RGS-SLAM: Robust Gaussian Splatting SLAM with One-Shot Dense Initialization

Supplementary Material

1. Reproducibility and Code Release

All components of the system are implemented in PyTorch with custom CUDA kernels for rasterization and analytic Jacobians. The repository provides the full SLAM pipeline, configuration files, scripts for evaluation on TUM RGB-D and Replica, and instructions for reproducing all quantitative tables and qualitative renderings in the main paper. The training and optimization settings match those described in Sections 3 and 4 of the main paper. An anonymized version of the codebase is available at:
https://anonymous.4open.science/r/RGS-SLAM

2. Discussion

2.1. Limitations

RGS-SLAM still exhibits several practical limitations despite the gains over residual-driven densification. The current evaluation focuses on indoor static scenes in Replica and TUM RGB-D, so the robustness of the one-shot Gaussian initialization in highly dynamic scenes, or under severe rolling shutter remains unclear. The dense correspondences are derived from a single pretrained DINOv3 backbone together with a confidence aware inlier classifier, which can degrade under drastic appearance changes, strong motion blur, or camera viewpoints far outside the training distribution, and in these regimes the triangulated seeds may become biased or incomplete. The fixed topology after each keyframe initialization improves stability but can leave persistent coverage gaps when large textureless surfaces, fine specular structures, or objects with weak visual support never accumulate enough consistent matches, which constrains the reconstruction quality. The current implementation also assumes a calibrated pinhole camera and synchronized RGB-D streams, without exploiting lidar cues or inertial measurements that are available on many robotic platforms. Finally, the system still requires a GPU with moderate memory to sustain high frame rates, and the practical impact of memory budgets, long-term map growth, and large-scale loop closure has not yet been characterized on resource constrained devices.

2.2. Societal Impact

RGS-SLAM advances camera based dense SLAM with a training-free one-shot Gaussian initialization that stabilizes optimization and improves throughput, which can benefit robotics, extended reality, and digital twin systems through more reliable mapping and safer physical interaction. At the same time, dense reconstruction of indoor environments raises privacy risks whenever RGB or RGB-D streams are captured and stored without informed consent, since metrically consistent maps can reveal layouts, personal belongings, and usage patterns. The method is training-free and easily integrated into existing SLAM stacks, which lowers the barrier for large-scale deployment and makes responsible use dependent on appropriate safeguards such as transparent data handling, limited retention of raw sensor data, access control to stored maps, and a preference for local processing. Our experiments rely on public benchmarks without personally identifiable information and the code release is intended for reproducible research, although future work should pair technical advances in robust mapping with privacy aware data design and interdisciplinary guidelines for ethical deployment.

2.3. Training Details

Optimization of Gaussian maps. The optimization settings used by our Gaussian SLAM system are summarized in Table 1. For both initialization and online mapping we use the Adam optimizer with a shared parameterization for geometry and appearance. The base learning rate is set to 2.0×10^{-3} for color and opacity parameters and 1.0×10^{-3} for positions and rotations, with $(\beta_1, \beta_2) = (0.9, 0.999)$. A cosine decay schedule is applied within each optimization window so that early iterations focus on rapid geometry refinement while later iterations stabilize the map. Initialization uses a window of five frames and runs for 1050 iterations before the system starts live tracking. During online operation every incoming frame is refined for 30 tracking iterations and each accepted keyframe-triggers 60 mapping iterations over the same local window. The loss combines an L_1 term and an SSIM term weighted by $\lambda_{\text{dssim}} = 0.2$. Dynamic density control uses the same thresholds across all experiments and adopts the opacity culling and densification settings listed in Table 1. Mixed precision training and gradient norm clipping are enabled to keep the per frame compute budget and memory stable.

Dense initialization baseline. As an additional baseline we adopt a dense correspondence based initialization strategy that shares the same differentiable Gaussian representation as our system. The training configuration is summarized in Table 2. The model is optimized for 30000 iterations with Adam and separate learning rates for spatial and appearance parameters. Position updates follow a decayed

Table 1. Optimization configuration for the Gaussian SLAM.

