T-FastGS: Transient-Robust and Efficient 3D Gaussian Splatting

Anonymous authors
Paper under double-blind review

Abstract

Transient objects break the multi-view consistency assumption of 3D Gaussian Splatting (3DGS), leading to degraded reconstruction quality. Existing methods struggle to yield robust reconstruction in the presence of transients while preserving high training efficiency. To tackle this challenge, we present T-FastGS, a transient-robust and efficient reconstruction framework. We find that transient content introduces ambiguity into error-aware Gaussian density control. To mitigate this problem, we propose Mask-Guided Error-Aware Density Control (MGDC), which constructs reliable cues for density adjustment by distinguishing static and transient regions. Moreover, we identify a timescale mismatch between transient mask learning and Gaussian optimization. Motivated by this finding, we introduce an Asynchronous Alternating Optimization (AAO) scheme that cuts redundant updates to the mask predictor and retains training efficiency. Experiments show that T-FastGS achieves competitive rendering quality with state-of-the-art transient-free 3DGS methods on two benchmark datasets, while delivering around $3\times$ or higher training speedup over these methods.

1 Introduction

Recent advances in 3D Gaussian Splatting (3DGS) (Kerbl et al., 2023; Lu et al., 2024; Huang et al., 2024; Yu et al., 2024) have enabled high-quality novel view synthesis with real-time rendering performance. 3DGS assumes the input scene is fully static **to maintain multi-view consistency**. However, real-world scenes commonly contain transient objects such as pedestrians, animals, and vehicles. These dynamic elements degrade reconstruction quality and corrupt the optimization signals for learning scene representations.

Current 3DGS methods for transient objects broadly fall into two categories. The first (Sabour et al., 2025; Fu et al., 2025; Seo et al., 2026) predicts transient masks to suppress their influence during Gaussian optimization. The second (Wang et al., 2025b; Lin et al., 2025; Wang et al., 2025b) explicitly decomposes scenes into static and transient components with separate representations. While these methods improve transient robustness, they are typically built upon vanilla 3DGS (Kerbl et al., 2023) and devote less attention to training efficiency. Meanwhile, recent efficient 3DGS works (Chen et al., 2025; Mallick et al., 2024; Ren et al., 2025) accelerate training by refining density control or rasterization pipelines. However, they adopt the static-scene assumption and fail to handle transient distractors. Collectively, achieving both transient robustness and high training efficiency remains challenging for 3DGS.

To tackle this challenge, we build our method on FastGS (Ren et al., 2025), an efficient 3DGS framework with error-aware density control. Reconstruction errors drive Gaussian densification and pruning, resting on the assumption that photometric errors primarily stem from inadequate static geometry modeling. This assumption breaks down in dynamic scenes, where transient objects introduce spurious errors unrelated to real geometric defects. Consequently, high-error regions become ambiguous, as errors may arise either from incomplete static reconstruction or transient noise. Simply discarding transient regions suppresses potential densification of underlying static structures in those regions. Furthermore, mask prediction and Gaussian training converge over distinct timescales. Incorporating transient prediction into the 3DGS pipeline via synchronous updates introduces extra optimization overhead and considerably degrades training speed.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

These observations lead to our key insights. First, transient objects alter the interpretation of error signals used for Gaussian density control. Reliable density control requires disentangling valid static structural cues from transient-induced errors, rather than simply masking out transient content. In other words, Gaussian density adjustment should rely on trustworthy static regions, while preserving densification opportunities within transient regions to reconstruct occluded static structures. Second, transient mask learning and Gaussian optimization should be decoupled to accommodate their different convergence patterns.

To this end, we propose T-FastGS, an efficient and robust framework for transient-free 3D Gaussian Splatting. First, we introduce **Mask-Guided Error-Aware Density Control (MGDC)** to eliminate reconstruction error ambiguity in density adjustment. MGDC builds reliable adjustment signals from static regions and leverages transient-region cues as supplements, enabling efficient and effective Gaussian density control under transients. Second, we introduce an **Asynchronous Alternating Optimization (AAO)** scheme to reduce the overhead from transient mask prediction. By decoupling mask updating from Gaussian training according to their different optimization timescales, AAO avoids redundant computation without degrading reconstruction quality.

Our contributions are summarized as follows:

- We propose **T-FastGS**, an efficient and robust 3D Gaussian Splatting framework that extends efficient Gaussian optimization to scenes with transient objects.
- We identify that transient objects invalidate the implicit assumption underlying error-aware Gaussian density control, and introduce **MGDC** to recover reliable density adjustment cues for robust Gaussian optimization.
- We recognize the mismatch in optimization timescales between transient mask prediction and Gaussian optimization, and propose **AAO**, which boosts training efficiency by asynchronously updating the transient mask predictor.
- Extensive experiments on two benchmark datasets show that T-FastGS achieves rendering quality comparable to state-of-the-art transient-free 3DGS methods, yielding around $3\times$ or higher speedup over the evaluated baselines.

2 Related Work

2.1 Transient-Robust 3DGS

To handle transient distractors, existing robust 3DGS methods can be mainly divided into two groups.

Transient Mask Prediction. The first group focuses on identifying transient regions via learned transient masks and suppresses their influence during Gaussian optimization. Specifically, WildGaussians (Kulhanek et al., 2024), SpotlessSplat (Sabour et al., 2025) and RobustSplat (Fu et al., 2025) use MLPs to predict transient masks from Stable Diffusion (Rombach et al., 2021) and DINOv2 (Oquab et al., 2023) features. Leveraging DINOv2 and SAM2 (Kirillov et al., 2023), T-3DGS (Markin et al., 2024) estimates transient masks by measuring the distribution discrepancy between features of input images and rendered views. While effective at removing transient distractors, these methods are build upon vanilla 3DGS (Kerbl et al., 2023) and pay less attention to training efficiency.

Joint Static-Transient Modeling. The second group jointly models static and transient components. For example, HybridGS (Lin et al., 2025) represents dynamic objects using 2D Gaussians while reconstructing static regions with 3D Gaussians, enabling flexible decomposition of transient components. DeGauss (Wang et al., 2025a) introduces extra Gaussian parameters to estimate probabilities for dynamic foreground and static background components. While these approaches explicitly model transient components, they demand additional representations and optimization procedures, which limits their compatible with efficiency-oriented frameworks.

