

RoMo-GS: Efficient Online 3D Gaussian Splatting for Robot Mapping via Exposure-Integrated Motion Blur

Sangwoo Jung^{1,2}, Di Deng¹, Jiahui Fu¹, Laura Herlant^{1*}, Ayoung Kim^{2*}, Yafei Hu^{1*}

¹RAI Institute, United States

²Department of Mechanical Engineering, Seoul National University, South Korea

*Co-corresponding Authors

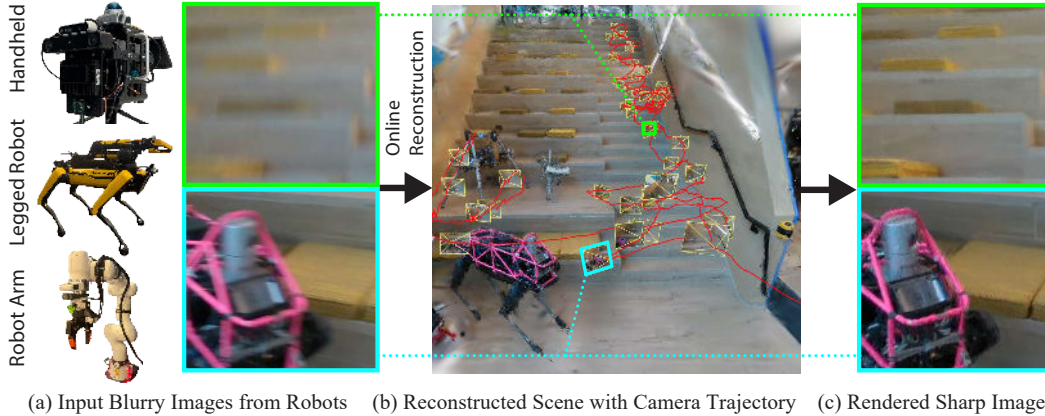


Figure 1: We introduce **RoMo-GS**, an efficient online 3D Gaussian Splatting system for real-world robotic mapping. Given a stream of motion-blurred frames captured by robots and front-end vision tracker outputs (a), RoMo-GS produces a compact 3DGS map (b) and sharp renderings online (c).

Abstract: Online 3D Gaussian Splatting (3DGS) is emerging as a practical way to build a representation for robotics applications. However, deploying it on real-world platforms exposes multiple challenges. Rapid maneuvers of a robot induce motion-blurred images, which corrupt the reconstructed geometry and cause camera pose drift. Furthermore, the number of Gaussians grows without bound during continuous online mapping. In this study, we present **RoMo-GS**, an online 3DGS system designed to address these challenges. Instead of averaging N discrete sub-frames as in prior motion blur-aware 3DGS, we analytically integrate each projected Gaussian over the exposure interval, yielding an $\mathcal{O}(1)$ complexity per Gaussian. Building on this, we bound the Gaussians’ growth with a Hessian-based pruning score, while jointly refining pose, velocity, and Gaussians with Adam and an online LM step over the Gaussians. On multiple public datasets and a self-collected dataset, RoMo-GS matches or exceeds offline deblurring 3DGS methods in rendering quality while running 15–150 $\times$ faster, and operates online at competitive quality with about 20% fewer Gaussians than the online 3DGS baseline.

Keywords: Online 3D Gaussian Splatting, Motion Deblurring, Online Density Control, Robotic Mapping

1 Introduction

Online dense 3D reconstruction is essential for robotics, enabling downstream tasks such as autonomous navigation and reliable interactions. While implicit representations such as Neural Radiance Field (NeRF) [1] have advanced novel-view synthesis, their slow rendering (~ 0.03 FPS) limits their online use on real-world robots. 3D Gaussian Splatting (3DGS) [2] addresses this by replacing the implicit field with explicit Gaussian primitives and a differentiable rasterizer, achieving real-time rendering (~ 100 FPS). Moving the rasterizer from controlled benchmarks onto actual robots, however, exposes the Gaussians to less controlled conditions. Rapid maneuvers and mechanical impacts

induce motion blur in the input frames, corrupting the reconstructed geometry and causing trajectory drift, while the Gaussians grow unbounded along the trajectory until memory exhaustion.

As 3DGS migrates to real-world robotic systems, contact impacts, vibration, and rapid maneuvers produce motion-blurred frames that corrupt the reconstructed geometry. Prior motion blur-aware 3DGS methods [3, 4, 5] mitigate this numerically, approximating the exposure integral by averaging N discrete subframes and incurring an $N \times$ training-cost overhead that is incompatible with online operation. We eliminate this trade-off by analytically evaluating the exposure-time integral of each projected Gaussian under a linear motion model, yielding $\mathcal{O}(1)$ complexity per Gaussian.

Beyond rendering, an online 3DGS system on a real-world robot faces remaining challenges. As the robot continuously explores the environment, new Gaussians are spawned with each incoming frame. Without a principled mechanism to prune the redundant or uninformative Gaussians, the number of Gaussians grows without bound and exhausts the robot’s memory budget. Naive pruning, on the other hand, degrades overall reconstruction quality. Furthermore, during rapid maneuvers, pose tracking may become unstable, introducing pose drift and corrupting structural integrity.

To address these constraints, we propose **RoMo-GS**, an online 3DGS pipeline designed for real-world robot deployment. To handle motion blur without multi-sampling, we analytically evaluate intra-exposure camera motion. To bound memory, we introduce an online Gaussian pruning score based on the opacity-axis diagonal of the Gauss-Newton Hessian. Finally, to resolve pose drift by tightening the tracking and mapping, we run a joint Adam loop over camera states and Gaussian parameters, followed by an online Levenberg-Marquardt (LM) step on the Gaussians. We evaluate RoMo-GS on multiple public datasets containing real-world and synthetic blurry scenes, as well as a custom multi-platform dataset. In summary, our contributions are as follows.

- **Exposure-Integrated Motion Blur.** We evaluate the exposure-time integral of each projected Gaussian’s motion in terms of the Error Function (ERF), replacing the discrete numerical approximations of prior approaches with our analytically evaluated formulation.
- **Single-Sample Blur Rasterization.** We integrate the proposed motion-blur model into the CUDA rasterizer to render motion-blurred frames in a single pass with $\mathcal{O}(1)$ complexity per Gaussian, removing the $N \times$ rasterization overhead of multi-sample averaging approaches.
- **Online Optimization and Density Control.** We jointly optimize pose, velocity, and Gaussians with Adam, followed by an online LM step on the Gaussians, and bound the map size with a Hessian-based pruning score that retains $\sim 20\%$ fewer Gaussians at comparable quality.

2 Related Work

2.1 Online Differentiable Rendering for Robotics

Driven by modern GPUs, neural rendering has emerged as essential for 3D reconstruction. NeRF [1] produces high-quality novel views via ray-tracing through an implicit Multi-Layer Perceptron (MLP), and subsequent works accelerate its training and rendering [6, 7, 8, 9]. 3DGS [2] achieves real-time rendering through an explicit representation and differentiable Gaussian rasterization.

Leveraging the speed of 3DGS, recent work has extended to online reconstruction. With RGB-D input, MonoGS [10] and SplatAM [11] couple 3DGS with pose tracking; RTG-SLAM [12] accelerates optimization through a compact representation. GS-ICP SLAM [13] fuses Generalized Iterative Closest Point (GICP) to enhance tracking speed and mapping quality, and Photo-SLAM [14] adopts Gaussian-Pyramid learning to accelerate optimization. With RGB input, HI-SLAM2 [15] integrates learning-based dense simultaneous localization and mapping (SLAM) for improved geometry, and MODP [16] presents hierarchical Gaussian management and global consistency optimization.

Still, deploying these systems on real robots breaks the assumption of high-quality inputs: vibration, contact impacts, and rapid maneuvers produce motion-blurred frames that degrade the reconstruction and drive pose drift, while unbounded Gaussian growth during online mapping inflates memory.