Config	Value
Optimizer	Adam
Adam betas	$\beta_1 = 0.9, \beta_2 = 0.999$
Base LR (color, opacity)	2.0×10^{-3}
Base LR (position, rotation)	1.0×10^{-3}
Learning rate schedule	Cosine decay
Loss	$L_1 + \lambda_{\text{dssim}} L_{\text{SSIM}}$
SSIM weight λ_{dssim}	0.2
Initialization iterations	1050
Tracking iterations per frame	30
Mapping iterations per keyframe	60
Local window size	5 frames
Densification interval	100 iterations
Opacity reset interval	3000 iterations
Opacity culling threshold	0.05
Gradient clipping max norm	1.0
Precision	Mixed FP16 / FP32

Table 2. Training configuration of the dense initialization baseline.

Config	Value
Total iterations	30000
Optimizer	Adam
Spherical harmonics degree	3
Batch size	64
Position LR (init $\rightarrow$ final)	$1.6 \times 10^{-4} \rightarrow 1.6 \times 10^{-6}$
Feature learning rate	2.5×10^{-3}
Opacity learning rate	2.5×10^{-2}
Scaling learning rate	5.0×10^{-3}
Rotation learning rate	1.0×10^{-3}
SSIM weight λ_{dssim}	0.2
Dense correspondence ratio	0.01
Densification interval	100 iterations
Densification range	iter. 500 to 15000
Densification gradient threshold	2.0×10^{-4}
Opacity reset iteration	30000
Exposure LR (init $\rightarrow$ final)	$1.0 \times 10^{-2} \rightarrow 1.0 \times 10^{-4}$
Random background	disabled
Logging train camera index	50
Logging test camera index	10
Dataset resolution	original
White background	false
Data device	CUDA

schedule from 1.6×10^{-4} to 1.6×10^{-6}, while feature, opacity, scaling, and rotation parameters use constant rates that match the public configuration. The reconstruction loss combines an ℓ_1 term with a structural similarity component with $\lambda_{\text{dssim}} = 0.2$. Densification is enabled from iteration 500 to 15000 with an interval of 100 iterations and a gradient based threshold of 2.0×10^{-4}. All experiments use degree three spherical harmonics, batch size 64, and a fixed background color without random perturbations on a CUDA device.

3. Implementation Details

Datasets. We evaluate on the TUM RGB-D and Replica benchmarks, consistent with Section 4 of the main paper. TUM RGB-D is used in both monocular and RGB-D configurations, following standard practice in visual SLAM. Replica is employed for photometric map evaluation and trajectory accuracy on the eight standard indoor scenes *room0* to *room2* and *office0* to *office4*. These splits match those used in the main tables so that rendering metrics, throughput, and localization error remain directly comparable across methods.

Evaluation Metrics. Camera tracking accuracy is measured using the root mean square error of the Absolute Trajectory Error (ATE) computed on keyframes. Photometric map quality is evaluated with three rendering metrics: PSNR, SSIM, and LPIPS. Geometric reconstruction quality is assessed with accuracy (Acc. [cm]), completeness (Comp. [cm]), and completeness ratio (Comp.Ratio [%]). Accuracy is defined as the mean nearest-neighbour distance from reconstructed points to the ground-truth surface. Completeness is the mean nearest-neighbour distance from ground-truth surface samples to the reconstruction. The completeness ratio is the proportion of ground-truth samples within a distance threshold τ from the reconstruction. Unless stated otherwise, we uniformly sample 50K points on each surface, set $\tau = 5\,\text{cm}$, and average per-scene scores over the benchmark sequences. For photometric rendering metrics, we evaluate every fifth frame and exclude keyframes to avoid bias toward training views. Reconstruction metrics are computed with the same surface-sampling protocol. Each experiment is repeated three times on identical hardware and stopping criteria, and all tables report the mean scores. In every table, the best result is typeset in bold and the second best is underlined.

