optimization of dense maps and poses, some recent works focus on joint optimization. However, optimization of the full-resolution depth map is not feasible due to the high number of independent variables. Therefore, the following research focuses on reducing the computational complexity of joint optimization. For this purpose, BA-Net [32] includes a depth map into the bundle adjustment layer utilizing a basis of depth maps and optimizing the linear combination coefficients. Code-SLAM [2] reduces the dimension of dense maps by an autoencoder-inspired architecture. DROID-SLAM [34] optimizes down-sampled

dense maps in a bundle-adjustment layer with a reprojection error, aided by optical flow revisions [33]. A recent work, FlowMap [28], estimates a dense depth map with a convolutional neural network and calculates the pose analytically using the optical flow. As world-centric alternatives to this approach, Neural Radiance Fields [17] and Gaussian Splatting [14] are utilized in the literature.

2.2. Neural Radiance Field Scene Representation

NeRF [17] encodes the scene as radiance fields utilizing a simple multi-layer perceptron (MLP). The original NeRF formulation exhibits slow training and rendering speeds. However, several improvements have been proposed on this initial formulation. The cone-shaped rendering [1] is utilized to address anti-aliasing, additional data structures are also employed, such as voxel grid [8, 15, 25], plenoctree [9, 37, 42], hash tables [18] and many more achieve orders of magnitude faster rendering and training compared to the original NeRF [17]. Surface-based methods [22, 38, 40] also unify surface and volume rendering.

The landmark work iNeRF [41] calculates camera poses given a NeRF representation by fixing the NeRF representation and minimizing rendering error by optimizing the camera pose around an initial guess. iMAP [31], as the first representation-only work, optimizes the pose by fixing the NeRF representation and optimizes the map based on the calculated pose. NICE-SLAM [49] introduces a hierarchical coarse-to-fine mapping approach. To decouple map and pose optimization, Orbeez-SLAM [5] leverages robust visual SLAM methods [19] and multi-resolution hash encoding [18]. NeRF-SLAM [26] introduces dense depth maps with covariance and poses generated by the robust dense-SLAM algorithm DROID-SLAM [34]. GO-SLAM employs loop closing and global dense bundle adjustment to achieve globally consistent reconstruction. NICER-SLAM [50] extends NICE-SLAM [49] incorporating off-the-shelf monocular depth and normal estimators. Recently, MoD-SLAM [48] utilizes cone-shaped projection in rendering [1]. GlORIE-SLAM [45] utilizes monocular depth estimation for mapping supervision.

2.3. 3D Gaussian Splatting Scene Representation

3D Gaussian Splatting represents the scene as a set of Gaussians of varying colors, shapes, and opacity. Several improvements are proposed for consistency and reconstruction quality. For example, 2D counterpart [10] is also proposed to enhance multi-view consistency. Moreover, the rendering depth with alpha-blending as in the original 3D Gaussian Splatting causes noisy surfaces; hence, more rigorous methods address this issue by utilizing varying depths per Gaussian according to the viewpoint [4, 44].

Due to its fast rendering speed and being an explicit scene representation as opposed to NeRF [17],
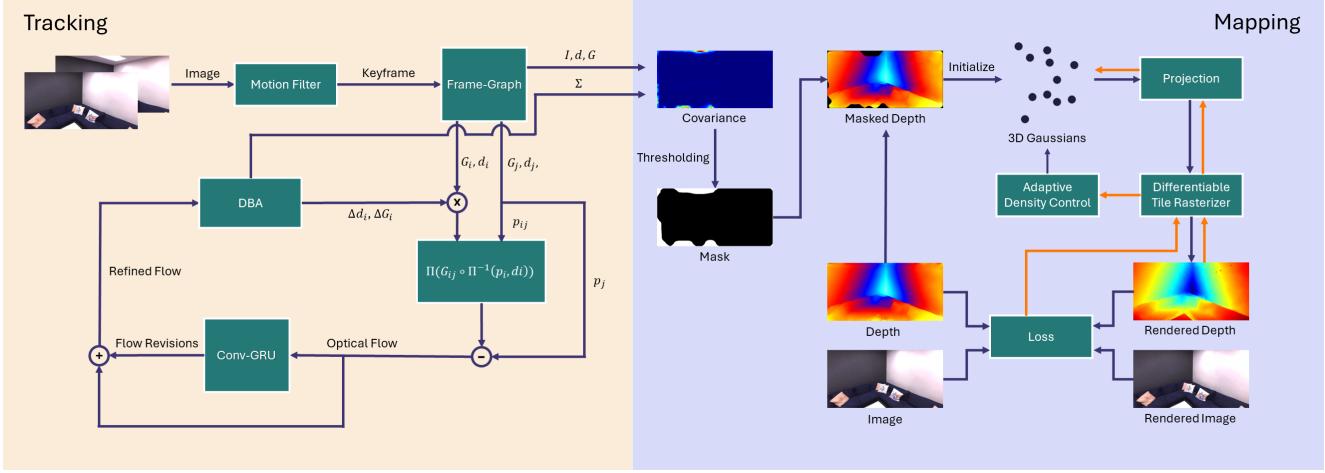
Figure 2. **System Overview.** Our system takes an RGB image stream as input and outputs the camera pose and scene representation in the form of a set of Gaussians. We decouple this objective into two parts: tracking and mapping. **Tracking:** Keyframes are created and added to the frame graph based on average optical flow. Pretrained GRU refines optical flow between keyframes. Dense bundle adjustment (DBA) is performed on the frame graph, minimizing reprojection error while optimizing the dense depth map and camera pose, and calculating depth map covariance simultaneously. After several iterations, depth maps and camera poses are expected to converge. **Mapping:** Keyframes' pose, depth, and covariance obtained from tracking are used for 3D reconstruction. We initialize Gaussians from low covariance regions utilizing the camera pose and depth map. 3D Gaussians are then projected onto the image plane and rendered utilizing a differentiable tile rasterizer. The loss function is a combination of depth and color loss. The depth loss is weighted by covariance. Finally, the loss is backpropagated to optimize Gaussians orientation, scaling, opacity, position, and color designated by orange arrows in the figure. Moreover, Gaussians are split, cloned, and pruned based on the local gradients.

Gaussian Splatting [14] has also quickly gained attention in the SLAM literature. MonoGS [16], GS-SLAM [39], and SplaTAM [13] are pioneering Gaussian-Splatting representation-only SLAM algorithms that jointly optimize Gaussians and the pose. Gaussian-SLAM [43] introduces sub-maps to mitigate neural forgetting. Photo-SLAM [11] decouples tracking and mapping by employing a traditional visual SLAM algorithm [19] as its tracking module and introduces a coarse-to-fine map optimization approach. RTG-SLAM [23] renders depth by considering only the foremost opaque Gaussians. Recent work, Splat-SLAM [27] uses proxy depth maps to supervise map optimization.