2.2 EFFICIENT 3DGS

To improve the training and rendering efficiency of 3DGS, existing approaches broadly focus on two directions.

Improved Gaussian Density Control. The first direction refines Gaussian density control strategies. To prevent excessive growth of Gaussians, Taming-GS (Mallick et al., 2024) introduces a budgeting mechanism that calculates Gaussian importance from primitive attributes. DashGaussian (Chen et al., 2025) dynamically schedules primitive growth according to rendering resolution, avoiding unnecessary densification during low-resolution optimization. For Gaussian pruning, MiniSplatting (Fang & Wang, 2024) proposes an intersection-preserving sampling strategy for efficient Gaussian simplification. Speedy-Splat (Hanson et al., 2025a) and PUP 3DGS (Hanson et al., 2025b) remove redundant Gaussians using Hessian-based importance estimation across training views. FastGS (Ren et al., 2025) further evaluates per-Gaussian contributions for densification and pruning by computing importance and pruning scores from multi-view reconstruction errors.

Rasterization Refinement. The second direction focuses on accelerating Gaussian rasterization. Taming-3DGS (Mallick et al., 2024) replaces per-pixel parallel backpropagation with per-splat counterparts. Methods such as StopThePop (Radl et al., 2024), Speedy-Splat (Hanson et al., 2025a), and FlashGS (Feng et al., 2025) accelerate rasterization by reducing Gaussian-tile pairs through more precise tile intersection tests. FastGS (Ren et al., 2025) integrates the compact box to prune Gaussian-tile pairs with minimal contribution, further lifting rasterization efficiency.

Despite these advances, current efficient 3DGS methods are primarily developed for static scenes. Their acceleration schemes and reconstruction quality can deteriorate when transient objects appear.

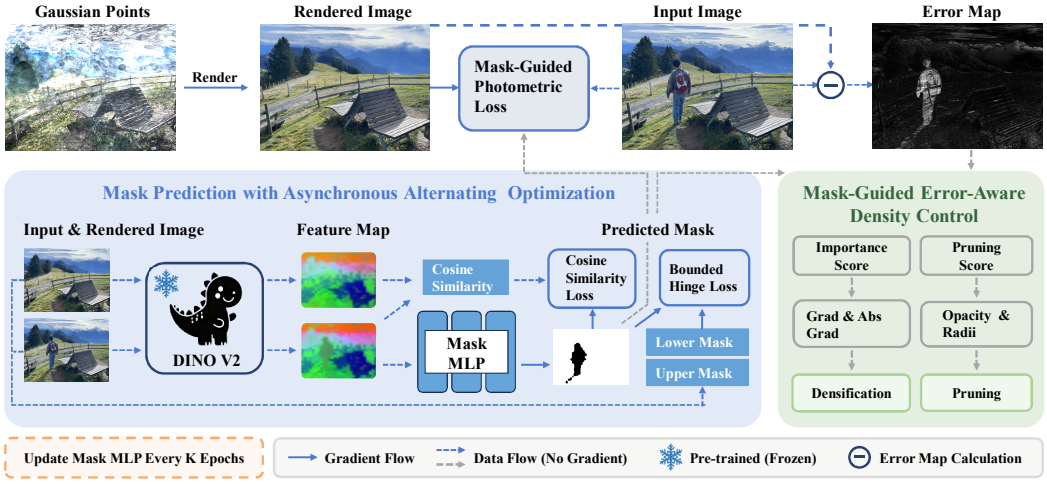


Figure 1: **Overview of the proposed T-FastGS framework.** We first present a Mask-Guided Error-Aware Density Control (MGDC) strategy to achieve reliable Gaussian densification and pruning in scenes with transient objects. Furthermore, by leveraging the distinct optimization timescales between mask prediction and Gaussian training, we introduce an Asynchronous Alternating Optimization (AAO) scheme to eliminate redundant mask updates and accelerate the overall training pipeline.

3 METHOD

3.1 OVERVIEW

Given input images that contain transient objects, our goal is to reconstruct a clean static 3D Gaussian representation while preserving fast training speed. As illustrated in Fig. 1, our T-FastGS framework incorporates two core components to boost reconstruction robustness and training efficiency in the presence of transients. First, Mask-Guided Error-Aware Density Control (MGDC) computes reliable cues for Gaussian densification and pruning, mitigating the adverse impact of

transient objects. Second, we present the Asynchronous Alternating Optimization (AAO) scheme to decouple transient mask learning from Gaussian optimization. This design cuts unnecessary mask updates and sustains high training efficiency.

3.2 T-FastGS

Preliminaries. Given a set of calibrated images $\{I_i\}_{i=1}^N$ along with their camera parameters $\{C_i\}_{i=1}^N$, 3D Gaussian Splatting represents a scene as a collection of anisotropic Gaussian primitives $\{G_k\}_{k=1}^M$. Each Gaussian is parameterized by its position, covariance, opacity, and color attributes. During rendering, the 3D Gaussians are projected onto the image plane and composited in depth order. The rendered pixel color $\hat{I}(p)$ is formulated as:

$$\hat{I}\left(p\right) = \sum_{k \in \mathcal{N}(p)} \alpha_k\left(p\right) c_k \prod_{j<k} \left(1 - \alpha_j\left(p\right)\right), \tag{1}$$

where c_k and $\alpha_k\left(p\right)$ denote the color and projected opacity of the k-th Gaussian at pixel p, and $\mathcal{N}(p)$ denotes the set of Gaussians overlapping pixel p, sorted by depth.

3DGS methods(Kerbl et al., 2023; Ren et al., 2025) optimize Gaussian parameters by minimizing the photometric loss:

$$\mathcal{L}_{\text{photo}} = \left(1 - \lambda\right) \mathcal{L}_1 \left(I, \hat{I}\right) + \lambda \left(1 - \mathcal{L}_{\text{ssim}} \left(I, \hat{I}\right)\right), \tag{2}$$

where $\mathcal{L}_1$ represents the mean absolute error, and $\mathcal{L}_{\text{ssim}}$ refers to the structural similarity computed over the entire image.