3.1. Compare Model Settings

We compare RGS-SLAM with a set of representative dense SLAM pipelines that cover implicit volumes, voxel grids, point clouds, and Gaussian splats under the same scene types and benchmarks. NICE-SLAM, Co-SLAM, Vox-Fusion, DI-Fusion, and SNI-SLAM operate on RGB-D input and maintain volumetric implicit or voxel based representations that are optimized per scene. iMAP and Point-SLAM map monocular or RGB-D streams to neural fields or point clouds with scene specific training and joint pose refinement. SplaTAM, Gauss-SLAM, MonoGS, and RK-

CVPR #9001

SLAM adopt 3D Gaussian splatting and couple a Gaussian renderer with pose and appearance optimization, while Photo-SLAM and GLORIE-SLAM combine differentiable rendering with explicit point or mesh structures.

3.2. Computing Resource Configuration

All experiments are run on a workstation equipped with two NVIDIA L40 GPUs and an Intel Xeon Platinum 8362 CPU at 2.80 GHz. Time-critical components, including 3D Gaussian rasterization and gradient computation, are implemented in CUDA, while the remaining SLAM pipeline is implemented in PyTorch. Mixed precision is enabled for rendering and backpropagation whenever this improves throughput without degrading stability. The tracking loop operates in real-time, and mapping is executed asynchronously within a bounded local window so that latency remains stable as the map grows.

4. Methodology Details

4.1. Gaussian Splatting Representation

Each Gaussian G_i is represented by a compact tuple containing the world space mean $\mu_i^W \in \mathbb{R}^3$, an opacity parameter $\alpha_i \in [0,1]$, a set of view dependent color coefficients, and a covariance parameterization. In the underlying model the covariance appears as a full matrix Σ_i^W, while in the implementation it is encoded as a rotation matrix $R_i \in \mathrm{SO}(3)$ and three axis aligned scales $s_i \in \mathbb{R}_+^3$ such that

$$\Sigma_i^W \;=\; R_i \,\mathrm{diag}(s_i^2)\, R_i^\top. \tag{1}$$

The rotation is stored as a unit quaternion and the scales are optimized in logarithmic space, which guarantees positive definiteness under gradient updates.

Colors are represented by second order spherical harmonics in camera space. For a viewing direction $\mathbf{v} \in \mathbb{S}^2$, the color of G_i is

$$C_i(\mathbf{v}) = \sum_{\ell=0}^{2} \sum_{m=-\ell}^{\ell} \mathbf{c}_{i,\ell m}\, Y_\ell^m(\mathbf{v}), \tag{2}$$

where $\mathbf{c}_{i,\ell m} \in \mathbb{R}^3$ are learned RGB coefficients and Y_ℓ^m are real spherical harmonics.

Projection to the image plane uses the calibrated camera intrinsics together with the Gaussian projection model, and screen space compositing applies standard alpha compositing. All attributes are packed into contiguous GPU buffers and updated in place. This layout lets the renderer handle several million Gaussians without fragmentation and keeps memory access patterns coherent during both forward and backward passes.

4.2. Tracking and Camera Pose Optimization

Given the current map $\mathcal{G}$ and a new RGB or RGB-D frame, pose tracking alternates between rendering and gradient based refinement of the camera pose T_{WC}. We render a synthesized image $\hat{I}(x; \mathcal{G}, T_{WC})$ at the native resolution and apply an affine brightness model with gain g_I and bias b_I for each frame,

$$\tilde{I}_I(x; \mathcal{G}, T_{WC}) = g_I\, \hat{I}(x; \mathcal{G}, T_{WC}) + b_I. \tag{3}$$

The parameters (g_I, b_I) are estimated by a small least squares problem

$$(g_I, b_I) = \arg\min_{g,b} \sum_{x \in \Omega_I} \Big(I(x) - g\, \hat{I}(x; \mathcal{G}, T_{WC}) - b \Big)^2, \tag{4}$$

and the resulting solution is substituted into the photometric objective so that pose updates remain invariant to slow exposure drift.