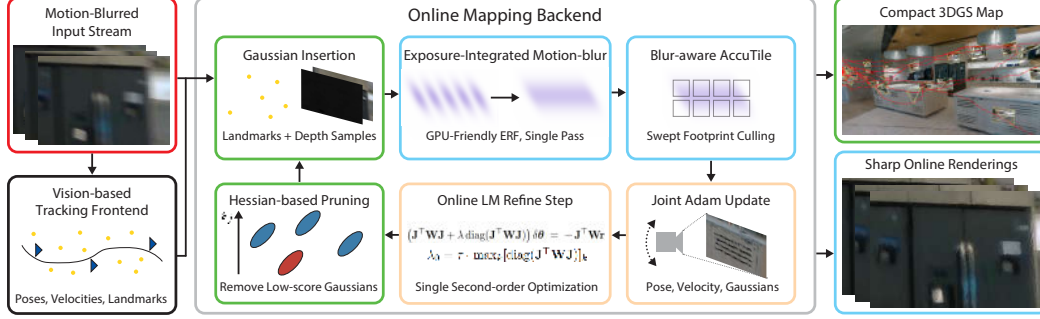


Figure 2: Overview of **RoMo-GS**. Given inputs from the vision-based tracker, we spawn initial Gaussians with the landmarks and depth samples. The exposure-integrated motion blur model renders a blurry image in a single rasterization. A joint Adam loop and an LM step refine the camera state and Gaussians, while Hessian-based score prunes redundant Gaussians. The pipeline produces a compact 3DGS map and sharp online renderings.

2.2 Deblurring Neural Rendering

Prior works have explored deblurring strategies within the neural rendering framework to mitigate motion blur. Early approaches, such as Deblur-NeRF [17], adopt 2D blind-deblurring with a learned blur kernel. Deblurring 3DGS [18] and BAGS [19] extend the kernel-based formulation with MLP- and convolutional neural network (CNN)-based kernels. Unblur-SLAM [20] combines a feed-forward deblurring network with multi-view kernel learning. Still, kernel-based approaches treat blur as an image-space artifact and lack a physical interpretation of the camera’s kinematics.

To preserve geometric consistency, methods such as DP-NeRF [21] model intra-exposure camera motion as a rigid-body trajectory. Building on this, BAD-NeRF [22] and BAD-Gaussians [3] model camera trajectories by interpolating with spline-based methods. Gaussian Splatting on the Move (GSOM) [4] derives a pixel-velocity motion blur model under a screen-space approximation, while CoMoGaussian [5] extends kinematic deblurring to continuous motion by leveraging neural Ordinary Differential Equations (ODEs). These kinematic motion-based approaches simulate blur by rendering and averaging N sub-frames along the trajectory and scale the training cost as $\mathcal{O}(N)$, prohibitive for online robot deployment. Instead, we leverage the explicit representation of 3DGS to analytically compute motion blur in a single rasterization pass.

2.3 Recent Advances in Enhancing 3DGS

To achieve fast convergence and compact scene representation in 3DGS, recent work has refined optimization and pruning. 3DGS-LM [23] appends second-order LM steps after the Adam loop to accelerate convergence, while Speedy-Splat [24] and FastGS [25] compact the Gaussians via offline score-based pruning and tighten per-Gaussian pixel coverage. Although effective offline, they rely on access to the converged scene or a full backward pass over all training views, so directly applying them to online 3DGS introduces additional compute overhead and degrades render quality.

A complementary line of work [26, 27] explored feed-forward Gaussian reconstruction, leveraging recent 3D vision foundation models [26, 28]. They exploit robust depth priors to suppress the floater artifacts that plague optimization-based pipelines. Still, their fixed Gaussian count removes the adaptive density control, which limits capturing high-frequency textures. We adopt the score-based pruning and the depth-prior bootstrapping, reformulating both to operate within an online system.

3 Methods

Our system is tracker-agnostic and compatible with any visual SLAM, including OKVIS2-X [29]. We adopt ORB-SLAM3 [30] as the front-end tracker and build on MODP [16], an online monocular 3DGS system. For optimization, standard 3DGS loss $\mathcal{L} = (1 - \lambda_{\text{ssim}})\mathcal{L}_1(\hat{I}, I) + \lambda_{\text{ssim}}(1 - \text{SSIM}(\hat{I}, I))$ with $\lambda_{\text{ssim}} = 0.2$ is used. An overview of our method is shown in Fig. 2.

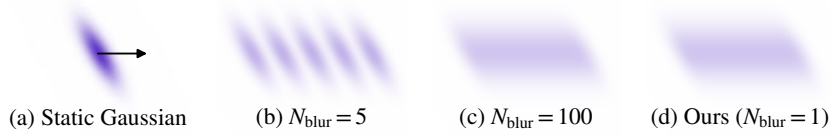


Figure 3: Comparison with multi-sample. (a) Static Gaussian with a motion arrow. (b) $N_{\text{blur}} = 5$, visible discretization. (c) $N_{\text{blur}} = 100$, smoother, but expensive. (d) Our model results same as (c) at a single sample.

3.1 Analytic Motion Blur Modeling for 3DGS

Image under Camera Motion. The observed image of frame i captured by a moving camera integrates the rasterized color $\mathbf{C}_i(\mathbf{x}, \mathbf{T}(t_i))$ evaluated at pixel $\mathbf{x}$ under pose $\mathbf{T}(t) \in \text{SE}(3)$:

$$\bar{\mathbf{C}}_i(\mathbf{x}) = \frac{1}{t_e} \int_{-t_e/2}^{t_e/2} \mathbf{C}_i(\mathbf{x}, \mathbf{T}(t_i + \tau)) d\tau, \quad (1)$$

with exposure t_e . Assuming constant linear and angular velocities $(\mathbf{v}_i, \boldsymbol{\omega}_i)$ during the exposure, $\mathbf{T}(t_i + \tau)$ admits the standard SE(3) linearization $[\mathbf{R}_i \exp(\tau[\boldsymbol{\omega}_i]_{\times}) \mid \mathbf{p}_i + \mathbf{R}_i \mathbf{v}_i \tau]$ around $(\mathbf{R}_i, \mathbf{p}_i)$. Still, direct rasterization of Eq. (1) is infeasible as the integral has no closed-form solution.

Screen-Space Approximation. Standard 3DGS represents the scene with 3D Gaussians $\{\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j, o_j, \mathbf{c}_j\}$, projected to screen-space $(\boldsymbol{\mu}'_{i,j}, \boldsymbol{\Sigma}'_{i,j})$ [31]. GSOM [4] restricts the camera motion to the projected means so that each $\boldsymbol{\mu}'_{i,j}$ moves linearly during the exposure with pixel velocity $\mathbf{v}'_{i,j} = -J_i(\boldsymbol{\omega}_i \times \hat{\boldsymbol{\mu}}_{i,j} + \mathbf{v}_i)$, where $\hat{\boldsymbol{\mu}}_{i,j} = \mathbf{R}_i^\top(\boldsymbol{\mu}_j - \mathbf{p}_i)$ and J_i is the affine projection Jacobian at $\hat{\boldsymbol{\mu}}_{i,j}$. Under this, GSOM averages the standard 3DGS $\alpha_{i,j}(\mathbf{x}, \boldsymbol{\mu}') = o_j \exp(-\frac{1}{2}(\boldsymbol{\mu}' - \mathbf{x})^\top (\boldsymbol{\Sigma}'_{i,j})^{-1}(\boldsymbol{\mu}' - \mathbf{x}))$ over N_{blur} uniform trajectory samples $\tau_k = ((k-1)/(N_{\text{blur}}-1) - 1/2)t_e$:

$$\bar{\mathbf{C}}_i(\mathbf{x}) = \frac{1}{N_{\text{blur}}} \sum_{k=1}^{N_{\text{blur}}} \sum_j \mathcal{T}_j^{(k)} \mathbf{c}_{i,j} \alpha_{i,j}(\mathbf{x}, \tilde{\boldsymbol{\mu}}'_{i,j}(t_i + \tau_k)), \quad \tilde{\boldsymbol{\mu}}'_{i,j}(t_i + \tau) = \boldsymbol{\mu}'_{i,j}(t_i) + \mathbf{v}'_{i,j} \tau, \quad (2)$$

where $\mathcal{T}_j^{(k)} = \prod_{l < j} (1 - \alpha_{i,l}(\mathbf{x}, \tilde{\boldsymbol{\mu}}'_{i,l}(t_i + \tau_k)))$ is the front-to-back transmittance at sample k . Still, the rasterization scales as $\mathcal{O}(N_{\text{blur}})$ [4], resulting in a proportional increase in running time.