Mask-Guided Photometric Loss. When transient objects appear in training views, the static scene assumption is violated. Following existing transient-free approaches (Sec. 2.1), we incorporate a binary transient mask M_b into the photometric loss (Eq. 2) to exclude transient pixels during optimization:

$$\mathcal{L}_{\text{masked-photo}} = \left(1 - \lambda\right) \mathcal{L}_1 \left(I \odot M_b, \hat{I} \odot M_b\right) + \lambda \left(1 - \mathcal{L}_{\text{ssim}} \left(I \odot M_b, \hat{I} \odot M_b\right)\right), \tag{3}$$

where $M_b\left(p\right) = 1$ indicates static pixels and $M_b\left(p\right) = 0$ indicates transient pixels.

Transient Mask Prediction. Following Fu et al. (2025), we use a lightweight two-layer MLP to predict transient masks. Given DINOv2 (Oquab et al., 2023) features F_I from image I, the MLP outputs pixel-wise transient probabilities P_m:

$$P_m = \sigma\left(\text{MLP}\left(F_I\right)\right). \tag{4}$$

Thresholding P_m yields the binary mask M_b:

$$M_b = \mathbf{1}(P_m > \tau_b), \tag{5}$$

where τ_b is the probability threshold. Additional implementation details are deferred to Appendix A.

Ambiguity in Error-Aware Density Control. To boost training efficiency, we adopt the error-aware density control strategy from FastGS (Ren et al., 2025), which computes importance and pruning scores for each Gaussian to guide densification and pruning. However, transient objects introduce ambiguity into error-aware density control. Specifically, the original FastGS randomly samples V training views and builds an indicator map $\mathcal{M}_j\big(j \in [0, V)\big)$ per view to flag high-error pixels:

$$\mathcal{M}_j = \mathbf{1}\left(\mathcal{N}\left(\mathcal{L}_1(I_j, \hat{I}_j)\right) > \tau_e\right), \tag{6}$$

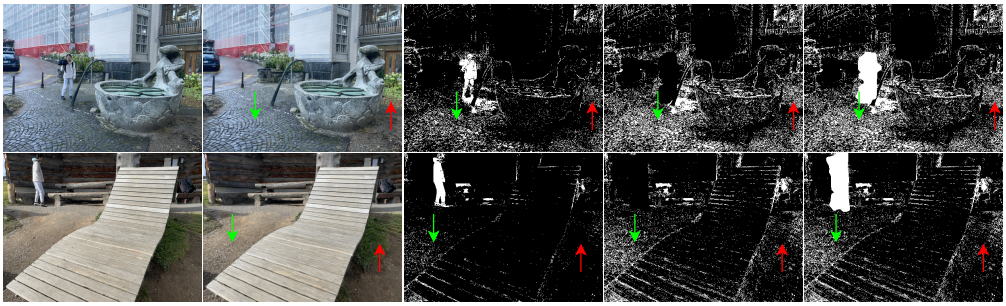
where $\mathcal{N}\left(\cdot\right)$ denotes min–max normalization and τ_e denotes the reconstruction error threshold.

When transients exist, high reconstruction errors stem either from under-optimized static regions requiring densification or from transient regions that should be excluded. Consequently, naive error-aware density control leads to unreliable importance and pruning scores, motivating our Mask-Guided Error-Aware Density Control (MGDC).

3.3 MASK-GUIDED ERROR-AWARE DENSITY CONTROL

The ambiguity leads to inaccurate construction of the indicator map: densification signals tend to concentrate on high-error transient regions and reduce densification opportunities for static regions (Fig. 2(c)).

To resolve this ambiguity, an intuitive solution is to construct the indicator map (Eq. 6) solely from static regions. However, this naive strategy discards valuable densification cues within transient areas, where underlying static structures still require refinement (Fig. 2(d)).



(a) (b) (c) (d) (e)

Figure 2: **Comparison of indicator maps under transient objects.** (a) Input image. (b) Intermediate rendering, where green/red arrows mark under-reconstructed regions requiring further densification. (c) Full-image error-based indicator maps. (d) Static-only indicator maps. (e) Our MGDC indicator maps.

Since the mask-guided photometric loss (Eq. 3) suppresses transient reconstruction and the opacity reset mechanism (Kerbl et al., 2023; Ren et al., 2025) gradually removes transient-related Gaussians, the surviving Gaussians within transient regions mainly correspond to underlying static content. Therefore, rather than discarding transient regions entirely, we propose MGDC to build a more robust indicator map (Fig. 2(e)) and adopt mask-guided error-aware strategies for Gaussian densification and pruning.

Mask-Guided Error-Aware Densification. For densification, MGDC first computes reconstruction errors exclusively over static regions. Given the $\mathcal{L}_1$ error map (E_j) and transient mask (M_j^b) for j-th view, the error restricted to static regions is defined as:

$$E_j^s = E_j \odot M_j^b, \tag{7}$$

followed by min–max normalization:

$$\hat{E}_j^s = \frac{E_j^s - \min(E_j^s)}{\max(E_j^s) - \min(E_j^s)}. \tag{8}$$

Applying the error threshold τ_e, we construct the static indicator map:

$$\mathcal{M}_j^s = \mathbf{1}\Big(\hat{E}_j^s > \tau_e\Big). \tag{9}$$

For transient regions, we treat all such areas as densification candidates. Thus, the transient indicator map is $\mathcal{M}_j^t = 1 - M_j^b$.

The final indicator map is generated by:

$$\mathcal{M}_j = \mathcal{M}_j^s \vee \mathcal{M}_j^t. \tag{10}$$

From these indicator maps of the sampled training views, we compute the densification importance score for each Gaussian:

$$S_d^i = \frac{1}{V} \sum_{j=1}^{V} \sum_{p \in \Omega_i} \mathbf{1}(\mathcal{M}_j\,(p) = 1), \tag{11}$$

where Ω_i denotes the set of pixels affected by the i-th Gaussian, and $\mathcal{M}_j(p) = 1$ marks pixels that have high errors.

By incorporating transient regions, MGDC retains opportunities to refine underlying static geometry within transient areas, bypassing unreliable importance estimates.