The tracking loss can be written in the form

$$\mathcal{L}_{\text{track}}(T_{WC}) = \sum_{x \in \Omega_I} w_I(x)\, \big\| I(x) - \tilde{I}_I(x; \mathcal{G}, T_{WC}) \big\|_1, \tag{5}$$

where $w_I(x)$ denotes a per pixel weight. In practice this weight is factored into opacity and gradient terms,

$$w_I(x) = w_\alpha(x)\, w_\nabla(x), \tag{6}$$

with

$$w_\alpha(x) = \mathrm{clip}\left(\frac{\hat{\alpha}(x)}{\tau_\alpha}, 0, 1 \right), \tag{7}$$

$$w_\nabla(x) = \mathrm{clip}\left(\frac{\|\nabla I(x)\|_2}{\tau_\nabla}, 0, 1 \right), \tag{8}$$

where $\hat{\alpha}(x)$ is the accumulated opacity at pixel x, τ_α and τ_∇ are fixed thresholds, and $\mathrm{clip}(z, a, b) = \min(\max(z, a), b)$. Pixels with low opacity or weak gradients therefore have reduced influence in the optimization.

The pose is updated in the minimal twist coordinates $\xi \in \mathbb{R}^6$ using a standard left multiplicative update rule on $\mathrm{SE}(3)$. The Jacobians of the camera projection and the image formation model are implemented analytically and are reused across all Gaussians that share the same pose, which reduces both computation and memory traffic. The derivative of the projected covariance with respect to pose is evaluated by an explicit chain rule involving the projection Jacobian and the local rotation. This avoids generic automatic differentiation through the entire renderer.

In practice, between thirty and sixty gradient steps per frame are performed using the Adam optimizer with a cosine learning rate schedule centered around 5×10^{-3}. This configuration yields stable tracking even when large parts of the image are textureless or underexposed.

4.3. Keyframe Scheduling by Co-Visibility

Keyframe scheduling uses a co-visibility based policy. For each incoming frame we maintain a binary visibility mask

CVPR #9001

that records, for every pixel, whether its accumulated screen space opacity exceeds a small threshold. Let V_a and V_b denote the sets of visible pixels in images I_a and I_b. The covisibility score is defined as

$$\mathrm{IoU}(I_a, I_b) = \frac{|V_a \cap V_b|}{|V_a \cup V_b|}. \tag{9}$$

The intersection and union are counted entirely on the GPU.

A frame I_k is promoted to a keyframe when the covisibility $\mathrm{IoU}(I_k, I_{k^\star})$ with the last keyframe $I_{k^\star}$ falls below a user defined threshold τ and when the relative translation and rotation exceed small geometric bounds. These additional bounds avoid accepting nearly redundant viewpoints that would increase memory and computation without improving triangulation baselines.

Accepted keyframes are stored in a bounded buffer $\mathcal{B}$ together with their poses. Between eight and twelve recent keyframes are kept, which is sufficient to form well-conditioned local triangulation baselines while keeping all multi view operations inexpensive. The same buffer also provides the neighbour set used in the mapping stage.

4.4. Dense Feature Matching and Triangulation

Dense matching and multi view initialization rely on DINOv3 features computed at a fixed feature resolution obtained by downsampling the RGB images. The descriptors are ℓ_2 normalized per pixel. For each reference keyframe I_r a neighbour set $\mathcal{N}_r \subset \mathcal{B}$ is selected based on pose proximity and parallax, using the current camera estimates.

Let $f_r(p)$ and $f_n(q)$ denote DINOv3 descriptors at pixels p and q in images I_r and I_n. For each neighbour $n \in \mathcal{N}_r$ and displacement $\mathbf{u}_{r\to n}(p)$, the raw descriptor similarity is

$$s_{r\to n}(p) = \big\langle f_r(p), f_n\big(p + \mathbf{u}_{r\to n}(p)\big)\big\rangle. \tag{10}$$

This similarity is combined with forward backward consistency and epipolar agreement into a scalar score

$$\tilde{\kappa}_{r\to n}(p) = w_{\text{sim}}\, s_{r\to n}(p) + w_{\text{fb}}\, \rho_{\text{fb}}(p) + w_{\text{epi}}\, \rho_{\text{epi}}(p), \tag{11}$$

where ρ_{fb} and ρ_{epi} quantify consistency of the forward backward displacement and the epipolar distance, and w_{sim}, w_{fb}, and w_{epi} are fixed weights. The confidence $\kappa_{r\to n}(p)$ is obtained by a piecewise linear mapping to $[0, 1]$,

$$\kappa_{r\to n}(p) = \mathrm{clip}\left(\frac{\tilde{\kappa}_{r\to n}(p) - \gamma_0}{\gamma_1 - \gamma_0}, 0, 1\right), \tag{12}$$

with fixed thresholds γ_0 and γ_1. This design keeps the inlier classifier training-free and dataset agnostic.