3. Proposed Method

We provide an overview of the proposed method in Fig. 2. Our tracking algorithm (Sec. 3.1) generates a dense depth map, depth uncertainty, and the camera pose for each keyframe. These outputs are then used to supervise our mapping algorithm (Sec. 3.2). The Gaussians are initialized based on the camera pose and dense depth and are optimized using color and weighted depth loss. Real-time operation is achieved through a sliding window of keyframes.

3.1. Tracking

We mainly employ DROID-SLAM [34] as our tracking module. DROID-SLAM maintains two state variables:

camera pose $\mathbf{G}_t$ and inverse depth $\mathbf{d}_t$ for each camera frame t. DROID-SLAM constructs a frame graph $(\mathcal{V}, \mathcal{E})$ of keyframes based on co-visibility. Keyframes are selected from all camera frames when the average magnitude of the optical flow for a frame is higher than a certain threshold. If there is a visual overlap between frames i and frame j, an edge is created between the ith and j^{th} vertex in $\mathcal{V}$. This graph is updated during inference. Given the initial pose and depth estimates $(\mathbf{G}_i, \mathbf{d}_i)$ and $(\mathbf{G}_j, \mathbf{d}_j)$ for frame i and j, the optical flow field is estimated by unprojecting the pixels from frame i, projecting them into frame j, and taking the pixel-wise position difference. In other words, the reprojected pixel locations p_{ij} is calculated as in Eq. (1)

$$p_{ij} = \Pi(\mathbf{G}_{ij} \circ \Pi^{-1}(\mathbf{p}_i, \mathbf{d}_i)), \quad \mathbf{p}_{ij} \in \mathbb{R}^{H \times W \times 2} \quad (1)$$

where $\mathbf{G}_{ij} = \mathbf{G}_j^{-1} \circ \mathbf{G}_i$. Then, the optical flow is initially calculated as $p_{ij} - p_j$. This estimate is fed into GRU along with a correlation vector which is an inner product between features of the frames. The GRU produces flow revisions $\mathbf{r}_{ij}$ and confidence weights $\mathbf{w}_{ij}$. the refined reprojected pixel locations $\mathbf{p}_{ij}^*$ are computed similarly to Eq. (1) incorporating the flow correction from the GRU. Then, the dense bundle adjustment layer minimizes the cost function in Eq. (2).

$$\mathbf{E}(\mathbf{G}', \mathbf{d}') = \sum_{i,j \in \mathcal{E}} \left\| \mathbf{p}^*_{ij} - \mathbf{p'}_{ij} \right\|^2_{\Sigma_{ij}}$$
$$\mathbf{p'}_{ij} = \Pi(\mathbf{G'}_{ij} \circ \Pi^{-1}(\mathbf{p}_i, \mathbf{d'}_i)) \qquad (2)$$

where $\Sigma_{ij} = \text{diag}(\mathbf{w}_{ij})$ and $\|.\|_\Sigma$ Mahalanobis norm weighted according to the weights $\mathbf{w}_{ij}$. Linearizing Eq. (2) around $(\mathbf{G}', \mathbf{d}')$ and solve for pose and depth updates $(\Delta \xi, \Delta \mathbf{d})$ using Gauss-Newton algorithm. The linearized system of equations becomes

$$H\mathbf{x} = \mathbf{b}, \quad H = \begin{bmatrix} C & E \\ E^T & P \end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} \Delta \xi \\ \Delta \mathbf{d} \end{bmatrix}, \mathbf{b} = \begin{bmatrix} \mathbf{v} \\ \mathbf{w} \end{bmatrix} \quad (3)$$

where H is the Hessian matrix, $\mathbf{x} = [\Delta \xi, \Delta \mathbf{d}]$ is the pose and depth updates, $\mathbf{b} = [\mathbf{v}, \mathbf{w}]$ is the pose and depth residuals, C is the block camera matrix. E is the camera/depth off-diagonal block matrices, and P is the diagonal matrix corresponding to disparities per pixel per keyframe. The bundle adjustment layer operates on the initial flow estimates and updates the keyframes' pose and depth map. Optical flow is then recalculated by refined poses and depth maps which are subsequently fed back into the dense bundle adjustment layer. After successive iterative refinements on the keyframe graph, the poses and depth maps are expected to converge.

After the dense bundle adjustment step, we compute the covariance for depth estimates. As shown in NeRF-SLAM [26], the same Hessian structure in Eq. (3) can be used to calculate covariance for depth estimates Σ_d and poses Σ_G as shown in Eq. (4). The depth covariance is used both as a mask for initializing Gaussians and as weights in the depth component of the loss function.

$$\Sigma_d = P^{-1} + P^{-T} E^T \Sigma_G E P^{-1}$$
$$\Sigma_G = (LL^T)^{-1} \qquad (4)$$

Keyframing We utilize all the keyframes that are actively optimized in the tracking process without any filtering. Each keyframe that participates in mapping contains its camera image I, depth map $\mathbf{d}$, depth covariance Σ_d, and pose $\mathbf{G}$. The mapping process accepts a keyframe only if it is not already in the sliding window. Note that, we do not send all the keyframes created in a mapping cycle, but only the most recent one. Therefore, this approach may result in some keyframes being missed during optimization. However, this design choice prevents abrupt changes in the sliding window caused by a sharp camera movement.

Global BA After the number of total keyframes exceeds the sliding window length for the Dense Bundle Adjustment,

we regularly perform Global Bundle Adjustment for all existing keyframes on a separate graph as described in GO-SLAM [47]. The graph is constructed utilizing a distance metric, where the distance between frame pairs is the average optical flow magnitude. Graph edges are established between consecutive keyframes and those that are close according to the distance metric. Dense bundle adjustment is then applied based on this graph every 10 keyframes. The pose and depth maps are updated at the start of every mapping cycle, along with their covariances. We perform one last global BA at the end of tracking.

3.2. Mapping

The mapping process is responsible for 3D reconstruction with keyframes equipped with pose, image, depth, and covariance acquired from the tracking process.