Mask-Guided Error-Aware Pruning. For pruning, transient regions must be excluded to prevent biased Gaussian removal. Accordingly, MGDC calculates pruning scores solely from static-region indicators. Following FastGS (Ren et al., 2025), we compute the masked photometric error E_j as:

$$E_j = (1-\lambda)\,\mathcal{L}_1\left(I_j \odot M_j^b, \hat{I}_j \odot M_j^b\right) + \lambda\left(1 - \mathcal{L}_{ssim}\left(I_j \odot M_j^b, \hat{I}_j \odot M_j^b\right)\right). \tag{12}$$

The pruning score aggregates multi-view static indicator statistics and masked photometric errors, followed by normalization:

$$S_p^i = \mathcal{N}\left(\sum_{j=1}^{N}\sum_{p\in\Omega_i} \mathbf{1}\big(\mathcal{M}_j^s(p)=1\big)\cdot E_j\right). \tag{13}$$

In contrast to densification, pruning excludes transient regions. Including them would lead to biased statistics, since the masked photometric error is independent of multi-view indicator statistics.

In summary, MGDC adopts distinct indicator construction strategies for error-aware density control. Transient regions are incorporated for densification to preserve underlying static structures, yet excluded for pruning to prevent unreliable pruning decisions. This asymmetric design enables robust Gaussian density management under transient objects.

3.4 Asynchronous Alternating Optimization

For transient-robust 3DGS reconstruction, accurate transient masks are critical for suppressing inconsistent observations. Existing approaches typically optimize the mask predictor synchronously together with Gaussian parameters. However, we observe that mask prediction and Gaussian optimization evolve at distinct timescales.

Timescale Mismatch. Specifically, predicted transient masks stabilize rapidly in early iterations, whereas Gaussian parameters keep evolving via iterative densification and pruning. To quantify mask stability, we compute the IoU between intermediate mask predictions and the final averaged mask:

$$\mathrm{IoU}_{\text{avg}} = \frac{1}{N}\sum_{i=1}^{N}\mathrm{IoU}\left(M_i^t, \bar{M}_i^{\text{final}}\right), \tag{14}$$

where M_i^t denotes the predicted mask of the i-th view at iteration t, $\bar{M}_i^{\text{final}}$ represents the mask averaged over the last few iterations.

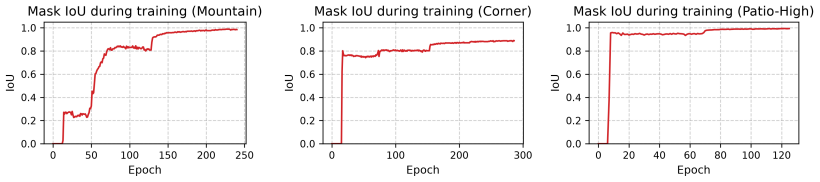


Figure 3: **Mask IoU evolution curves during training.** IoU between intermediate predictions and final averaged masks on three scenes from NeRF On-the-go dataset (Ren et al., 2024). The curves show that mask predictor evolves at a low frequency.

We plot the evolution of Mask IoU over training iterations in Fig. 3. The IoU rises quickly in the early training phase and plateaus across different scenes. Although scenes with complex backgrounds (e.g., Mountain) may exhibit multiple convergence phases, mask predictions remain stable for long optimization intervals after each update. This indicates that mask prediction evolves at a much lower frequency than Gaussian optimization, making synchronous mask updates computationally redundant.

AAO. Motivated by this finding, we propose Asynchronous Alternating Optimization (AAO), which decouples mask prediction from Gaussian optimization by updating the mask predictor periodically.

Concretely, the mask predictor is updated every K epochs:

$$\Theta_G^{t+1} = \Theta_G^t - \eta_G \nabla_{\Theta_G} \mathcal{L}_G, \tag{15}$$

$$\Theta_M^{t+1} = \begin{cases} \Theta_M^t - \eta_M \nabla_{\Theta_M} \mathcal{L}_M, & t \bmod K = 0 \\ \Theta_M^t, & t \bmod K \neq 0 \end{cases} \tag{16}$$

where Θ_G and Θ_M correspond to Gaussian parameters and mask predictor parameters, respectively.

By eliminating unnecessary updates of the mask predictor, AAO reduces redundant computation while maintaining stable optimization.

4 Experimental Results

Datasets. We evaluate our method on two challenging benchmark datasets: NeRF On-the-go (Ren et al., 2024) and RobustNeRF (Sabour et al., 2023). these datasets contain casually captured indoor and outdoor scenes with diverse transient object ratios, posing challenges to reconstruction fidelity.

Implementation Details. We adopt the model architecture and training pipeline from FastGS (Ren et al., 2025), retaining all hyperparameters except those related to Gaussian density control. Under this setting, we develop two model variants, T-FastGS and T-FastGS-Big, to trade off rendering quality and training speed. In our asynchronous alternating optimization, we cache predicted transient masks to further reduce computational overhead and guarantee that the mask predictor is trained for at least 30 epochs. More implementation details can be found in the Appendix A .

Baselines. We benchmark T-FastGS against two groups of methods. The static-scene baselines include vanilla 3DGS (Kerbl et al., 2023) and FastGS (Ren et al., 2025). We also compare with recent transient-robust reconstruction approaches: SpotLessSplats (Sabour et al., 2025), T-3DGS (Markin et al., 2024), RobustSplat (Fu et al., 2025) and PDF-GS (Seo et al., 2026). For fair comparison, we use their publicly released implementations and adopt identical camera matrices across all experiments. We evaluate performance both qualitatively and quantitatively using PSNR, SSIM, LPIPS, training time, and Gaussian count N_{GS}.

4.1 Evaluation on NeRF On-the-go Dataset

We first evaluate T-FastGS on the NeRF On-the-go dataset (Ren et al., 2024). Figure 4 presents qualitative comparisons. FastGS exhibits noticeable artifacts. Among transient-free baselines, our method benefits from the proposed MGDC and recovers superior high-frequency texture details. Table 1 shows that T-FastGS achieves competitive PSNR, SSIM and LPIPS across all three scenes with the fewest Gaussians, while delivering $\geq 3\times$ faster training than existing transient-free methods. Notably, T-FastGS-Big outperforms prior approaches on LPIPS. This confirms that our method improves robustness to transients while retaining high efficiency.