Exposure-Integrated Motion Blur. We retain the screen-space approximation but analytically evaluate the exposure-time integral using the GPU-friendly ERF, replacing the discrete N_{blur} -sample average. Letting $\boldsymbol{\delta} = \boldsymbol{\mu}'_{i,j} - \mathbf{x}$, the time-averaged opacity contribution of Gaussian j at pixel $\mathbf{x}$ is

$$\bar{\alpha}_{i,j}(\mathbf{x}) = \frac{o_j}{t_e} \int_{-t_e/2}^{t_e/2} \exp\left(-\frac{1}{2}(\boldsymbol{\delta} + \mathbf{v}'_{i,j} \tau)^\top (\boldsymbol{\Sigma}'_{i,j})^{-1}(\boldsymbol{\delta} + \mathbf{v}'_{i,j} \tau)\right) d\tau. \quad (3)$$

With $a = \mathbf{v}'_{i,j}^\top \boldsymbol{\Sigma}'_{i,j}^{-1} \mathbf{v}'_{i,j}$, $b = \mathbf{v}'_{i,j}^\top \boldsymbol{\Sigma}'_{i,j}^{-1} \boldsymbol{\delta}$, $c = \boldsymbol{\delta}^\top \boldsymbol{\Sigma}'_{i,j}^{-1} \boldsymbol{\delta}$, $\tau^* = -b/a$, and $u = \sqrt{a/2}(\tau - \tau^*)$, Eq. (3) reduces to an ERF difference-based exposure-integrated representation as

$$\bar{\alpha}_{i,j}(\mathbf{x}) = \frac{o_j}{t_e} \sqrt{\frac{\pi}{2a}} \exp\left(-\frac{c - b^2/a}{2}\right) [\text{erf}(u_+) - \text{erf}(u_-)], \quad u_{\pm} = \sqrt{\frac{a}{2}}\left(\pm \frac{t_e}{2} - \tau^*\right), \quad (4)$$

with the detailed derivation can be found in App. §A. The blurred image is then composited front-to-back with $\bar{\alpha}_{i,j}$ replacing the static $\alpha_{i,j}$, collapsing Eq. (2) into a single rasterization pass. Although the ERF is a non-elementary integral, it is available as a hardware intrinsic on modern GPUs and evaluates with $\mathcal{O}(1)$ complexity per Gaussian. Fig. 3 compares the proposed motion-blur model with the multi-sample approach, while numerical stability around zero velocity is discussed in App. §B.

3.2 Optimization

Pose and Velocity Optimization. We adopt the pose correction of MODP [16] and the velocity correction of GSOM [4]: active keyframe i receives $\hat{\mathbf{T}}_i = [\text{Exp}(\delta \mathbf{q}_i) \mathbf{R}_i \mid \mathbf{p}_i + \delta \mathbf{p}_i]$, $\hat{\mathbf{v}}_i = \mathbf{v}_i + \Delta \mathbf{v}_i$, $\hat{\boldsymbol{\omega}}_i = \boldsymbol{\omega}_i + \Delta \boldsymbol{\omega}_i$ where $(\delta \mathbf{p}_i, \delta \mathbf{q}_i, \Delta \mathbf{v}_i, \Delta \boldsymbol{\omega}_i)$ are trained jointly with the Gaussians by Adam.

LM Refinement with Online 3DGS. After the Adam loop, we apply an LM step on the Gaussian parameters with the camera frozen, following 3DGS-LM [23]. We solve a Huber-weighted, diagonally damped Gauss-Newton normal equation by preconditioned conjugate gradient (App. §C). Since LM is invoked on a freshly inserted keyframe rather than a converged scene, λ_0 is set following Nielsen’s data-driven approach [32], proportional to the dominant entry of $\text{diag}(\mathbf{J}^\top \mathbf{W} \mathbf{J})$.

Blur-Aware AccuTile Rasterization. We replace the AABB tile assignment of standard 3DGS with the AccuTile from Speedy-Splat [24], sweeping the screen-space ellipse row-by-row to recover exact tile ranges. To bound the time-extended representation of Eq. (4), we union the per-row range over three samples $\tau \in \{-t_e/2, 0, t_e/2\}$ along the $\mathbf{v}'_{i,j}$, covering the swept hull of the Gaussian.

3.3 Pruning and Densification

MODP [16] prunes Gaussians by opacity and densifies via learned monocular depth [33]; we retain this design and add a Hessian-based pruning score and a depth-based Gaussian initialization.

Hessian-based Online Gaussian Pruning. Inspired by the Hessian-based importance score of Speedy-Splat [24], we score each Gaussian by the per-pixel squared sensitivity of the photometric loss L to its opacity o_j as $s_j = \sum_i \sum_{\mathbf{x}} (\partial L_i / \partial o_j)^2$, which is the opacity-axis Gauss-Newton diagonal of the training loss and is acquired from the gsplat [34] backward at no extra cost (App. §D). Gaussians whose loss-weighted effect covers few pixels score low and are pruned, while those observed in more than 10 keyframes are eligible.

Depth-based Additional Gaussian Initialization. Beyond the ORB-SLAM3 landmarks, we sample extra points from the depth image at pixels in high-frequency regions that are poorly explained by the current render, with probability $P(\mathbf{x}) = \text{clip}(\|(\nabla^2 G_\sigma) * I_i\|, 0, 1) \cdot \text{SSIM}(\hat{I}_i, I_i)(\mathbf{x})$. The first factor targets high-frequency regions, while the second down-weights well-optimized pixels.

4 Experiment

4.1 Experimental Setup

For evaluation, we use three public datasets: the Deblur-NeRF [17] synthetic dataset, ETH3D [35], and TUM RGB-D [36]. To validate our framework with diverse real-world motion blur induced by different robots, we captured custom indoor sequences using a Boston Dynamics Spot quadrupedal robot, a Franka Emika robot arm, and a handheld platform. We report peak signal-to-noise ratio (PSNR), structural similarity index measure (SSIM), and learned perceptual image patch similarity (LPIPS) scores, running speed, and the number of Gaussians. All experiments are conducted on a single NVIDIA RTX A6000 GPU.

4.2 Deblur 3DGS Comparison

We compare our method against existing offline global-optimization motion-blur-aware 3DGS baselines: BAD-Gaussians [3], Deblurring 3DGS [18], GSOM [4], BAGS [19], and CoMoGaussian [5]. The evaluation uses the Deblur-NeRF [17] synthetic dataset and the camera-shaking sequences of ETH3D [35]. We omit the Deblur-NeRF real-world sequences as ORB-SLAM3 fails on them. Hessian-based pruning is disabled here so that the comparison focuses on the deblurring quality. Quantitative results appear in Table 7 and Table 2; qualitative comparisons are shown in Fig. 4.