Representation We adopt Gaussian Splatting [14] as scene representation. A Gaussian function is described by Eq. (5)

$$G(\mathbf{x}) = \exp\left(\frac{1}{2} (\mathbf{x} - \mu)^T \Sigma^{-1} (\mathbf{x} - \mu) \right) \qquad (5)$$

where μ and Σ are the mean and covariance which define the position and shape of this Gaussian. To ensure that the covariance remains semi-definite during optimization, covariance Σ is decomposed into $RSS^T R^T$ where R is the rotation matrix and S is the scaling matrix. In addition to position, rotation, and scaling, opacity α and color c are also optimized. Although the original implementation parameterizes color as spherical harmonic coefficients, our algorithm optimizes the color directly. The projection of a 3D covariance is formulated as $\Sigma' = JR\Sigma R^T J^T$ where R is the rotation component of the world-to-camera transformation T_{cw} and J is the Jacobian of the affine approximation of the projective transformation P [51]. The position is projected directly as $\mu' = PT_{cw}\mu$.

Rendering A set of Gaussians $\mathcal{N}$ visible from a viewpoint, is first projected onto the image plane. 2D Gaussians are then sorted according to their depths and are rasterized via α-blending as described in Eq. (6) for color and depth.

$$\hat{C} = \sum_{i \in \mathcal{N}} c_i \alpha_i \prod_{j=1}^{i-1} (1 - \alpha_j), \quad \hat{D} = \sum_{i \in \mathcal{N}} d_i \alpha_i \prod_{j=1}^{i-1} (1 - \alpha_j) \quad (6)$$

Hierarchical Optimization Since dense depth maps for keyframes are available, we adopt a training strategy similar to RGB-D MonoGS [16] but utilizing a coarse-to-fine training strategy inspired by Photo-SLAM [11] and Instant-NGP [18].

For each keyframe, an image pyramid is constructed by downsampling image, depth, and covariance by a factor of s using bilinear interpolation, as in Eq. (7)

$$\text{KF}_i^l = \{I_i^l, \mathbf{d}_i^l, \Sigma_{di}^l\}$$
$$I_i^l = I_i^0 \downarrow s^l, \quad \mathbf{d}_i^l = \mathbf{d}_i^0 \downarrow s^l, \quad \Sigma_{di}^l = \Sigma_{di}^0 \downarrow s^l \quad (7)$$

where $\downarrow$ denotes the downsampling operation with linear interpolation and l is the pyramid level and $I_i^0, \mathbf{d}_i^0, \Sigma_{di}^0$ are the full resolution image, depth, and covariance respectively. In Photo-SLAM [11], the authors utilize a sharp downsampling factor s of 0.5 and a 2-level pyramid. In contrast, we employ a smoother downsampling factor $s = 0.8$ similar to Instant-NGP [18] and a 3-level pyramid.

In each pyramid level, Gaussians are initialized by unprojection as follows: The points are sampled randomly from the most recent keyframe by using a downsampling factor θ. The sampled points are then unprojected according to depth maps. To account for the noise in depth maps, regions with high covariance are masked out to make the Gaussian initialization more robust to noise. Eq. (8) describes a mask for a given normalized depth covariance.

$$M = \{(i, j) \mid \sigma_{ij} < 0.2\} \quad (8)$$

where M represents the binary mask matrix and i and j represent pixel location. The mask is created by normalizing the covariance Σ between 0 and 1 and identifying the pixel values below 0.2 normalized covariance σ. The mask is then smoothed using thresholding operation as described in Eq. (8) with a maximum filter followed by a majority filter. An example of a mask for a given covariance is shown in figure Fig. 3.
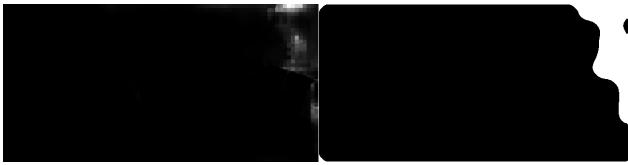

Figure 3. **An example of normalized covariance(left) and corresponding mask(right).** The mask is created by thresholding normalized covariance with a maximum filter and smoothing with a majority filter. The white region on the mask is left out and not used during Gaussian initialization.

The map optimization is performed on a sliding window in a coarse-to-fine fashion. We maintain the last N keyframes within the sliding window to meet the real-time requirements. As the number of iterations increases, we switch to training with higher resolutions in the image pyramid. At the beginning of optimization at each level l, Gaussians are unprotected according to its depth map $\mathbf{d}_i^l$. We render the Gaussians from keyframes' viewpoints in the sliding window, and the loss function is calculated based on the rendered image and depth. Camera images and dense depth maps are utilized as ground truth in mapping supervision.

We employ a loss function that combines weighted depth loss L_{depth} and color loss L_{color} which are defined as below

$$L_{\text{depth}} = \left\| D - \hat{D} \right\|_{\Sigma_d^{-1}}^1, \quad L_{\text{color}} = \left\| C - \hat{C} \right\|^1 \quad (9)$$

where D and C are the ground truth depth and image, respectively, and $\hat{D}$ and $\hat{C}$ are the rendered depth and image according to Eq. (6). The depth loss L_{depth} is weighted by the inverse covariance to ensure that the pixels with high uncertainty are weighted less. The combined loss is given by $L = \alpha L_{\text{color}} + (1 - \alpha)L_{\text{depth}}$. We set $\alpha = 0.5$ throughout all of our experiments. The loss is then backpropagated through a differentiable rendering pipeline where the position, opacity, covariance, maps, and color of the Gaussians are optimized.

Post Processing We refine the mapping results by optimizing the map for several iterations following the conventions established in MonoGS [16], GlORIE-SLAM [45] and Splat-SLAM [27]. For this purpose, we randomly select single frames and optimize the map with the same loss function used in the mapping. We perform the same number of iterations in MonoGS [16] and Splat-SLAM [27] for fairness.

3.3. Training Strategy

A subtle yet crucial point regarding our training strategy is that dense depth maps may be noisy; however, they are unlikely to disrupt depth order. In other words, having a position learning rate such that Gaussians switch positions during training is redundant and hinders optimization convergence. This effect is illustrated in Fig. 4. It should be noted that this is never the case for standard Gaussian Splatting training where the method typically starts with a sparse SfM point cloud. However, since Gaussians are initialized from a dense depth map, they are quite close to each other.