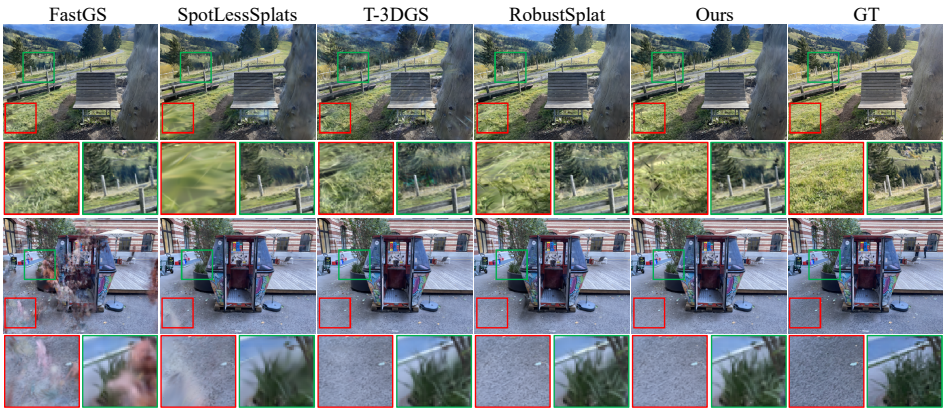


Figure 4: Qualitative results on two scenes from the NeRF On-the-go dataset.

Table 1: Quantitative comparison on NeRF On-the-go Dataset. The best results are highlighted in bold, and the second in underline.

Method	Mountain					Corner					Spot				
	PSNR ↑	SSIM ↑	LPIPS ↓	Time ↓	N_{GS} ↓	PSNR ↑	SSIM ↑	LPIPS ↓	Time ↓	N_{GS} ↓	PSNR ↑	SSIM ↑	LPIPS ↓	Time ↓	N_{GS} ↓
3DGS (Kerbl et al., 2023)	19.21	0.689	0.228	9min25s	1.60M	22.65	0.832	0.164	7min23s	1.34M	18.80	0.730	0.322	9min27s	1.54M
FastGS(Ren et al., 2025)	21.17	0.724	0.212	2min27s	0.22M	24.45	0.862	0.144	2min24s	0.11M	21.44	0.798	0.242	2min31s	0.19M
SpotLessSplats (Sabour et al., 2025)	21.25	0.660	0.244	23min04s	0.61M	25.26	0.848	0.155	24min03s	0.72M	23.94	0.812	0.207	25min02s	0.53M
T-3DGS (Markin et al., 2024)	20.34	0.700	0.227	30min01s	1.99M	25.96	0.885	0.119	26min50s	1.01M	23.67	0.887	0.130	27min13s	1.10M
RobustSplat (Fu et al., 2025)	21.33	**0.734**	0.204	11min50s	1.12M	26.25	**0.895**	0.105	10min18s	0.70M	25.17	0.906	0.102	10min05s	0.54M
PDF-GS (Seo et al., 2026)	21.34	0.728	0.202	11min38s	1.10M	26.14	0.887	0.112	9min58s	0.71M	25.58	0.898	0.115	9min48s	0.48M
T-FastGS (Ours)	**21.36**	0.728	0.212	**3min15s**	**0.23M**	**26.41**	0.892	0.112	**3min03s**	**0.11M**	**25.86**	0.904	0.102	**3min17s**	**0.18M**
T-FastGS-Big (Ours)	21.23	0.730	**0.190**	3min47s	0.46M	26.33	0.892	**0.102**	3min04s	0.20M	25.60	**0.908**	**0.095**	3min44s	0.31M

4.2 Evaluation on RobustNeRF Dataset

To further evaluate the efficiency and robustness of our method, we perform comparisons against baselines on the RobustNeRF dataset (Sabour et al., 2023). Qualitative comparisons are provided in Fig. 5, demonstrating that our method produces transient-robust reconstructions with fine details. Quantitative results are listed in Table 2. Our method maintains competitive performance relative to SOTA approaches and yields around 3× or higher speedup.

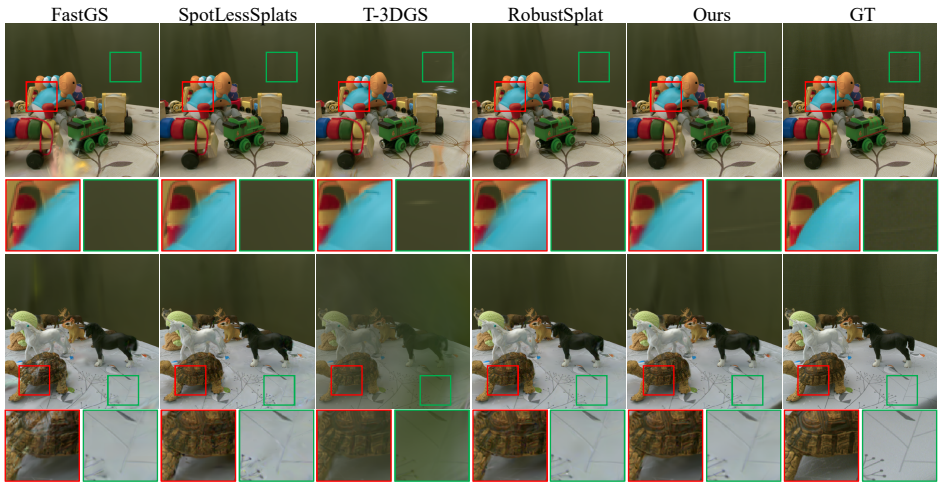


Figure 5: Qualitative results on two scenes from the RobustNeRF dataset.

Table 2: Quantitative comparison on RobustNeRF dataset. The best results are highlighted in bold, and the second in underline.