On the Deblur-NeRF synthetic dataset, BAD-Gaussians attains the best overall metrics. It is the only method that performs photometric bundle adjustment, allowing it to correct the noisy ORB-SLAM3 trajectory through global optimization. We therefore treat it as an indirect upper-bound reference and primarily compare against the remaining methods, which share the same front-end poses. Among those, our framework reaches the highest PSNR and SSIM, demonstrating robust deblurring even under severe motion blur and inaccurate initial poses. Our method also completes

Table 1: Deblur-NeRF Synthetic Dataset Experiment Results

	BAD-Gaussians	Deblurring 3DGS	GSOM	BAGS	CoMoGaussian	Ours
PSNR $\uparrow$	28.367	24.174	22.895	24.606	23.650	25.148
SSIM $\uparrow$	0.903	0.768	0.775	0.767	0.748	0.799
LPIPS $\downarrow$	0.055	0.097	0.274	0.090	0.102	0.179
Train Time (sec)	2073.03	623.92	1431.63	1938.75	6203.25	41.68
# of Gaussians	261,602	168,837	121,553	141,271	55,845	75,813

Table 2: ETH3D Real World Dataset Experiment Results

	Deblurring 3DGS	GSOM	BAGS	CoMoGaussian	Ours
PSNR $\uparrow$	14.278	25.879	19.414	19.764	25.367
SSIM $\uparrow$	0.545	0.881	0.842	0.842	0.905
LPIPS $\downarrow$	0.430	0.186	0.202	0.147	0.113
Train Time (sec)	685.73	1648.60	2922.50	7518.50	117.74
# of Gaussians	141,926	112,874	153,041	54,777	117,669

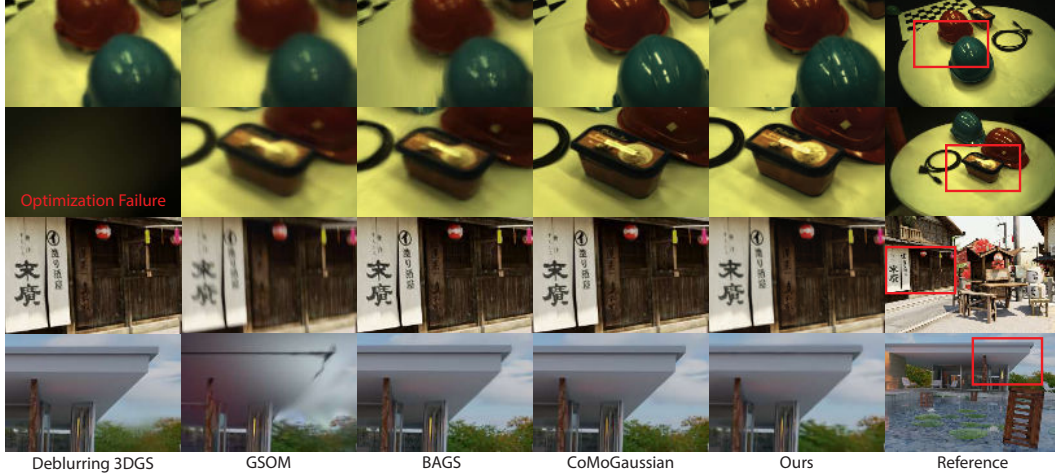


Figure 4: Top: ETH3D real-world. Bottom: Deblur-NeRF synthetic. GSOM fails to handle motion blur on both. Deblurring 3DGS and BAGS handle the synthetic case but collapse on ETH3D. CoMoGaussian is competitive on both, while Ours matches CoMoGaussian’s visual quality and trains 70–150 $\times$ faster online.

running in under one minute, against 10–60 min for the offline baselines. LPIPS is slightly higher, reflecting a marginal loss of perceptual sharpness which is expected from online, sequential optimization compared with offline, global refinement.

For ETH3D, clean ground-truth is unavailable. We therefore use the outputs of BAD-Gaussians (~ 4 hours of offline optimization) as a reference, while noting that this introduces a bias toward multi-sample rendering. Due to the aggressive camera motion, Deblurring 3DGS, GSOM, and BAGS fail to handle motion blur in Fig. 4. Deblurring 3DGS even fails to optimize on the second sequence, presenting a limitation of the MLP kernel. Ours attains the best SSIM and LPIPS among the baselines and a PSNR comparable to GSOM, with the fastest optimization. Achieving the best LPIPS on these challenging real-world sequences indicates that our motion-blur model avoids the discretization artifacts of multi-sample and remains robust under real-world motion. In Fig. 4, Ours produces fidelity comparable to CoMoGaussian, whereas the remaining baselines yield blurry renderings.

4.3 Online 3DGS Comparison

We compare our method with existing online 3DGS including RTG-SLAM [12], GS-ICP [13], MonoGS [10], Photo-SLAM [14], HI-SLAM2 [15], and MODP [16]. Additionally, we include tttLRM [27], a recent feed-forward Gaussian reconstruction method, for a comprehensive comparison. We evaluate on the TUM [36] real-world dataset and our custom dataset. The deblur model is disabled in this experiment, as clean ground-truth is unavailable, and the baselines do not target motion-blurred input. Therefore, we leveraged the original input RGB images as the ground-truth reference, as in previous online Gaussian Splatting methods. Quantitative results are reported in Table 3 and Table 4, and qualitative results on the custom dataset are shown in Fig. 5. Details about the custom dataset are provided in App. §E.

Table 3: TUM Real World Dataset Experiment Results

	RTG-SLAM	GS-ICP	MonoGS	Photo-SLAM	HI-SLAM2	tttLRM	MODP	Ours
PSNR $\uparrow$	16.794	20.600	24.247	17.187	21.811	17.954	<u>25.267</u>	25.284
SSIM $\uparrow$	0.617	0.768	0.803	0.652	0.784	0.714	0.842	0.840
LPIPS $\downarrow$	0.461	0.220	0.227	0.334	0.200	0.363	0.187	0.194
Running Speed (fps)	7.540	<u>30.143</u>	2.420	46.585	4.827	12.239	11.335	11.044
# of Gaussians	303,425	1,584,402	37,811	66,395	93,609	3,726,362	210,275	167,234

Table 4: Custom Real World Dataset Experiment Results

	RTG-SLAM	GS-ICP	MonoGS	Photo-SLAM	HI-SLAM2	tttLRM	MODP	Ours
PSNR $\uparrow$	20.460	18.503	23.387	21.254	23.640	21.316	26.994	26.822
SSIM $\uparrow$	0.718	0.740	0.793	0.757	0.832	0.748	0.871	0.867
LPIPS $\downarrow$	0.512	0.380	0.398	0.441	0.276	0.382	0.202	0.213
Running Speed (fps)	7.275	<u>29.081</u>	1.424	53.744	4.066	13.822	9.091	9.150
# of Gaussians	405,179	3,279,379	81,868	268,928	117,082	3,725,979	385,557	309,111

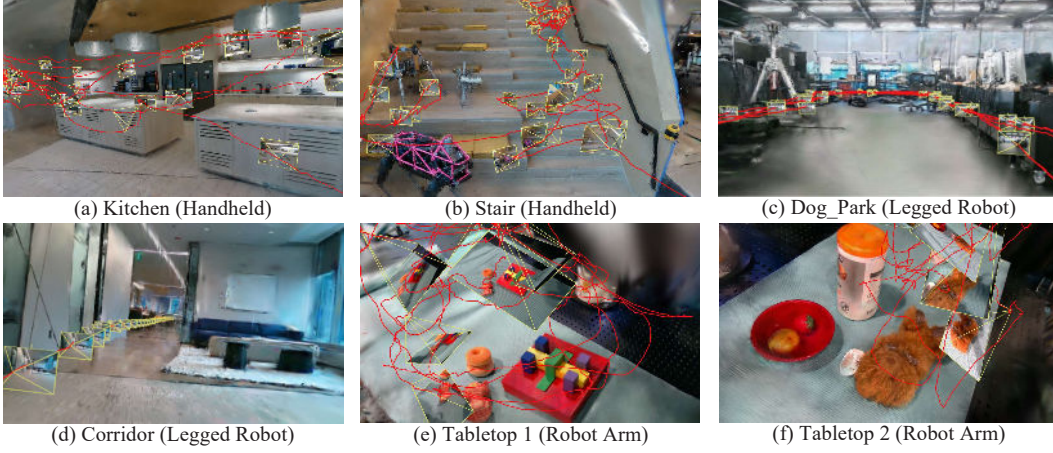


Figure 5: Scene reconstruction result of RoMo-GS on custom dataset. The red line shows the robot trajectory, including jittering due to the system’s physical interaction with the environment, and the yellow squares are the camera frustums.