As illustrated in Fig. 4, case **A)** high learning rates cause the optimization to bounce Gaussians around the desired position. Conversely, the polar opposite in **C)** also hinders the convergence. Since setting a perfect learning rate for each iteration is neither feasible nor practical, we choose a learning rate that decays during training according to Eq. (10) to reduce this TV static noise during training. We initialize the learning rate to cover the full range needed to detail the model from coarse to fine while allowing for gradual decay.

$$\text{lr}(t) = \exp((1 - t)\ln(\text{lr}_i) + t\ln(\text{lr}_f)) \quad (10)$$

where $t = n/\tau$ is the iteration number n over decay constant τ, and lr_i, lr_f are the initial and final learning rate, respectively. The impact of learning rate and its decay in training performance are examined in Sec. 4.
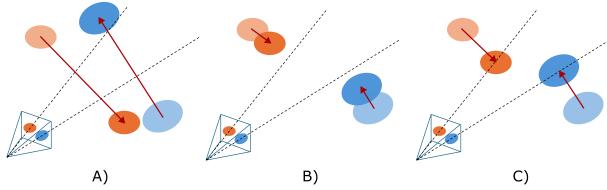
Figure 4. **Three hypothetical cases to encounter in training.** Dashed lines pass through ground truth Gaussian positions from the camera center. The faded Gaussians represent their previous positions. Red lines are the position update steps along the gradient direction. In **A)**, a large position update causes the order of Gaussians to change, creating TV-static-like noise in training. In **B)**, multiple iterations are needed to move Gaussians to the correct place because of small position updates. **C)** represents the ideal case where position update is exactly the position error.

We densify Gaussians in high loss gradient regions at every 150 iterations. Densification is achieved by cloning small Gaussians and by splitting large ones. The occluded Gaussians are also pruned at the end of each sliding window optimization to ensure that only the necessary Gaussians for accurate reconstruction are retained.

4. Experiments

We evaluate our system on various synthetic and real-world datasets. The ablation studies and hyperparameter analyses are also demonstrated to justify our design choices.

4.1. Experimental Setup

Datasets We evaluate the system in Replica [29], TUM RGB-D [30], ScanNet [6], and EuRoC MAV [3] datasets. Replica is a dataset of synthetic indoor scenes. The TUM RGB-D dataset consists of sequences that are recorded in small indoor office environments. The ScanNet dataset consists of 6 sequences of real-world indoor environments. The EuRoC is a dataset collected on board a Micro Aerial Vehicle (MAV) containing stereo images of relatively large-scale indoor environments. All datasets are evaluated without clipping except EuRoC. We clip from the start of the sequences to skip typical pauses at the beginning. We run all sequences 3 times and report the average results to mitigate the effect of the non-deterministic nature of multi-processing.

Metrics Following the view synthesis SLAM literature convention, we evaluate our system using PSNR, SSIM, and LPIPS [46]. We also provide depth L1[cm] metric compared to the ground truth depth in the Replica dataset. The evaluation is performed after post-processing every 5 frames in sequences skipping the keyframes used for mapping. This approach aligns with the evaluation methods used in MonoGS [16] and Splat-SLAM [27].

Implementation Details Our system runs on a PC with a 3.6GHz AMD Ryzen Threadripper PRO 5975WX and an NVIDIA RTX 4090 GPU. In all our experiments, we set $l = 0.8$, $\theta = 128$, $\alpha = 0.5$, $\mathrm{lr}_i = 1.6 \times 10^{-4}$, $\mathrm{lr}_f = 1.6 \times 10^{-6}$, $\tau = 3000$ for hyperparameters in mapping. We set $\beta = 2000$ for the EuRoC [3] and Replica [29] datasets and $\beta = 26000$ for the TUM RGB-D [30] and the ScanNet [6] datasets. These values are consistent with those used in MonoGS [43] and Splat-SLAM [27]. For tracking, pre-trained GRU weights from DROID-SLAM [34] are utilized. We set the mean optical flow threshold for keyframe selection to 4.0 pixels, and the local dense bundle adjustment window to 16. Optimizations in the tracking module are performed in LieTorch [35] framework. The mapping process accepts only the latest keyframe created after finishing its optimization step if the latest keyframe is not already in the sliding window.

Baselines We compare our system to state-of-the-art RGB-only Gaussian Splatting and NeRF SLAM algorithms, including MonoGS [16], Photo-SLAM [11], GlORIE-SLAM [45], and Splat-SLAM [27].

MonoGS [16] is the state-of-the-art representation-only SLAM algorithm that utilizes the Gaussian scene representation for tracking and mapping. Photo-SLAM, like GlORIE-SLAM [45], Splat-SLAM [27], and our system, features a decoupled design for tracking and mapping. One key difference is that Photo-SLAM lacks dense depth maps while mapping. GlORIE-SLAM and Splat-SLAM utilize monocular depth estimation [7] and the dense bundle adjustment layer. The most important difference between them is that GlORIE-SLAM [45] models the scene with NeRF [17] and Splat-SLAM [27] does so with 3D Gaussian Splatting [14].

4.2. Evaluation

We compare our system with state-of-the-art algorithms based on rendering quality, 3D reconstruction accuracy, and runtime performance.

Rendering and Reconstruction Accuracy We evaluate rendering and reconstruction accuracy for the Replica [29] in Tab. 1. Our algorithm's performance is quite similar to Splat-SLAM [27] in Replica [29]. In Tab. 2, we compare head-to-head with GlORIE-SLAM [45] on the ScanNet [6], where we trail behind Splat-SLAM [27]. In Tab. 3, we rank just behind Splat-SLAM, outperforming other algorithms on the TUM RGB-D [30] dataset. However, we are superior in terms of on-the-fly map optimization to Splat-SLAM as shown in Tab. 6. We place the first in the the EuRoC [3] dataset demonstrating a significant margin over Photo-SLAM [11]. A qualitative comparison

is shown in Fig. 1. Our experiments reveal that sequences focusing on a centered object in an unbounded scene, such as TUM-RGBD f3/off, are particularly challenging.

Metrics	Mono-GS [16]	GlORIE-SLAM [45]	Photo-SLAM [11]	Splat-SLAM [27]	Ours
PSNR↑	31.22	31.04	33.30	**36.45**	36.21
SSIM↑	0.91	0.91	0.93	0.95	**0.96**
LPIPS↓	0.21	0.12	-	0.06	**0.05**
Depth L1↓	-	-	-	**2.41**	4.34

Table 1. **Rendering and Tracking Results on Replica [29] for RGB-Methods.** The results are averaged over 8 scenes and each scene result is the average of 3 runs. We take the numbers from [27] except for ours. The best results are highlighted as first , second . Our method shows similar performance to Splat-SLAM [27] and outperforms all the other methods.