Before triangulation, correspondences are thinned in image space with a blue noise pattern in order to avoid redundant seeds in locally homogeneous regions. The aggregated confidence $\bar{\kappa}(p)$ is cached for each surviving reference pixel and encodes agreement across multiple neighbours. Linear triangulation uses a homogeneous linear system solved with a double precision singular value decomposition. Candidates with very small parallax or a reprojection error above a conservative threshold are discarded, and among hypotheses obtained from different neighbours the one with the smallest reprojection error and a sufficiently large baseline angle is kept.

4.5. Gaussian Initialization and Joint Mapping

Each valid triangulated point spawns a Gaussian G_i whose world mean is set to the reconstructed point $\mu_i^W = \hat{X}(p)$. A local orthonormal frame $U_i = [\mathbf{t}_1, \mathbf{t}_2, \mathbf{v}]$ is constructed by fitting a plane to triangulated neighbours inside a fixed radius around $\hat{X}(p)$, where $\mathbf{v}$ is the normal and $\mathbf{t}_1, \mathbf{t}_2$ span the tangent plane in the neighbourhood. The covariance is initialized as an anisotropic ellipsoid aligned with this frame,

$$\Sigma_i^W = U_i\ \mathrm{diag}\big(s_\perp^2, s_\perp^2, s_\parallel^2\big)\, U_i^\top, \tag{13}$$

where the tangential scale $s_\perp$ is obtained by back projecting a one pixel footprint through the projection Jacobian at the reference view and the axial scale $s_\parallel$ is a calibrated function of the baseline angle and triangulation residual, which increases depth uncertainty in poorly conditioned configurations.

The initial color c_i is the median of bilinear RGB samples across all supporting views after applying the exposure parameters estimated during tracking. The initial opacity α_i is obtained as a monotone mapping of $\bar{\kappa}(p)$,

$$\alpha_i = \alpha_{\min} + \big(\alpha_{\max} - \alpha_{\min}\big)\, \bar{\kappa}(p), \tag{14}$$

with fixed bounds $\alpha_{\min}$ and $\alpha_{\max}$ in $(0, 1)$. Unreliable seeds with low aggregated confidence therefore enter the map as visually weak Gaussians and are easy to prune. Before insertion, Poisson disk subsampling in world space is applied to promote uniform coverage and to avoid excessive density in locally flat regions.

After insertion, the mapping module maintains an observation count m_i, a cumulative visibility score v_i, and an exponential moving average of the world position $\bar{\mu}_i^W$ for each Gaussian. The moving average is updated after each mapping step according to

$$\bar{\mu}_i^{W\,(t+1)} = (1 - \eta)\, \bar{\mu}_i^{W\,(t)} + \eta\, \mu_i^{W\,(t+1)}, \tag{15}$$

with a fixed smoothing factor $\eta \in (0, 1)$.

The mapping loss is evaluated over a sliding window $\mathcal{W}$ around the latest keyframe with per frame brightness parameters g_I and b_I and per pixel weights $w_I(x)$ as in Eq. (6). A regularizer discourages highly elongated covariances, avoids degenerate transmittance, and anchors early updates to $\bar{\mu}_i^W$. The coefficients of this regularizer are kept fixed across all sequences and are not adapted per scene,

which keeps the behaviour of the optimizer comparable on TUM and Replica.

Pose only updates and full map updates are alternated, and robust per pixel weights are used so that outliers in the photometric residual have limited influence. A lightweight merging operation averages color and covariance for pairs of Gaussians that have almost identical screen space footprints and similar appearance. Pruning removes Gaussians that violate bounds on m_i, v_i, $\mathrm{tr}(\Sigma_i^W)$, or α_i. These maintenance steps keep the representation compact and prevent numerical instabilities caused by extremely large or extremely transparent splats.