On both TUM and our custom datasets, the rendering quality of GS-ICP and Photo-SLAM remains degraded, a trade-off for their fast optimization speed. tttLRM and GS-ICP show a higher number of Gaussians than the others, indicating limited online density control. MonoGS and HI-SLAM2 attain competitive metrics with a small number of Gaussians, but at the cost of the slowest running speed and a long post-optimization stage. Ours achieves comparable quality with MODP, while retaining $\sim 20\%$ fewer Gaussians, which follows from the Hessian-based online pruning.

4.4 Ablation Studies

4.4.1 Comparison with Multi-Sample Rendering

To isolate the contribution of our exposure-integrated blur formulation, we compare it against the discrete multi-sample strategy of GSOM [4] and a no-blur baseline, all within the same framework. We evaluate on the ETH3D dataset with multi-sample variants at $N_{\text{blur}} = 5$ and $N_{\text{blur}} = 10$. Table 5 reports the quantitative results and Fig. 6 shows the qualitative comparison.

Rendering Quality. The w/o blur reduction baseline yields the worst scores across all metrics, and multi-sampling improves them. Our exposure-integrated blur is slightly behind multi-sampling on PSNR and SSIM but beats the $N_{\text{blur}} = 5$ variant on LPIPS. This perceptual gain arises because the pseudo ground truth is generated by BAD-Gaussians, which itself uses discrete multi-sampling and therefore biases the evaluation. In Fig. 6, our model produces cleaner reconstructions than the multi-sample variants while requiring fewer Gaussians to represent the scene; the multi-sample variants spawn additional Gaussians to compensate for the temporal gaps caused by discrete rasterization.

Table 5: Comparison with Multi-Sample Rendering on ETH3D Dataset

	Ours (Exposure-Integrated)	w/o Blur Reduction	Multi Sample ($N_{\text{blur}} = 5$)	Multi Sample ($N_{\text{blur}} = 10$)
PSNR $\uparrow$	25.367	24.355	25.951	26.003
SSIM $\uparrow$	0.905	0.892	0.911	0.911
LPIPS $\downarrow$	0.113	0.202	0.121	0.107
Running Speed (fps)	3.695	5.698	2.448	1.592
# of Gaussians	117,669	129,128	124,831	126,902

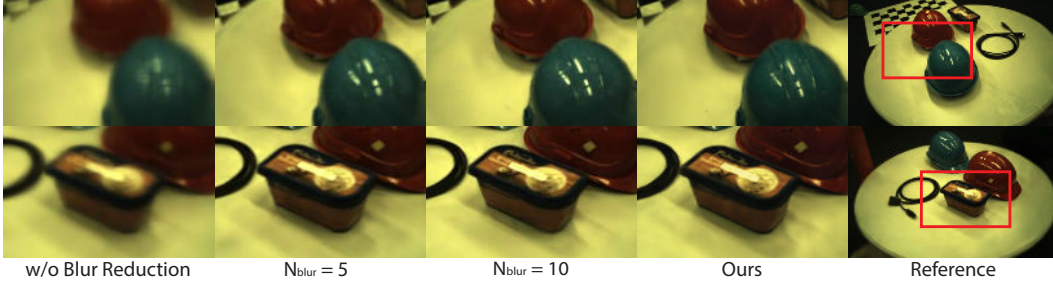
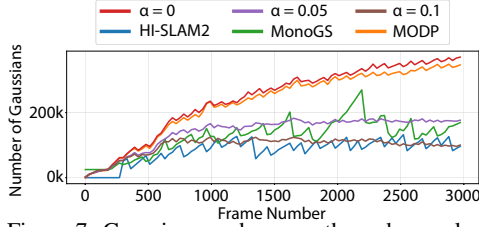
Figure 6: Comparison of exposure-integrated blur with multi-sample blur. $N_{\text{blur}} = 5$ partially reduces motion blur. Compared with $N_{\text{blur}} = 10$, our model provides a competitive result with $2\times$ running speed.

Figure 7: Gaussian number growth on dog_park sequence. The number of Gaussians converges with Hessian-based Online Gaussian Pruning, while the Gaussian number of others grows unbounded or fluctuates unstably.

Table 6: Online Pruning Ratio Comparison

	# of Gaussians	PSNR $\uparrow$	SSIM $\uparrow$	LPIPS $\downarrow$
$\alpha = 0$	374,989	27.803	0.917	0.158
$\alpha = 0.05$	170,913	27.384	0.912	0.170
$\alpha = 0.1$	97,962	26.820	0.903	0.185
MODP	349,984	27.891	0.919	0.154
HI-SLAM2	102,411	24.365	0.886	0.189
MonoGS	148,612	24.695	0.867	0.221

Running Speed. The no-blur baseline is the fastest, while our model incurs an overhead from evaluating the two ERF terms of Eq. (4) during the rasterization pass. Unlike multi-sampling whose cost scales as $\mathcal{O}(N_{\text{blur}})$, ours runs in $\mathcal{O}(1)$ regardless of N_{blur} , keeping it viable for online framework.

4.4.2 Effect of Hessian-based Online Gaussian Pruning

We ablate the Hessian-based pruning ratio $\alpha \in \{0, 0.05, 0.1\}$ on the dog_park custom sequence (3000 frames, 4 loops). Fig. 7 plots the Gaussian number over frame index and Table 6 reports the final counts. For $\alpha = 0$ and MODP, the Gaussian number grows without bound, while at $\alpha = 0.05$, the count plateaus near frame 1000, around the first loop closure. This confirms that our score retains the Gaussians anchored to each keyframe and drops the redundant ones. For HI-SLAM2 and MonoGS, the Gaussian number fluctuates due to the absence of robust online density control.

We also run an experiment with a controlled Gaussian count of $\sim 150\text{K}$ for RoMo-GS, compared to Deblurring 3DGS ($\sim 169\text{K}$). As shown in Table 7, increasing the Gaussian count consistently improves the quantitative metrics, while LPIPS remains higher than Deblurring 3DGS. We attribute this gap to per-Gaussian time averaging before alpha compositing, which may slightly smooth object boundaries and thus penalize LPIPS.

Table 7: Experiment with $\sim 150\text{K}$ Gaussians on the Deblur-NeRF Synthetic Dataset

	PSNR	SSIM	LPIPS	# of Gaussians
Ours	25.148	0.799	0.179	75,813
Ours (150K Gaussians)	25.499	0.815	0.164	150,164
Deblurring 3DGS	24.174	0.768	0.097	168,837

4.4.3 Additional Comparison with MODP on Blurry Scene

On Table 3 and Table 4, RoMo-GS matches MODP while using ~ 20 fewer Gaussians. To show clear qualitative improvements of RoMo-GS on severe motion blur, we report an additional experiment on the ETH3D dataset. As shown by the comparison at Table 8 and Fig. 8, RoMo-GS is particularly beneficial for online mapping under rapid camera motion.