Method	Metric	0000	0059	0106	0169	0181	0207	Avg.
MonoGS [16]	PSNR↑	16.91	19.15	18.57	20.21	19.51	18.37	18.79
	SSIM↑	0.62	0.69	0.74	0.74	0.75	0.70	0.71
	LPIPS↓	0.70	0.51	0.55	0.54	0.63	0.58	0.59
GlORIE-SLAM [45]	PSNR↑	23.42	20.66	20.41	25.23	21.28	23.68	22.45
	SSIM↑	**0.87**	**0.87**	0.83	0.84	**0.91**	0.76	**0.85**
	LPIPS↓	0.26	0.31	0.31	0.21	0.44	0.29	0.30
Splat-SLAM [27]	PSNR↑	**28.68**	**27.69**	**27.70**	**31.14**	**31.15**	**30.49**	**29.48**
	SSIM↑	0.83	**0.87**	**0.86**	**0.87**	0.84	**0.84**	**0.85**
	LPIPS↓	**0.19**	**0.15**	**0.18**	**0.15**	**0.23**	**0.19**	**0.18**
IG-SLAM (Ours)	PSNR↑	24.68	20.09	25.30	27.85	25.80	26.69	25.07
	SSIM↑	0.74	0.68	0.83	0.82	0.83	0.78	0.78
	LPIPS↓	0.29	0.39	0.22	0.19	0.27	0.27	0.27

Table 2. **Rendering Performance on ScanNet [6].** Each scene result is the average of 3 runs. We take the numbers from [27] except for ours. Our method shows competitive performance to the state-of-the-art methods exhibiting the second high visual quality results.

Runtime Analysis We assess real-time performance of our algorithm in Tab. 5. We benchmark the runtime on a 3.6GHz AMD Ryzen Threadripper PRO 5975WX and an NVIDIA GeForce RTX 4090 with 24 GB of memory. Our system operates at 9.94 fps, making it 8 times faster than Splat-SLAM [27] in a single-process implementation. Our method outperforms other algorithms without compromising visual quality. The reference multi-process implementation of our method achieves a frame rate of 16 fps. Our method's peak memory consumption and map size are comparable to existing methods.

4.3. Ablations

Post-processing, decay, and weighted depth loss are our system design choices. We present ablation studies to validate and support each of these design decisions.

Post Processing We show post processing ablation results in Tab. 6. PSNR and Depth L1 metrics are recalculated

Method	Metric	f1/desk	f2/xyz	f3/off	Avg.
Photo-SLAM [11]	PSNR↑	20.97	21.07	19.59	20.54
	SSIM↑	0.74	0.73	0.69	0.72
	LPIPS↓	0.23	0.17	0.24	0.21
MonoGS [16]	PSNR↑	19.67	16.17	20.63	18.82
	SSIM↑	0.73	0.72	0.77	0.74
	LPIPS↓	0.33	0.31	0.34	0.33
GlORIE-SLAM [45]	PSNR↑	20.26	25.62	21.21	22.36
	SSIM↑	0.79	0.72	0.72	0.74
	LPIPS↓	0.31	0.09	0.32	0.24
Splat-SLAM [27]	PSNR↑	**25.61**	**29.53**	**26.05**	**27.06**
	SSIM↑	**0.84**	**0.90**	**0.84**	**0.86**
	LPIPS↓	**0.18**	**0.08**	0.20	**0.15**
IG-SLAM (Ours)	PSNR↑	24.45	26.35	25.27	25.36
	SSIM↑	0.80	0.85	0.83	0.83
	LPIPS↓	0.20	0.10	**0.17**	0.16

Table 3. **Rendering Performance on TUM-RGBD [30].** Each scene result is the average of 3 runs. We take the numbers from [27] except for ours. Our method demonstrates similar performance to Splat-SLAM [27] in challenging indoor environments showing a clear performance margin to the other methods.

Method	Metric	MH-01	MH-02	V1-01	V2-01	**Avg.**
Photo-SLAM [11]	PSNR↑	13.95	14.20	17.07	15.68	15.23
	SSIM↑	0.42	0.43	0.62	0.62	0.52
	LPIPS↓	0.37	0.36	**0.27**	0.32	0.33
IG-SLAM (Ours)	PSNR↑	**22.33**	**22.31**	**20.55**	**24.59**	**22.44**
	SSIM↑	**0.78**	**0.77**	**0.79**	**0.85**	**0.80**
	LPIPS↓	**0.22**	**0.23**	0.29	**0.18**	**0.23**

Table 4. **Rendering Performance on EuRoC [3].** Each scene result is the average of 3 runs. We take the numbers for Photo-SLAM [11] from their work. We successfully show the scalability of our system. Photorealistic 3D reconstruction comparison of large indoor environment EuRoC [3] MH-01 is shown in Fig. 1 .

	GO-SLAM [47]	GlORIE-SLAM [45]	MonoGS [16]	Splat-SLAM [27]	**Ours**
GPU Usage [GiB]	18.50	15.22	**14.62**	17.57	16.20
Map Size [MB]	-	114.0	6.8	**6.5**	14.8
Avg. FPS	8.36	0.23	0.32	1.24	**9.94**

Table 5. Memory and Running Time Evaluation on Replica [29] room0. We measure the runtime statistics on the single process implementation of our method. We take the numbers from [27] except for ours. Our peak memory usage and map size are comparable to existing works. Our method achieves to exhibit state-of-the-art 3D reconstruction in higher frame rates compared to other methods.

for every 500 post-processing iterations. Our method exhibits a relatively small visual quality degradation when post-processing is skipped (indicated as 0K in Tab. 6) whereas visual quality significantly drops with no post-processing for Splat-SLAM [27]. Our system exhibits diminishing returns with increased post-processing iterations. We attribute the fast convergence of our map and the minimal reliance on post-processing to our training strategy.

Nbr of Final Iterations β	Metric	0K	0.5K	1K	2K
Splat-SLAM [27]	PSNR ↑	30.50	39.87	40.59	41.20
	Depth L1 ↓	6.55	2.37	2.34	2.40
Ours	PSNR ↑	**38.30**	**40.92**	**41.53**	**41.68**
	Depth L1 ↓	**2.63**	**2.18**	**2.17**	**2.30**

Table 6. **Post-processing iterations ablation on Replica [29] `office0`**. The numbers for Splat-SLAM [27] are taken from their work. Due to the fast convergence of mapping during tracking, we do not heavily rely on post-processing. The reconstruction benefits only a little from post-processing.

Decay We demonstrate learning rate decay ablation in Tab. 7. We compare 3 learning rates without decay with decaying learning rates. The selected 3 learning rates are $lr_f) = 1.6 \times 10^{-6}$ for lower bound, $lr_i) = 1.6 \times 10^{-4}$ for upper bound, and the mean learning rate value 5×10^{-5} calculated according to Eq. (10). We conduct this experiment with and without post-processing. As seen in no post-processing experiment in Tab. 7, learning with decay greatly enhances the visual quality compared to other non-decaying learning rate setups. Qualitative results are shown in Fig. 5. As observed, the fine details are not captured with non-decaying learning rates. Moreover, a post-processing step completely shadows the convergence problems of constant learning rate as seen in the experiment with post-processing in Tab. 7.