Method	Android					Crab2					Yoda				
	PSNR ↑	SSIM ↑	LPIPS ↓	Time ↓	N_{GS} ↓	PSNR ↑	SSIM ↑	LPIPS ↓	Time ↓	N_{GS} ↓	PSNR ↑	SSIM ↑	LPIPS ↓	Time ↓	N_{GS} ↓
3DGS (Kerbl et al., 2023)	23.21	0.792	0.161	7min24s	1.12M	30.44	0.917	0.180	4min41s	0.50M	28.84	0.904	0.193	4min51s	0.58M
FastGS (Ren et al., 2025)	24.44	0.818	0.153	2min31s	0.14M	32.86	0.929	0.176	2min30s	0.10M	32.29	0.934	0.164	2min33s	0.11M
SpotLessSplats (Sabour et al., 2025)	24.42	0.810	0.159	24min47s	0.70M	34.11	0.931	0.169	25min33s	0.25M	34.70	0.937	0.156	25min23s	0.27M
T-3DGS (Markin et al., 2024)	24.79	0.830	0.133	27min47s	1.28M	32.45	0.924	0.187	25min40s	0.42M	32.26	0.922	0.181	25min50s	0.39M
RobustSplat (Fu et al., 2025)	24.76	**0.832**	**0.125**	11min28s	0.86M	34.78	0.940	0.157	9min21s	0.22M	35.26	0.944	0.154	10min24s	0.25M
PDF-GS (Seo et al., 2026)	24.35	0.825	0.127	11min21s	0.88M	**35.12**	0.940	0.154	8min57s	0.26M	**35.32**	**0.945**	0.149	7min29s	0.27M
T-FastGS (Ours)	**24.94**	**0.832**	0.134	**3min24s**	**0.14M**	34.65	0.938	0.163	**3min10s**	**0.10M**	35.01	0.944	0.154	**3min27s**	**0.11M**
T-FastGS-Big (Ours)	24.83	0.828	0.126	3min55s	0.33M	34.79	**0.941**	**0.148**	3min16s	0.15M	34.70	0.944	**0.139**	3min41s	0.14M

4.3 Ablation Study

To evaluate the contributions of MGDC and AAO, we perform ablation studies on the NeRF On-the-go dataset (Ren et al., 2024). All experiments are repeated at least three times to compute the mean and standard deviation for each metric.

Effect of MGDC. We compare three indicator map construction schemes: (1) Global Error, following vanilla FastGS, where the indicator map is constructed using photometric errors over the entire image; (2) Static-only, where photometric errors are computed exclusively within static regions; and (3) MGDC, which applies asymmetric static-transient processing for indicator map construction.

As shown in Tab. 3 and Fig. 6, Static-only yields the worst reconstruction quality with the fewest Gaussians, showing that fully excluding transient regions discards valuable cues for density adjustment. Global Error introduces unreliable reconstruction error statistics and leads to inaccurate

Table 3: Ablation study of the MGDC strategy on two NeRF On-the-go scenes. Best results are shown in bold.

Settings	Mountain					Spot				
	PSNR ↑	SSIM ↑	LPIPS ↓	Time ↓	N_{GS} ↓	PSNR ↑	SSIM ↑	LPIPS ↓	Time ↓	N_{GS} ↓
Global Error	21.17±2.0e-1	0.725±4.9e-3	0.216±4.3e-3	204s±7.7s	0.209M	25.73±7.9e-2	**0.904±8.7e-4**	0.103±6.4e-4	205s±7.8s	0.165M
Static-only	21.12±1.8e-1	0.726±3.4e-3	0.217±2.5e-3	205s±6.1s	**0.207M**	25.78±6.5e-2	0.901±2.2e-4	0.108±1.0e-4	209s±11.7s	**0.134M**
MGDC (Ours)	**21.36±2.3e-2**	**0.728±2.8e-3**	**0.212±2.8e-3**	**195s±5.3s**	0.227M	**25.86±7.5e-2**	**0.904±9.9e-4**	**0.102±9.6e-4**	**197s±1.1s**	0.177M



Input Global Error Static-only MGDC (Ours)

Figure 6: Qualitative results for two NeRF On-the-go scenes. Red arrows mark rendering artifacts from Global Error and Static-only.

Gaussian density control. By contrast, MGDC avoids error contamination from transient objects while preserving densification opportunities within transient regions, achieving the best rendering metrics. Notably, MGDC produces the largest Gaussian count yet achieves the shortest training time, attributed to its improved error-aware density control mechanism.

Effect of AAO. We further ablate the effectiveness of AAO on both RobustSplat (Fu et al., 2025) and our T-FastGS. As reported in Table 4, integrating AAO reduces training time by 24.4% for RobustSplat and 38.9% for T-FastGS. We observe negligible change and even modest gains in reconstruction quality, alongside a slight rise in Gaussian count. These results verify that AAO substantially improves training efficiency without sacrificing reconstruction quality. Additional ablations can be found in the Appendix B .

Table 4: Ablation study of AAO on RobustSplat and T-FastGS using the NeRF On-the-go Dataset.

Settings	PSNR ↑	SSIM ↑	LPIPS ↓	Time ↓	N_{GS} ↓
RobustSplat	23.05 ± 5.5e-2	0.817 ± 7.9e-4	0.149 ± 5.3e-4	11min09s ± 0.7s	**0.98M**
RobustSplat w/ AAO	**23.10 ± 5.7e-2**	0.817 ± 7.4e-4	**0.148 ±2.9e-4**	**8min26s ± 3.2s**	1.00M
T-FastGS w/o AAO	23.10 ± 3.6e-2	0.813 ±6.6e-4	0.158 ±3.1e-4	5min16s ±2.8s	**0.19M**
T-FastGS w/ AAO	**23.14 ±4.6e-2**	0.813 ±1.9e-4	0.158 ±1.3e-4	**3min13s ± 3.0s**	0.20M

5 Conclusion

In this work, we present T-FastGS, a transient-robust and efficient framework for 3D Gaussian Splatting. We demonstrate that error-aware density control degrades in the presence of transient objects, and propose MGDC to generate reliable cues for accurate Gaussian density adjustment. We further identify the timescale mismatch between Gaussian optimization and mask prediction, which motivates our AAO scheme to eliminate redundant mask updates and improve training efficiency. Beyond transient prediction, AAO also suggests a broader design consideration: asynchronous updates can be beneficial when different components in a 3DGS pipeline exhibit mismatched optimization timescales. **Limitations.** DINOv2 feature extraction and iterative optimization for mask prediction introduce computational overhead. Accordingly, T-FastGS cannot fully attain the training speed of vanilla FastGS. Future work will explore highly sparse feature extraction and optimized mask update schedules to further enhance training efficiency.