Table 8: Comparison with MODP on the camera_shake_1 sequence of ETH3D

	PSNR	SSIM	LPIPS	# of Gaussians
Ours	25.648	0.908	0.113	106,285
MODP	23.726	0.879	0.244	129,165

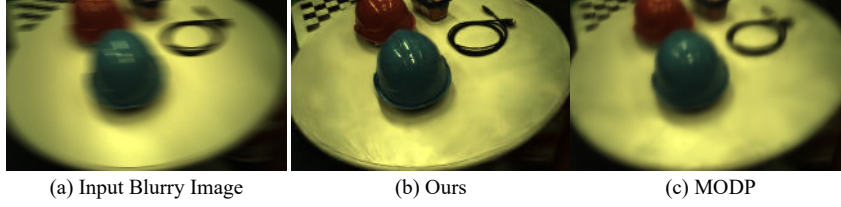


Figure 8: RoMo-GS and MODP on the blurry input (camera_shake_1 seq of ETH3D).

4.4.4 Complete System Ablation Study

We additionally ablate deblurring, pruning, joint pose/velocity, and the online LM step to evaluate the complete RoMo-GS pipeline on the ETH3D dataset as in Table 9. The evaluation metrics get worse when we remove each module of RoMo-GS, except for Hessian-based pruning.

Table 9: Complete system ablation study of RoMo-GS on the camera_shake_1 sequence of ETH3D

Pruning	Deblurring	LM	Pose/velocity	PSNR	SSIM	LPIPS	# of Gaussians
✓	✓	✓	✓	24.145	0.876	0.232	62,582
✗	✓	✓	✓	25.648	0.908	0.113	106,285
✗	✗	✓	✓	23.485	0.876	0.247	139,051
✗	✗	✗	✓	23.457	0.876	0.250	131,815
✗	✗	✗	✗	23.059	0.872	0.251	122,100

4.4.5 Tracking Accuracy

We additionally evaluate the tracking accuracy on the TUM [36] dataset. Specifically, we compare the root mean square error (RMSE) of the Absolute Trajectory Error (ATE) from the Gaussian-jointly-optimized poses against native ORB-SLAM3 [30], which serves as the input pose stream to our mapper. On average, ATE_t is reduced by about **2%** and ATE_r by about **0.4%**. While the improvement is modest, it confirms that jointly optimizing the camera state with the Gaussian map not only sharpens rendering but also provides a marginal refinement of the front-end pose estimates.

5 Conclusion

We present **RoMo-GS**, an efficient online 3DGS system designed for robotic mapping. Replacing the N -sample averaging of prior offline global motion-blur-aware 3DGS methods, we analytically integrate each projected Gaussian’s contribution over the exposure window into a single ERF evaluation, running with $\mathcal{O}(1)$ complexity per Gaussian. Around this kernel, we jointly refine pose, velocity, and Gaussians with an Adam loop followed by an online LM step, and bound the map size by online density control driven by a Hessian-based score. On the Deblur-NeRF synthetic dataset and the ETH3D real-world dataset, RoMo-GS matches or exceeds offline comparisons while running 15–150 $\times$ faster and remaining online; on TUM and our self-collected multi-platform dataset, RoMo-GS remains competitive with $\sim 20\%$ fewer Gaussians than the online baseline MODP [16].

6 Limitation

Our derivation assumes a linear motion model, which cannot capture curved intra-exposure trajectories, so extending with $\mathfrak{se}(3)$ Lie algebra is required. The proposed motion blur model also requires the exposure time t_e as a prior; jointly optimizing t_e or reading it from camera metadata would address this. Finally, the front-end tracker constrains mapping quality, which can lead to back-end reconstruction failure when the front-end vision tracker fails. Tighter coupling between the 3DGS map and the tracker may provide more accurate results and prevent front-end tracker failure.

ACKNOWLEDGMENT

The authors thank and acknowledge ETH Zürich for allowing us to use their experimental space for real-world testing, data collection, and demo video filming.

References

- [1] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106, 2021.
- [2] B. Kerbl, G. Kopanas, T. Leimkühler, and G. Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42(4), July 2023.
- [3] L. Zhao, P. Wang, and P. Liu. Bad-gaussians: Bundle adjusted deblur gaussian splatting. In *European Conference on Computer Vision*, pages 233–250. Springer, 2024.
- [4] O. Seiskari, J. Ylilammi, V. Kaatrasalo, P. Rantalankila, M. Turkulainen, J. Kannala, E. Rahtu, and A. Solin. Gaussian splatting on the move: Blur and rolling shutter compensation for natural camera motion. In *European conference on computer vision*, pages 160–177. Springer, 2024.
- [5] J. Lee, D. Kim, D. Lee, S. Cho, M. Lee, W. Lee, T. Kim, D. Wee, and S. Lee. Comogaussian: Continuous motion-aware gaussian splatting from motion-blurred images. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 26415–26424, 2025.
- [6] A. Chen, Z. Xu, A. Geiger, J. Yu, and H. Su. Tensorf: Tensorial radiance fields. In *European conference on computer vision*, pages 333–350. Springer, 2022.
- [7] S. Fridovich-Keil, A. Yu, M. Tancik, Q. Chen, B. Recht, and A. Kanazawa. Plenoxels: Radiance fields without neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5501–5510, 2022.
- [8] S. J. Garbin, M. Kowalski, M. Johnson, J. Shotton, and J. Valentin. Fastnerf: High-fidelity neural rendering at 200fps. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 14346–14355, 2021.
- [9] A. Yu, R. Li, M. Tancik, H. Li, R. Ng, and A. Kanazawa. Plenotrees for real-time rendering of neural radiance fields. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 5752–5761, 2021.
- [10] 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*, pages 18039–18048, 2024.
- [11] 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*, pages 21357–21366, 2024.
- [12] Z. Peng, T. Shao, Y. Liu, J. Zhou, Y. Yang, J. Wang, and K. Zhou. Rtg-slam: Real-time 3d reconstruction at scale using gaussian splatting. In *ACM SIGGRAPH 2024 conference papers*, pages 1–11, 2024.
- [13] S. Ha, J. Yeon, and H. Yu. Rgb-d gs-icp slam. In *European conference on computer vision*, pages 180–197. Springer, 2024.
- [14] 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*, pages 21584–21593, 2024.

- [15] W. Zhang, Q. Cheng, D. Skuddis, N. Zeller, D. Cremers, and N. Haala. Hi-slam2: Geometry-aware gaussian slam for fast monocular scene reconstruction. *IEEE Transactions on Robotics*, 41:6478–6493, 2025.
- [16] S. Wu, Z. Lv, Y. Zhu, D. Frost, Z. Li, L.-Q. Yan, C. Ren, R. Newcombe, and Z. Dong. Monocular online reconstruction with enhanced detail preservation. In *Proceedings of the Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Papers*, pages 1–11, 2025.
- [17] L. Ma, X. Li, J. Liao, Q. Zhang, X. Wang, J. Wang, and P. V. Sander. Deblur-nerf: Neural radiance fields from blurry images. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12861–12870, 2022.
- [18] B. Lee, H. Lee, X. Sun, U. Ali, and E. Park. Deblurring 3d gaussian splatting. In *European Conference on Computer Vision*, pages 127–143. Springer, 2024.
- [19] C. Peng, Y. Tang, Y. Zhou, N. Wang, X. Liu, D. Li, and R. Chellappa. Bags: Blur agnostic gaussian splatting through multi-scale kernel modeling. In *European Conference on Computer Vision*, pages 293–310. Springer, 2024.
- [20] Q. Zhang, D. Rozumny, F. Girlanda, S. Karaoglu, M. Pollefeys, T. Gevers, and M. R. Oswald. Unblur-slam: Dense neural slam for blurry inputs. *arXiv preprint arXiv:2603.26810*, 2026.
- [21] D. Lee, M. Lee, C. Shin, and S. Lee. Dp-nerf: Deblurred neural radiance field with physical scene priors. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12386–12396, 2023.
- [22] P. Wang, L. Zhao, R. Ma, and P. Liu. Bad-nerf: Bundle adjusted deblur neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4170–4179, 2023.
- [23] L. Höllein, A. Božič, M. Zollhöfer, and M. Nießner. 3dgs-lm: Faster gaussian-splatting optimization with levenberg-marquardt. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 26740–26750, 2025.
- [24] A. Hanson, A. Tu, G. Lin, V. Singla, M. Zwicker, and T. Goldstein. Speedy-splat: Fast 3d gaussian splatting with sparse pixels and sparse primitives. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 21537–21546, 2025.
- [25] S. Ren, T. Wen, Y. Fang, and B. Lu. Fastgs: Training 3d gaussian splatting in 100 seconds. *arXiv preprint arXiv:2511.04283*, 2025.
- [26] H. Lin, S. Chen, J. H. Liew, D. Y. Chen, Z. Li, G. Shi, J. Feng, and B. Kang. Depth anything 3: Recovering the visual space from any views. In *ICLR*, 2026.
- [27] C. Wang, H. Tan, W. Yifan, Z. Chen, Y. Liu, K. Sunkavalli, S. Bi, L. Liu, and Y. Hu. tttrm: Test-time training for long context and autoregressive 3d reconstruction. In *CVPR*, 2026.
- [28] J. Wang, M. Chen, N. Karaev, A. Vedaldi, C. Rupprecht, and D. Novotny. Vggt: Visual geometry grounded transformer. In *CVPR*, 2025.
- [29] S. Boche, J. Jung, S. B. Laina, and S. Leutenegger. Okvis2-x: Open keyframe-based visual-inertial slam configurable with dense depth or lidar, and gnss. *IEEE Transactions on Robotics (T-RO) - Special Issue: Visual SLAM*, 2025.
- [30] 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*, 37(6):1874–1890, 2021.