Metric	Learning Rate	1.6×10^{-6}	5×10^{-5}	1.6×10^{-4}	1.6×10^{-4} w/ decay
w/o Post Processing					
PSNR ↑		31.92	35.84	34.71	**38.30**
Depth L1 ↓		5.37	2.71	2.76	**2.63**
w/ Post Processing					
PSNR ↑		39.71	39.91	40.85	**41.68**
Depth L1 ↓		2.73	**2.17**	2.20	2.30

Table 7. **Learning Rate Hyperparameter Search on Replica [29] `office0`**. Our system benefits greatly from a slow learning rate combined with decay. In the presence of reliable depth maps, a high learning rate contributes to TV-static noise and slows down map convergence.

Depth Loss The weighted depth loss ablation results are shown in Tab. 8. The weighted depth loss that is given in Eq. (9) is compared to the scenarios with no depth loss in the overall loss function ($\alpha = 1$) and with raw depth values without weighting them by depth covariance. Post-processing is disabled to ensure the results are not obscured.

Metric	Weighted	No Depth	Raw Depth
PSNR↑	**31.91**	31.56	30.81
Depth L1 ↓	**6.33**	13.16	6.39

Table 8. **Weighted Depth Loss Ablation on Replica [29] `office2`.** Weighted depth loss enables better reconstruction without decreasing visual quality.

The weighted loss is superior to other choices as observed in Tab. 8. A pure color loss performs well in terms of visual quality but deteriorates reconstruction quality. Using raw depth values in the loss function performs worse than the weighted loss regarding visual quality. Therefore, weighting the depth prevents visual quality from decreasing due to high uncertainty regions while keeping the reconstruction quality up by supervising depth. We speculate visual quality differences are not dramatic because our system initializes Gaussians according to depth maps regardless of the loss function. Therefore, initialized Gaussians are already in the vicinity of the corresponding depth value.

Figure 5. **Qualitative results for learning rate decay ablation study.** The four cases studied in Tab. 7 are shown in the figure. The results are given as constant learning rates of 1.6×10^{-4} at *top-left*, 5×10^{-5} at *top-right*, 1.6×10^{-6} at *bottom-left* and the decaying 1.6×10^{-4} learning rate at *bottom-left* as reference.

5. Limitations

The dense bundle adjustment is not feasible in full resolution. Therefore, dense depth maps are optimized at a lower resolution and upsampled back to the original resolution. We observe that this upsampling operation results in blurry edges. Therefore, utilizing upsampled dense depth maps to supervise the system results in poor performance at locations where sharp changes in depth occur.

6. Conclusion

We showed that the depth supervision from a robust dense-SLAM method greatly enhances 3D reconstruction performance. Additionally, utilizing depth uncertainty as a mask for Gaussian initialization and as weights for depth loss aids the mapping process. We also highlighted the nuance between sparse and dense Gaussian initialization and its implications on mapping optimization. Our experiments demonstrated that dense SLAM-based 3D reconstruction can provide both state-of-the-art visual quality and a high frame rate even in relatively large scenes.

References

[1] Jonathan T Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P Srinivasan. Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 5855–5864, 2021. 2

[2] Michael Bloesch, Jan Czarnowski, Ronald Clark, Stefan Leutenegger, and Andrew J. Davison. Codeslam - learning a compact, optimisable representation for dense visual SLAM. *CoRR*, abs/1804.00874, 2018. 2

[3] Michael Burri, Janosch Nikolic, Pascal Gohl, Thomas Schneider, Joern Rehder, Sammy Omari, Markus W Achtelik, and Roland Siegwart. The euroc micro aerial vehicle datasets. *The International Journal of Robotics Research*, 35 (10):1157–1163, 2016. 1, 6, 7, 2

[4] Danpeng Chen, Hai Li, Weicai Ye, Yifan Wang, Weijian Xie, Shangjin Zhai, Nan Wang, Haomin Liu, Hujun Bao, and Guofeng Zhang. Pgsr: Planar-based gaussian splatting for efficient and high-fidelity surface reconstruction. *arXiv preprint arXiv:2406.06521*, 2024. 2

[5] Chi-Ming Chung, Yang-Che Tseng, Ya-Ching Hsu, Xiang-Qian Shi, Yun-Hung Hua, Jia-Fong Yeh, Wen-Chin Chen, Yi-Ting Chen, and Winston H Hsu. Orbeez-slam: A real-time monocular visual slam with orb features and nerf-realized mapping. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9400–9406. IEEE, 2023. 1, 2

[6] Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas A. Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. *CoRR*, abs/1702.04405, 2017. 6, 7, 1

[7] Ainaz Eftekhar, Alexander Sax, Jitendra Malik, and Amir Zamir. Omnidata: A scalable pipeline for making multi-task mid-level vision datasets from 3d scans. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10786–10796, 2021. 1, 6

[8] Peter Hedman, Pratul P. Srinivasan, Ben Mildenhall, Jonathan T. Barron, and Paul Debevec. Baking neural radiance fields for real-time view synthesis. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 5875–5884, 2021. 2

[9] Tao Hu, Shu Liu, Yilun Chen, Tiancheng Shen, and Jiaya Jia. Efficientnerf efficient neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12902–12911, 2022. 2

[10] Binbin Huang, Zehao Yu, Anpei Chen, Andreas Geiger, and Shenghua Gao. 2d gaussian splatting for geometrically accurate radiance fields. In *ACM SIGGRAPH 2024 Conference Papers*, pages 1–11, 2024. 2

[11] Huajian Huang, Longwei Li, Hui Cheng, and Sai-Kit Yeung. Photo-slam: Real-time simultaneous localization and photo-realistic mapping for monocular stereo and rgb-d cameras. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21584–21593, 2024. 1, 2, 3, 4, 5, 6, 7

[12] Mohammad Mahdi Johari, Camilla Carta, and François 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*, pages 17408–17419, 2023. 1

[13] Nikhil Keetha, Jay Karhade, Krishna Murthy Jatavallabhula, Gengshan Yang, Sebastian Scherer, Deva Ramanan, and Jonathon 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*, pages 21357–21366, 2024. 3

[14] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42 (4), 2023. 1, 2, 3, 4, 6