6 STATEMENTS

AI USE STATEMENT

In this work, we used generative AI tools for English language polishing of the manuscript and pseudocode drafting. We have not used generative AI tools for designing the core algorithm, deriving mathematical formulations, conducting experiments, analyzing experimental results, or writing the main research ideas. Data collection and annotation are not applicable tasks for generative AI in this work. We have reviewed all AI-assisted work. We manually checked the AI-polished text for technical accuracy and consistency; all AI-generated pseudocode and LaTeX snippets were inspected, modified, and validated by the authors. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

To facilitate the reproducibility of our method, we provide anonymous source code and evaluation scripts as supplementary materials. The algorithmic formulation of T-FastGS is described in Section 3 . Additional ablation studies, qualitative results, quantitative results, and hyperparameter settings are provided in the appendix. All metrics and evaluation protocols are fully defined in the experimental section 4 .

REFERENCES

Youyu Chen, Junjun Jiang, Kui Jiang, Xiao Tang, Zhihao Li, Xianming Liu, and Yinyu Nie. Dashgaussian: Optimizing 3d gaussian splatting in 200 seconds. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11146–11155. IEEE, 2025.

Guangchi Fang and Bing Wang. Mini-splatting: Representing scenes with a constrained number of gaussians. In *European conference on computer vision*, pp. 165–181. Springer, 2024.

Guofeng Feng, Siyan Chen, Rong Fu, Zimu Liao, Yi Wang, Tao Liu, Boni Hu, Linning Xu, Zhilin Pei, Hengjie Li, et al. Flashgs: Efficient 3d gaussian splatting for large-scale and high-resolution rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 26652–26662, 2025.

Chuanyu Fu, Yuqi Zhang, Kunbin Yao, Guanying Chen, Yuan Xiong, Chuan Huang, Shuguang Cui, and Xiaochun Cao. Robustsplat: Decoupling densification and dynamics for transient-free 3dgs. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 27126–27136. IEEE, 2025.

Alex Hanson, Allen Tu, Geng Lin, Vasu Singla, Matthias Zwicker, and Tom Goldstein. Speedysplat: Fast 3d gaussian splatting with sparse pixels and sparse primitives. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 21537–21546. IEEE, 2025a.

Alex Hanson, Allen Tu, Vasu Singla, Mayuka Jayawardhana, Matthias Zwicker, and Tom Goldstein. Pup 3d-gs: Principled uncertainty pruning for 3d gaussian splatting. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5949–5958. IEEE, 2025b.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

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*, pp. 1–11, 2024.

Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, George Drettakis, et al. 3d gaussian splatting for real-time radiance field rendering. *ACM Trans. Graph.*, 42(4):139–1, 2023.

Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. In *2023 IEEE/CVF international conference on computer vision (ICCV)*, pp. 3992–4003. IEEE, 2023.

Jonas Kulhanek, Songyou Peng, Zuzana Kukelova, Marc Pollefeys, and Torsten Sattler. Wildgaussians: 3d gaussian splatting in the wild. *arXiv preprint arXiv:2407.08447*, 2024.

Jingyu Lin, Jiaqi Gu, Lubin Fan, Bojian Wu, Yujing Lou, Renjie Chen, Ligang Liu, and Jieping Ye. Hybridgs: Decoupling transients and statics with 2d and 3d gaussian splatting. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 788–797. IEEE, 2025.

Tao Lu, Mulin Yu, Linning Xu, Yuanbo Xiangli, Limin Wang, Dahua Lin, and Bo Dai. Scaffold-gs: Structured 3d gaussians for view-adaptive rendering. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 20654–20664. IEEE, 2024.

Saswat Subhajyoti Mallick, Rahul Goel, Bernhard Kerbl, Markus Steinberger, Francisco Vicente Carrasco, and Fernando De La Torre. Taming 3dgs: High-quality radiance fields with limited resources. In *SIGGRAPH Asia 2024 Conference Papers*, pp. 1–11, 2024.

Alexander Markin, Vadim Pryadilshchikov, Artem Komarichev, Ruslan Rakhimov, Peter Wonka, and Evgeny Burnaev. T-3dgs: Removing transient objects for 3d scene reconstruction. *arXiv preprint arXiv:2412.00155*, 2024.

Maxime Oquab, Timothée Darcet, Theo Moutakanni, Huy V. Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Russell Howes, Po-Yao Huang, Hu Xu, Vasu Sharma, Shang-Wen Li, Wojciech Galuba, Mike Rabbat, Mido Assran, Nicolas Ballas, Gabriel Synnaeve, Ishan Misra, Herve Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. Dinov2: Learning robust visual features without supervision, 2023.

Lukas Radl, Michael Steiner, Mathias Parger, Alexander Weinrauch, Bernhard Kerbl, and Markus Steinberger. StopThePop: Sorted Gaussian Splatting for View-Consistent Real-time Rendering. *ACM Transactions on Graphics*, 43(4), 2024.

Shiwei Ren, Tianci Wen, Yongchun Fang, and Biao Lu. Fastgs: Training 3d gaussian splatting in 100 seconds. *arXiv preprint arXiv:2511.04283*, 2025.

Weining Ren, Zihan Zhu, Boyang Sun, Jiaqi Chen, Marc Pollefeys, and Songyou Peng. Nerf on-the-go: Exploiting uncertainty for distractor-free nerfs in the wild. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 8931–8940. IEEE, 2024.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models, 2021.

Sara Sabour, Suhani Vora, Daniel Duckworth, Ivan Krasin, David J Fleet, and Andrea Tagliasacchi. Robustnerf: Ignoring distractors with robust losses. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 20626–20636. IEEE, 2023.

Sara Sabour, Lily Goli, George Kopanas, Mark Matthews, Dmitry Lagun, Leonidas Guibas, Alec Jacobson, David Fleet, and Andrea Tagliasacchi. Spotlesssplats: Ignoring distractors in 3d gaussian splatting. *ACM Transactions on Graphics*, 44(2):1–11, 2025.

Kangmin Seo, MinKyu Lee, Tae-Young Kim, ByeongCheol Lee, JoonSeoung An, and Jae-Pil Heo. Pdf-gs: Progressive distractor filtering for robust 3d gaussian splatting. *arXiv preprint arXiv:2604.12580*, 2026.