- [31] M. Zwicker, H. Pfister, J. Van Baar, and M. Gross. Ewa splatting. *IEEE Transactions on Visualization & Computer Graphics*, 8(03):223–238, 2002.
- [32] H. B. Nielsen et al. *Damping parameter in Marquardt’s method*, volume 248. IMM, 1999.
- [33] E. Dexheimer and A. J. Davison. Learning a depth covariance function. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13122–13131, 2023.
- [34] V. Ye, R. Li, J. Kerr, M. Turkulainen, B. Yi, Z. Pan, O. Seiskari, J. Ye, J. Hu, M. Tancik, and A. Kanazawa. gsplat: An open-source library for gaussian splatting. *Journal of Machine Learning Research*, 26(34):1–17, 2025.
- [35] T. Schops, T. Sattler, and M. Pollefeys. Bad slam: Bundle adjusted direct rgb-d slam. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 134–144, 2019.
- [36] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers. A benchmark for the evaluation of rgb-d slam systems. In *Proc. of the International Conference on Intelligent Robot Systems (IROS)*, Oct. 2012.

Appendices

A Derivation of the Exposure-Integrated Motion Blur

Starting from Eq. (3), the quadratic exponent can be written as $a\tau^2 + 2b\tau + c$, where $a \geq 0$ measures how fast the Mahalanobis exponent grows along the Gaussian's sweep direction and c is the standard 3DGS exponent of the stationary Gaussian. For $a > 0$, let $\tau^* = -b/a$ denote the closest-approach time. The exponent then attains its minimum $c - b^2/a$, and completing the square gives

$$a\tau^2 + 2b\tau + c = a(\tau - \tau^*)^2 + (c - \frac{b^2}{a}), \quad (5)$$

which factors out the constant term from Eq. (3) and leaves a shifted Gaussian integral over the exposure window. Substituting $u = \sqrt{a/2}(\tau - \tau^*)$ and using $\text{erf}(u) = 2/\sqrt{\pi} \int_0^u \exp(-t^2)dt$,

$$\int_{-t_e/2}^{t_e/2} \exp(-\frac{a}{2}(\tau - \tau^*)^2) d\tau = \sqrt{\frac{\pi}{2a}} [\text{erf}(u_+) - \text{erf}(u_-)], \quad (6)$$

with $u_{\pm} = \sqrt{a/2}(\pm t_e/2 - \tau^*)$. Combining the prefactor o_j/t_e of Eq. (3) with the constant $\exp(-(c - b^2/a)/2)$ from Eq. (5) and the ERF difference from Eq. (6) yields Eq. (4).

Geometric Interpretation. Eq. (4) factors into three terms, $P = (o_j/t_e)\sqrt{\pi/(2a)}$, $G = \exp(-(c - b^2/a)/2)$, and $E = \text{erf}(u_+) - \text{erf}(u_-)$, each of which admits an intuitive reading. The prefactor P normalizes the sweep by its effective duration: $\sqrt{1/a}$ scales inversely with how fast the Gaussian crosses the pixel, so P grows when the Gaussian lingers near $\mathbf{x}$ and shrinks when it sweeps past quickly. The exponential G is the kernel value at the Mahalanobis closest-approach time τ^* along the projected trajectory. The ERF difference E measures the normalized Gaussian mass captured within the finite exposure window; it approaches 2 when the window covers nearly all mass around τ^* and shrinks toward zero when this mass lies far outside $[-t_e/2, t_e/2]$. Thus, PGE decomposes the analytic footprint into a duration-normalized amplitude, a peak kernel value, and an exposure-window coverage term.

B Numerical Stability near the Stationary Limit

The exposure-integrated footprint of Eq. (4) contains the prefactor $P = (o_j/t_e)\sqrt{\pi/(2a)}$, which diverges as the Mahalanobis sweep coefficient $a = \mathbf{v}'_{i,j}{}^\top \Sigma'_{i,j}{}^{-1} \mathbf{v}'_{i,j}$ approaches zero, even though the integrand of Eq. (3) remains finite. Physically, $a \rightarrow 0$ occurs when the projected pixel velocity $\mathbf{v}'_{i,j}$ vanishes or when the Gaussian is highly elongated along the sweep direction, making the Mahalanobis growth along that direction nearly zero. In these cases, the Gaussian is effectively stationary on the image plane over the exposure window.

Continuity at $a \rightarrow 0$. The apparent singularity of P is removable. Recall that the quadratic exponent in Eq. (3) can be written as $a\tau^2 + 2b\tau + c$. For $a > 0$, the error-function arguments in Eq. (4) are

$$u_{\pm} = \sqrt{\frac{a}{2}} \left(\pm \frac{t_e}{2} - \tau^* \right), \quad \tau^* = -\frac{b}{a}. \quad (7)$$

Equivalently, we can write

$$u_{\pm} = u_0 \pm \Delta, \quad u_0 = \frac{b}{\sqrt{2a}}, \quad \Delta = \frac{t_e}{2} \sqrt{\frac{a}{2}}. \quad (8)$$

For $a > 0$, since Σ'^{-1} is positive definite, Cauchy-Schwarz gives $b^2 \leq ac$. Therefore $u_0^2 = b^2/(2a) \leq c/2$, so u_0 remains bounded as $a \rightarrow 0$.

$$\begin{aligned} E &= \text{erf}(u_+) - \text{erf}(u_-) = \text{erf}(u_0 + \Delta) - \text{erf}(u_0 - \Delta) \\ &= \frac{4}{\sqrt{\pi}} e^{-u_0^2} \Delta + \mathcal{O}(\Delta^3) = \frac{2}{\sqrt{\pi}} t_e \sqrt{\frac{a}{2}} \exp\left(-\frac{b^2}{2a}\right) + \mathcal{O}(a^{3/2}). \end{aligned} \quad (9)$$

Substituting this expression into Eq. (4), the $\sqrt{a}$ factors cancel and the exponential terms combine as

$$\exp\left(-\frac{c - b^2/a}{2}\right) \exp\left(-\frac{b^2}{2a}\right) = \exp\left(-\frac{c}{2}\right). \quad (10)$$

Hence,

$$\lim_{a \rightarrow 0} \bar{\alpha}_{i,j}(\mathbf{x}) = o_j \exp\left(-\frac{1}{2} \boldsymbol{\delta}^\top \boldsymbol{\Sigma}'_{i,j}{}^{-1} \boldsymbol{\delta}\right) = \alpha_{i,j}(\mathbf{x}, \boldsymbol{\mu}'_{i,j}), \quad (11)$$

recovering the standard static 3DGS footprint. This is consistent with the original integral: when $a = 0$, the projected sweep has zero Mahalanobis length, the quadratic exponent becomes constant over the exposure interval, and Eq. (3) collapses to $o_j \exp(-c/2)$.