4.6. System Schedule and Computational Profile

The runtime schedule separates tracking and mapping. Each incoming frame is tracked for K_t gradient steps using the photometric loss described above. This stage updates only the current pose and does not modify the map. If the co-visibility test does not accept the frame as a keyframe, the system immediately proceeds to the next image.

Let ρ denote the empirical fraction of frames that are selected as keyframes. The average number of gradient based optimization steps per frame is then approximately

$$N_{\text{steps}}^{\text{frame}} \approx K_t + \rho\, K_m, \tag{16}$$

where K_m is the number of joint refinement iterations executed after each keyframe.

When a keyframe is accepted, a set of neighbours from $\mathcal{B}$ is selected and dense matching, confidence aggregation, triangulation, and Gaussian initialization are executed in a single GPU pass. The resulting Gaussians are inserted into $\mathcal{G}$ and immediately participate in mapping. The system then runs K_m iterations of joint refinement over the window $\mathcal{W}$ using the mapping loss and regularizer defined above, followed by a maintenance pass that performs merging and pruning.

This schedule replaces iterative residual-driven densification with a one-shot dense seed and thereby reduces the early drift of newly added parameters. The number of mapping iterations required to reach a given photometric fidelity decreases, which improves wall clock throughput while keeping the renderer and optimization objectives identical to those used in the main method description.

5. Additional Experiments

5.1. Additional Rendering on Replica

On the Replica dataset, extended qualitative comparisons in Figure 1 show that our keyframe-triggered single-step initialization yields sharper object boundaries and more stable shading than the residual-driven densification baseline across living room and office scenes. The reconstructed views exhibit fewer transparency artifacts around thin structures such as chair legs and table edges, and color transitions remain consistent across viewpoints, which confirms that the proposed initialization creates a well-conditioned Gaussian map for subsequent optimization. Challenging regions for Gaussian splatting, including large textureless walls and slanted ceilings, also show reduced blotchy artifacts because the one-shot dense seed avoids early gaps in coverage. These qualitative trends agree with the quantitative gains in reconstruction metrics reported in the main paper and indicate that the initializer improves both convergence speed and the final visual fidelity of the radiance field.

5.2. Camera Tracking on Replica Offices

To assess tracking robustness, we visualize top view camera trajectories for two Replica office scenes in Figure 2 and 3. The proposed system maintains tight alignment with ground truth over long paths that include turns, loops, and revisits, and the strong overlap between the predicted and reference trajectories indicates that the jointly optimized poses and Gaussians provide accurate geometric constraints for downstream mapping and loop closure. In the office0 sequence the path combines slow pans and rapid rotations around the desk area, yet our trajectory returns to previously visited regions without noticeable misalignment, while competitor methods accumulate drift near corners and walls. In the office2 sequence the camera passes through narrow corridors before entering a wider workspace, and the estimated path from our method preserves the global layout without evident shearing or scale distortion. These qualitative patterns are consistent with the Absolute Trajectory Error reported in the main paper and support the claim that dense Gaussian initialization yields a stable optimization landscape for pose refinement.

5.3. Cluttered Desk Reconstruction

In a cluttered desk sequence with strong self occlusion and fine scale objects such as cables, pencils, and plush toys, shown in Figure 4, we evaluate an additional reconstruction. The left image shows the input RGB view and the right image illustrates the corresponding Gaussian map rendered from a novel viewpoint. The reconstruction preserves thin structures and surface boundaries while avoiding the truncation and over smoothing artifacts observed in residual driven pipelines, which demonstrates that the proposed initialization and refinement strategy scales to scenes with complex object layouts and high frequency details. Fine elements such as tripod legs, monitor edges, and scattered stationery remain clearly separated from the background even when objects move partially in and out of view, so the map integrates evidence from multiple viewpoints without duplicated Gaussians.