[15] Lingjie Liu, Jiatao Gu, Kyaw Zaw Lin, Tat-Seng Chua, and Christian Theobalt. Neural sparse voxel fields. *Advances in Neural Information Processing Systems*, 33:15651–15663, 2020. 2

[16] Hidenobu Matsuki, Riku Murai, Paul HJ Kelly, and Andrew J Davison. Gaussian splatting slam. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18039–18048, 2024. 1, 3, 4, 5, 6, 7

[17] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *CoRR*, abs/2003.08934, 2020. 1, 2, 6

[18] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM transactions on graphics (TOG)*, 41(4):1–15, 2022. 2, 4, 5

[19] Raul Mur-Artal and Juan D Tardós. Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE transactions on robotics*, 33(5):1255–1262, 2017. 1, 2, 3

[20] Richard A. Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J. Davison, Pushmeet Kohi, Jamie Shotton, Steve Hodges, and Andrew Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *2011 10th IEEE International Symposium on Mixed and Augmented Reality*, pages 127–136, 2011. 2

[21] Richard A. Newcombe, Steven J. Lovegrove, and Andrew J. Davison. Dtam: Dense tracking and mapping in real-time. In *2011 International Conference on Computer Vision*, pages 2320–2327, 2011. 2

[22] Michael Oechsle, Songyou Peng, and Andreas Geiger. Unisurf: Unifying neural implicit surfaces and radiance fields for multi-view reconstruction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5589–5599, 2021. 2

[23] Zhexi Peng, Tianjia Shao, Yong Liu, Jingke Zhou, Yin Yang, Jingdong Wang, and Kun Zhou. Rtg-slam: Real-time 3d reconstruction at scale using gaussian splatting. In *ACM SIGGRAPH 2024 Conference Papers*, pages 1–11, 2024. 3

[24] Christian Reiser, Songyou Peng, Yiyi Liao, and Andreas Geiger. Kilonerf: Speeding up neural radiance fields with

thousands of tiny mlps. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 14335–14345, 2021. 1

[25] Konstantinos Rematas and Vittorio Ferrari. Neural voxel renderer: Learning an accurate and controllable rendering tool. In *CVPR*, 2020. 2

[26] Antoni Rosinol, John Leonard, and Luca Carlone. Nerfslam: Real-time dense monocular slam with neural radiance fields. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3437–3444. IEEE, 2023. 1, 2, 4

[27] Erik Sandström, Keisuke Tateno, Michael Oechsle, Michael Niemeyer, Luc Van Gool, Martin R Oswald, and Federico Tombari. Splat-slam: Globally optimized rgb-only slam with 3d gaussians. *arXiv preprint arXiv:2405.16544*, 2024. 2, 3, 5, 6, 7, 8

[28] Cameron Smith, David Charatan, Ayush Tewari, and Vincent Sitzmann. Flowmap: High-quality camera poses, intrinsics, and depth via gradient descent. *arXiv preprint arXiv:2404.15259*, 2024. 2

[29] Julian Straub, Thomas Whelan, Lingni Ma, Yufan Chen, Erik Wijmans, Simon Green, Jakob J. Engel, Raul Mur-Artal, Carl Yuheng Ren, Shobhit Verma, Anton Clarkson, Mingfei Yan, Brian Budge, Yajie Yan, Xiaqing Pan, June Yon, Yuyang Zou, Kimberly Leon, Nigel Carter, Jesus Briales, Tyler Gillingham, Elias Mueggler, Luis Pesqueira, Manolis Savva, Dhruv Batra, Hauke M. Strasdat, Renzo De Nardi, Michael Goesele, Steven Lovegrove, and Richard A. Newcombe. The replica dataset: A digital replica of indoor spaces. *CoRR*, abs/1906.05797, 2019. 6, 7, 8, 1

[30] Jürgen Sturm, Nikolas Engelhard, Felix Endres, Wolfram Burgard, and Daniel Cremers. A benchmark for the evaluation of rgb-d slam systems. In *2012 IEEE/RSJ international conference on intelligent robots and systems*, pages 573–580. IEEE, 2012. 6, 7, 1

[31] Edgar Sucar, Shikun Liu, Joseph Ortiz, and Andrew J Davison. imap: Implicit mapping and positioning in real-time. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6229–6238, 2021. 1, 2

[32] Chengzhou Tang and Ping Tan. Ba-net: Dense bundle adjustment network. *CoRR*, abs/1806.04807, 2018. 2

[33] Zachary Teed and Jia Deng. Raft: Recurrent all-pairs field transforms for optical flow. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part II 16*, pages 402–419. Springer, 2020. 2

[34] Zachary Teed and Jia Deng. DROID-SLAM: deep visual SLAM for monocular, stereo, and RGB-D cameras. *CoRR*, abs/2108.10869, 2021. 1, 2, 3, 6

[35] Zachary Teed and Jia Deng. Tangent space backpropagation for 3d transformation groups. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10338–10347, 2021. 6

[36] Hengyi Wang, Jingwen Wang, and Lourdes 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*, pages 13293–13302, 2023. 1

[37] Liao Wang, Jiakai Zhang, Xinhang Liu, Fuqiang Zhao, Yanshun Zhang, Yingliang Zhang, Minye Wu, Jingyi Yu, and Lan Xu. Fourier plenoctrees for dynamic radiance field rendering in real-time. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13524–13534, 2022. 2

[38] Peng Wang, Lingjie Liu, Yuan Liu, Christian Theobalt, Taku Komura, and Wenping Wang. Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. *arXiv preprint arXiv:2106.10689*, 2021. 2

[39] Chi Yan, Delin Qu, Dan Xu, Bin Zhao, Zhigang Wang, Dong Wang, and Xuelong Li. Gs-slam: Dense visual slam with 3d gaussian splatting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 19595–19604, 2024. 1, 3

[40] Lior Yariv, Jiatao Gu, Yoni Kasten, and Yaron Lipman. Volume rendering of neural implicit surfaces. *Advances in Neural Information Processing Systems*, 34:4805–4815, 2021. 2

[41] Lin Yen-Chen, Pete Florence, Jonathan T Barron, Alberto Rodriguez, Phillip Isola, and Tsung-Yi Lin. inerf: Inverting neural radiance fields for pose estimation. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1323–1330. IEEE, 2021. 2

[42] Alex Yu, Ruilong Li, Matthew Tancik, Hao Li, Ren Ng, and Angjoo Kanazawa. Plenoctrees for real-time rendering of neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5752–5761, 2021. 2

[43] Vladimir Yugay, Yue Li, Theo Gevers, and Martin R Oswald. Gaussian-slam: Photo-realistic dense slam with gaussian splatting. *arXiv preprint arXiv:2312.10070*, 2023. 1, 3, 6