Practical Implementation. Although Eq. (4) is continuous at the limit, evaluating it in single precision at small a is unstable: $\sqrt{1/a}$ amplifies floating-point noise, b/a produces 0/0-type cancellations, and $\text{erf}(u_\pm)$ loses significant digits when $|u_\pm| \ll 1$. We therefore branch in the rasterizer on a tolerance ε_a :

$$\bar{\alpha}_{i,j}(\mathbf{x}) = \begin{cases} PGE & \text{if } a > \varepsilon_a, \\ o_j \exp(-\frac{1}{2} \boldsymbol{\delta}^\top \boldsymbol{\Sigma}'_{i,j}{}^{-1} \boldsymbol{\delta}) & \text{otherwise,} \end{cases} \quad (12)$$

with $\varepsilon_a = 10^{-6} \text{ px}^2$ in our implementation. The static branch is mathematically the $a \rightarrow 0$ limit of the analytically integrated motion-blur model expression by Eq. (11), so the rendered image is continuous across the branch boundary. In addition, we (i) reject samples with numerically negative quadratic-form values ($c < 0$) caused by finite-precision conic round-off, (ii) skip Gaussians whose minimum exponent $(c - b^2/a)/2$ exceeds a cut-off so that G underflows to zero, and (iii) clamp $\bar{\alpha}_{i,j} \leq 1 - \varepsilon$ before front-to-back compositing, matching the safeguards of standard 3DGS [2].

Gradient Continuity. The same branch is applied in the backward pass to keep the gradients well-behaved near the stationary limit. On the static branch, $\partial \bar{\alpha} / \partial \boldsymbol{\mu}'$, $\partial \bar{\alpha} / \partial \boldsymbol{\Sigma}'^{-1}$ reduce to the standard 3DGS VJPs, while $\partial \bar{\alpha} / \partial \mathbf{v}'_{i,j} = \mathbf{0}$ on this branch because the rendered opacity is evaluated by the static footprint. Empirically, this guarantees that Gaussians whose projected velocity transiently passes through zero, common for far-field Gaussians or for keyframes captured during near-stationary segments of the trajectory, still receive valid gradients from the geometric residual and continue to optimize as ordinary static primitives.

C LM Refinement Details

Let $\boldsymbol{\theta}$ collect the Gaussian parameters and $\mathbf{r}$ denote the per-pixel residual under the standard 3DGS loss, with Huber weights collected in $\mathbf{W}$. The damped Gauss-Newton normal equation is

$$(\mathbf{J}^\top \mathbf{W} \mathbf{J} + \lambda \text{diag}(\mathbf{J}^\top \mathbf{W} \mathbf{J})) \delta \boldsymbol{\theta} = -\mathbf{J}^\top \mathbf{W} \mathbf{r}, \quad (13)$$

solved by preconditioned conjugate gradient with the diagonal of $\mathbf{J}^\top \mathbf{W} \mathbf{J}$ as the preconditioner. λ is initialized per-keyframe as $\lambda_0 = \tau \cdot \max_k [\text{diag}(\mathbf{J}^\top \mathbf{W} \mathbf{J})]_k$ with $\tau = 10^{-3}$, following Nielsen [32], and updated by the standard accept/reject rule.

D Online Hessian-Based Pruning Details

Chain rule. Under standard 3DGS compositing, the opacity-axis gradient for Gaussian j at pixel $\mathbf{x}$ in keyframe i factors as

$$\frac{\partial L_i}{\partial o_j} = g_{i,j}(\mathbf{x}) \frac{\partial L}{\partial \alpha_{i,j}(\mathbf{x})}, \quad g_{i,j}(\mathbf{x}) = \exp\left(-\frac{1}{2} (\boldsymbol{\mu}'_{i,j} - \mathbf{x})^\top \boldsymbol{\Sigma}'_{i,j}{}^{-1} (\boldsymbol{\mu}'_{i,j} - \mathbf{x})\right), \quad (14)$$

where $g_{i,j}$ is the screen-space Gaussian kernel value and $\partial L / \partial \alpha$ carries the standard front-to-back compositing chain. Summing the squares over visible pixels and active keyframes gives the score in the main text.

Comparison with Speedy-Splat. Speedy-Splat scores Gaussians by the squared kernel-value sensitivity of the rendered image, $\sum_p (\partial I / \partial G_{i,p})^2$, extracted by an offline backward pass over all training views. Our score is the opacity-axis Gauss-Newton diagonal of the same training loss and is read off the gsplat backward, already executed during online optimization, adding no rendering or extra backward cost.

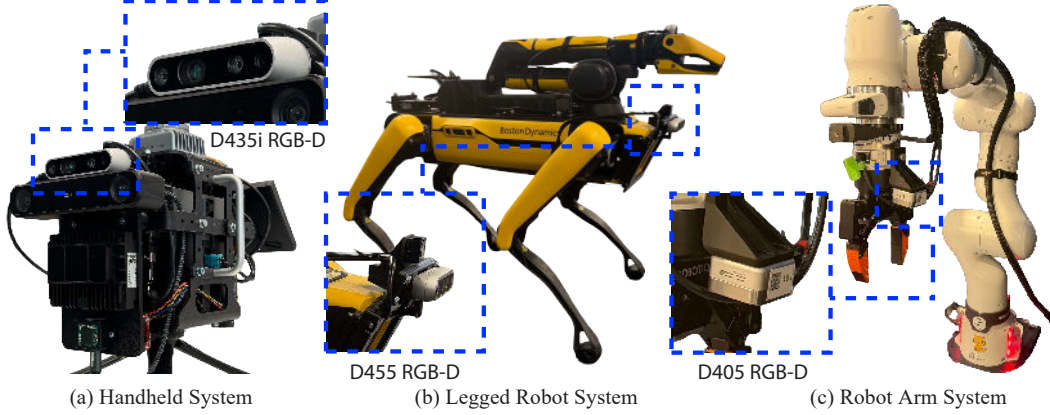


Figure 9: Real-world robot platforms used to acquire the custom dataset. (a) Handheld system with an Intel RealSense D435i RGB-D camera. (b) Spot quadruped robot with an Intel RealSense D455 RGB-D camera. (c) Franka robot arm with an Intel RealSense D405 RGB-D camera.

E Custom Dataset

For a broader evaluation, we acquire a custom dataset using three real-world robot platforms: a handheld rig, a legged robot, and a robot arm. The handheld and legged-robot systems cover large-scale scenes under different motion patterns and viewing directions, while the robot-arm system captures classical tabletop scenes with controlled manipulator motion. Fig. 9 shows the hardware setup for each platform. The dataset contains seven sequences in total: two handheld sequences, two legged-robot sequences, and three robot-arm sequences.

- **Stair:** Captured in a large stair area using the handheld system. The sequence includes motion blur induced by human locomotion and vertical camera motion.
- **Kitchen:** Captured in a kitchen area using the handheld system. The sequence includes motion blur induced by human locomotion and horizontal camera motion.
- **Dog_Park:** Captured in a legged-robot experimental area using a Spot quadruped robot. The robot traverses the same area in four loops, producing legged-locomotion induced motion blur.
- **Corridor:** Captured in a square corridor containing office and kitchen areas using a Spot quadruped robot. The robot completes one loop, and the sequence includes motion blur induced by legged locomotion.
- **Arm1, Arm2, Arm3:** Captured on classic tabletop scenes with various objects using a Franka robot arm. These sequences include robot-arm-induced motion blur.