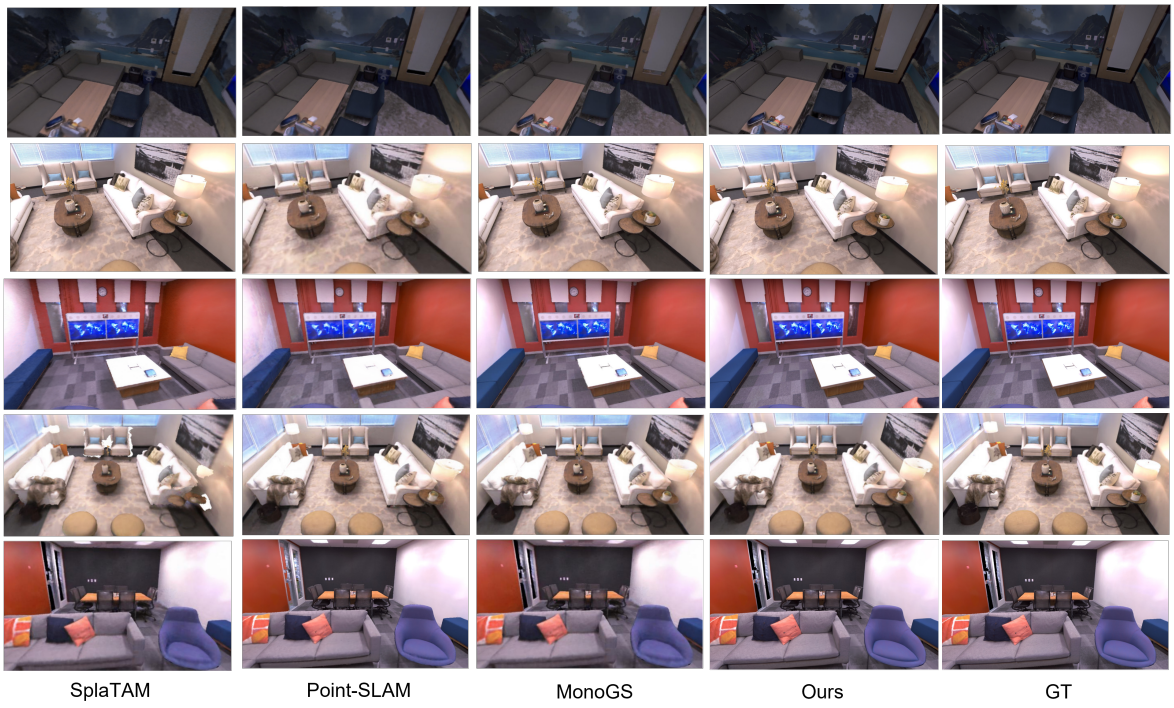

Figure 1. Rendering results on the Replica dataset. The proposed keyframe-triggered single-step initialization produces sharper edges, fewer transparency artifacts, and more consistent colors than residual-driven densification.