[44] Baowen Zhang, Chuan Fang, Rakesh Shrestha, Yixun Liang, Xiaoxiao Long, and Ping Tan. Rade-gs: Rasterizing depth in gaussian splatting. *arXiv preprint arXiv:2406.01467*, 2024. 2

[45] Ganlin Zhang, Erik Sandström, Youmin Zhang, Manthan Patel, Luc Van Gool, and Martin R Oswald. Glorie-slam: Globally optimized rgb-only implicit encoding point cloud slam. *arXiv preprint arXiv:2403.19549*, 2024. 1, 2, 5, 6, 7

[46] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 586–595, 2018. 6

[47] Youmin Zhang, Fabio Tosi, Stefano Mattoccia, and Matteo Poggi. Go-slam: Global optimization for consistent 3d instant reconstruction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3727–3737, 2023. 1, 4, 7

[48] Heng Zhou, Zhetao Guo, Shuhong Liu, Lechen Zhang, Qihao Wang, Yuxiang Ren, and Mingrui Li. Mod-slam: Monocular dense mapping for unbounded 3d scene reconstruction. *arXiv preprint arXiv:2402.03762*, 2024. 1, 2

[49] Zihan Zhu, Songyou Peng, Viktor Larsson, Weiwei Xu, Hujun Bao, Zhaopeng Cui, Martin R Oswald, and Marc Pollefeys. Nice-slam: Neural implicit scalable encoding for slam.

In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12786–12796, 2022. 1, 2

[50] Zihan Zhu, Songyou Peng, Viktor Larsson, Zhaopeng Cui, Martin R Oswald, Andreas Geiger, and Marc Pollefeys. Nicer-slam: Neural implicit scene encoding for rgb slam. In *2024 International Conference on 3D Vision (3DV)*, pages 42–52. IEEE, 2024. 1, 2

[51] Matthias Zwicker, Hanspeter Pfister, Jeroen Van Baar, and Markus Gross. Ewa volume splatting. In *Proceedings Visualization, 2001. VIS'01.*, pages 29–538. IEEE, 2001. 4

IG-SLAM: Instant Gaussian SLAM

Supplementary Material

IG-SLAM is a dense SLAM system capable of photorealistic 3D reconstruction, while simultaneously running at high frame rates. In this supplementary material, we provide additional results.

7. Method

We describe additional details about our method.

7.1. Covariance Mask

Assume the covariance for a depth map is given by Σ, we normalize covariance between [0,1] by Eq. (11)

$$\tilde{\Sigma}(u,v) = \frac{\Sigma(u,v) - \min\left(\Sigma(u,v)\right)}{\max\left(\Sigma(u,v)\right) - \min\left(\Sigma(u,v)\right)} \qquad (11)$$

where (u,v) are the pixel coordinates. A Maximum filter with a kernel size of 32 is applied to normalized covariance. Pixels with normalized covariance less than 0.2 are selected. Additionally, a majority filter with a kernel size of 32 is applied to obtain smooth valid regions in the mask.

7.2. Pruning and Densification

We follow the same procedure for pruning and identification in MonoGS [16] n. Pruning is based on occlusion-aware visibility: if new Gaussians initialized in the last keyframes are not visible from this keyframe at the end of the optimization, they are removed. Additionally, for every 150 mapping iterations, Gaussians with opacity lower than 0.1 are removed. Densification is performed by splitting large Gaussians and cloning small ones in regions with high loss gradients, also every 150 mapping iterations.

8. Additional Results

We provide additional tracking and mapping results.

9. Tracking

We do not improve over GO-SLAM [47] in terms of tracking performance, as it is outside the scope of our work. However, we include the tracking results of Replica [29], TUM-RGB-D [30], and ScanNet [6] in Tab. 9, Tab. 10, and Tab. 11 for reference.

Metric	R-O	R-1	R-2	O-0	O-1	O-2	O-3	O-4
ATE(cm)	0.45	0.39	0.31	0.33	0.50	0.39	0.47	0.68

Table 9. **Tracking Accuracy ATE RMSE [cm] ↓ on Replica [29].** Each scene result is the average of 3 runs.

Metric	f1/desk	f2/xyz	f3/off
ATE(cm)	2.73	0.35	2.08

Table 10. **Tracking Accuracy ATE RMSE [cm] ↓ on TUM-RGBD [30].** Each scene result is the average of 3 runs.

Metric	0000	0059	0106	0169	0181	0207
ATE(cm)	6.16	71.46	7.38	8.46	8.60	9.55

Table 11. **Tracking Accuracy ATE RMSE [cm] ↓ on ScanNet [6].** Each scene result is the average of 3 runs.

9.1. Mapping

The results of each scene of the Replica [29] are given in Tab. 12. Full evaluations on EuRoC [3] Machine Hall and Vicon Room are given in Tab. 13 and Tab. 14. Moreover, additional qualitative results of EuRoC [3] are exhibited in Fig. 6

Metric	R-0	R-1	R-2	O-0	O-1	O-2	O-3	O-4
PSNR↑	32.33	34.64	35.29	41.68	41.30	34.68	34.92	34.80
SSIM ↑	0.93	0.95	0.96	0.98	0.98	0.95	0.96	0.96
LPIPS↓	0.07	0.06	0.05	0.02	0.03	0.06	0.05	0.07
Depth L1↓	4.79	3.04	4.15	2.23	1.94	6.40	7.67	4.45

Table 12. **Full evaluation on Replica [29].** Each scene result is the average of 3 runs.

Metric	MH-01	MH-02	MH-03	MH-04	MH-05
PSNR↑	22.33	22.31	20.78	23.62	19.85
SSIM ↑	0.78	0.77	0.71	0.82	0.70
LPIPS↓	0.22	0.23	0.28	0.19	0.35

Table 13. **Full evaluation on EuRoC [3] `Machine Hall`.** Each scene result is the average of 3 runs.

Metric	V1-01	V1-02	V1-03	V2-01	V2-02	V2-03
PSNR↑	20.55	22.86	20.11	24.59	23.70	21.62
SSIM ↑	0.79	0.84	0.74	0.85	0.83	0.74
LPIPS↓	0.29	0.26	0.42	0.18	0.23	0.41

Table 14. **Full evaluation on EuRoC [3] `Vicon Room`** Each scene result is the average of 3 runs.

Figure 6. **Qualitative results of IG-SLAM on EuRoC [3].** The results in the *top row*, *middle row*, and *bottom row* are from `MH-02`, `MH-03`, `V1-01` respectively.