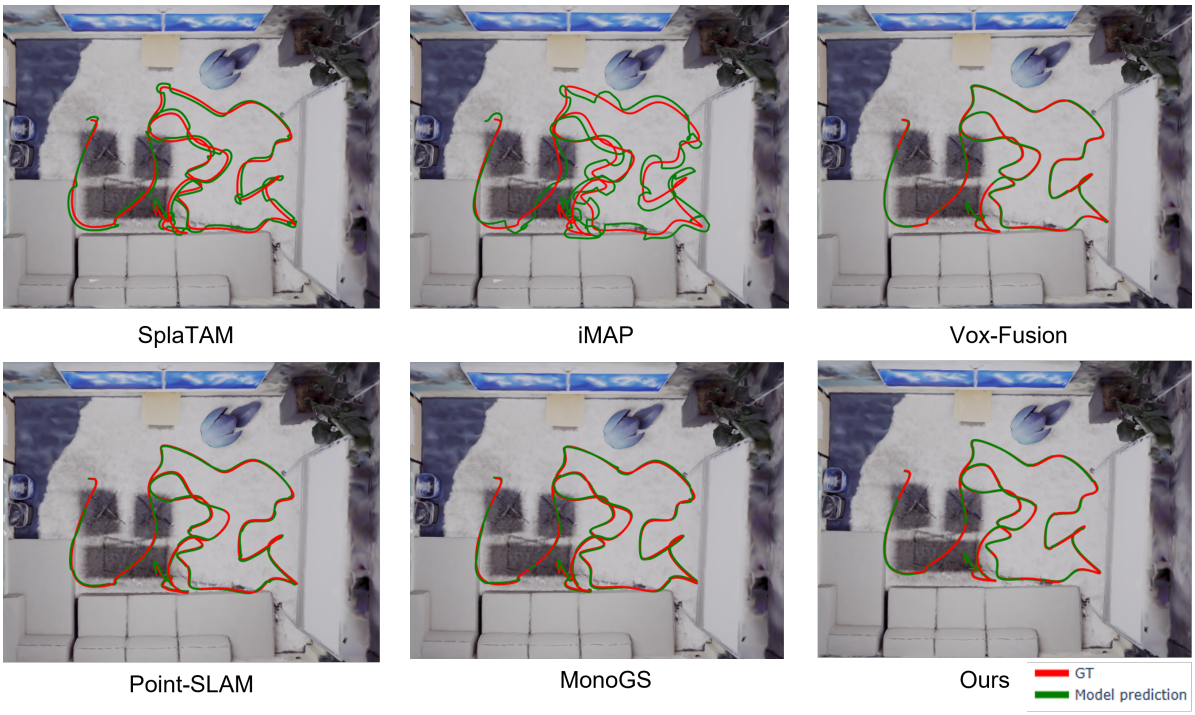

Figure 2. Tracking on Replica office0. Top-view trajectories, with ground truth in red and model predictions in green.

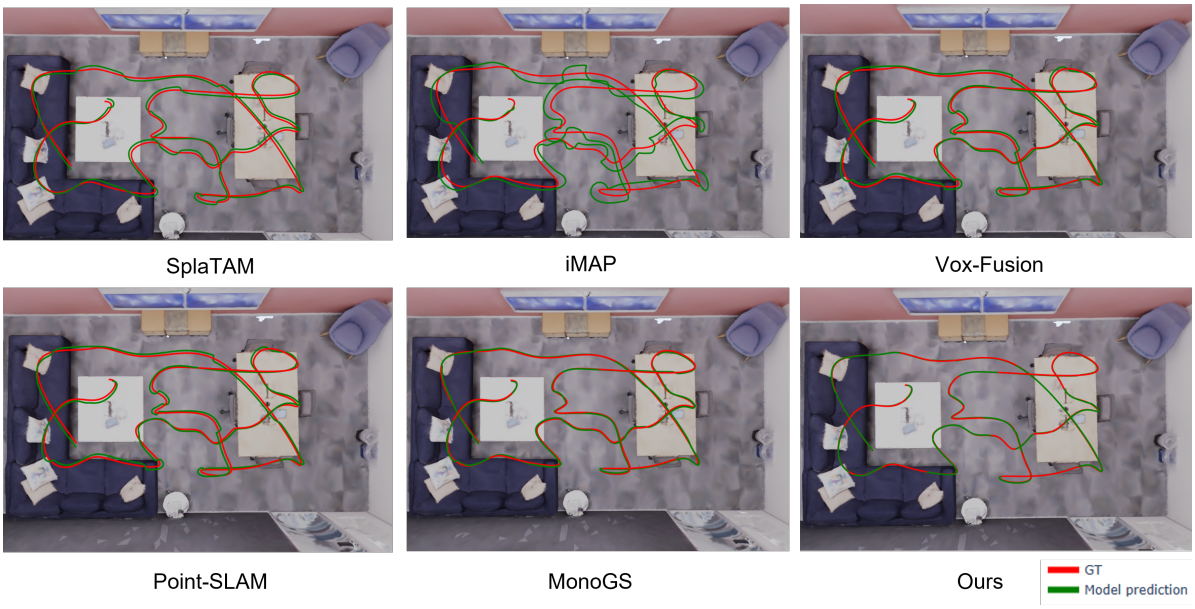

Figure 3. Tracking on Replica office2. Top-view trajectories, with ground truth in red and model predictions in green.

Figure 4. Qualitative reconstruction on a cluttered desk scene, where the left input RGB frame and the right Gaussian map view demonstrate dense coverage with preserved fine structures and reduced truncation artifacts.