Rui Wang, Quentin Lohmeyer, Mirko Meboldt, and Siyu Tang. Degauss: Dynamic-static decomposition with gaussian splatting for distractor-free 3d reconstruction. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 6294–6303. IEEE, 2025a.

Yihao Wang, Marcus Klasson, Matias Turkulainen, Shuzhe Wang, Juho Kannala, and Arno Solin. Desplat: Decomposed gaussian splatting for distractor-free rendering. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 722–732. IEEE, 2025b.

Zehao Yu, Anpei Chen, Binbin Huang, Torsten Sattler, and Andreas Geiger. Mip-splatting: Alias-free 3d gaussian splatting. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 19447–19456. IEEE, 2024.

A More Details for Our Method

A.1 Transient Mask Prediction

Loss Function. Following SpotLessSplats (Sabour et al., 2025) and RobustSplat (Fu et al., 2025), we supervise the mask predictor using two complementary objectives: a cosine-similarity loss and a dynamically bounded hinge loss.

Given the input feature F_I and rendered feature $F_{\hat{I}}$, we compute their pixel-wise cosine similarity and convert it into a soft static-region target:

$$M_{\text{cos}} = \max\left(2\cos(F_I, F_{\hat{I}}) - 1, 0\right). \tag{17}$$

This transformation maps the similarity values into $[0, 1]$, where $M_{\text{cos}} = 1$ indicates highly consistent regions likely belonging to the static scene, while values below 0.5 are suppressed to zero. The cosine-based supervision is defined as

$$\mathcal{L}_{\text{cos}} = \frac{1}{N} \left\| M_{\text{pred}} - M_{\text{cos}} \right\|_1, \tag{18}$$

where the predicted mask M_{pred} is obtained by Eq. 4 , and N denotes the number of pixels in M_{pred}.

To provide adaptive weak supervision, we estimate dynamic lower and upper bounds from the historical distribution of $\mathcal{L}_1$ residuals between the input image and the rendered image. Specifically, we maintain an exponential moving average of residual histograms and determine two residual thresholds according to the 60% and 80% cumulative probabilities. The corresponding binary masks are generated by incorporating local neighborhood consistency, yielding confidence-aware lower and upper bounds $L^{(t)}$ and $U^{(t)}$ for mask prediction.

and constrain the predicted probabilities within this interval. The bounded hinge loss is formulated as

$$\mathcal{L}_{\text{hinge}} = \max(M_{\text{pred}} - U^{(t)}, 0) + \max(L^{(t)} - M_{\text{pred}}, 0). \tag{19}$$

This constraint only penalizes predictions outside the dynamically estimated interval, while allowing predictions within the interval to remain unconstrained.

The overall objective for training the mask predictor is

$$\mathcal{L}_{\text{mlp}} = \lambda \mathcal{L}_{\text{cos}} + (1 - \lambda)\mathcal{L}_{\text{hinge}}, \tag{20}$$

where λ balances the two terms, and we set $\lambda = 0.5$ during training.

Implementation Details. We use AdamW as the optimizer with a learning rate of 0.01. For feature extraction, we adopt DINOv2 ViT-S/14 (Oquab et al., 2023), which provides an effective balance between feature representation quality and computational efficiency. The extracted 384-dimensional features are fed into a two layer MLP. Specifically, the MLP consists of one hidden layer with 16 neurons followed by a ReLU activation. The final linear layer outputs a scalar value, which is transformed into the predicted mask probability through a Sigmoid function. We set the probability threshold $\tau_b = 0.25$ in Eq. 5 .

Following RobustSplat (Fu et al., 2025), we also employ a coarse-to-fine feature extraction strategy. Specifically, coarse-resolution features are extracted from images resized to 224×224, while fine-resolution features are obtained from images with a resolution of 504×504.

To better capture potential transient boundaries, we further apply a morphological dilation operation with a kernel size of 7 to the predicted masks.

A.2 Gaussian Training

Implementation Details. All experiments are conducted on a single NVIDIA RTX 4090 GPU.

We follow the network architecture and training pipeline of FastGS (Ren et al., 2025) and keep the original hyperparameters unchanged, except for the Gaussian densification start iteration, densification interval, reconstruction error threshold, and gradient magnitude threshold.

During training, the mask predictor starts optimization after 1,500 iterations. To obtain reliable mask estimation before Gaussian density adjustment, we delay Gaussian densification until 5,000 iterations and perform densification and pruning every 100 iterations until 15,000 iterations.

Following the original 3DGS (Kerbl et al., 2023) and FastGS (Ren et al., 2025), we reset Gaussian opacity starting at 3,000 iterations with an interval of 3,000 iterations. After each opacity reset, we temporarily disable mask predictor optimization for the following three epochs to avoid unstable mask updates caused by abrupt changes in Gaussian opacity.

The thresholds for MGDC are set as follows. For T-FastGS, we set the reconstruction error threshold $\tau_e = 0.1$ in Eq. 6 and Eq. 9, and the gradient magnitude threshold to 0.0012. For T-FastGS-Big, these thresholds are adjusted based on scene complexity. Detailed configurations are provided in the released code.

B More Experimental Results

B.1 More Ablations

Effect of MGDC. Complementing Tab. 3, Tab. 5 presents quantitative results on two additional scenes across the three density control variants. All experiments are repeated at least three times to compute the mean and standard deviation for each metric. Overall, MGDC achieves superior reconstruction performance and faster training speed, with only a modest increase in Gaussian count.

Table 5: Ablation study of our MGDC strategy on two additional NeRF On-the-go scenes. Best results are highlighted in bold.

Settings	Fountain					Patio				
	PSNR ↑	SSIM ↑	LPIPS ↓	Time ↓	N_{GS} ↓	PSNR ↑	SSIM ↑	LPIPS ↓	Time ↓	N_{GS} ↓
Global Error	21.03±5.9e-2	0.690±1.1e-3	0.230±9.9e-4	219.6±2.8	**0.260M**	21.33±6.3e-3	**0.826±7.6e-4**	0.145±4.3e-4	192.2±7.2	0.106M
Static-only	20.98±4.7e-2	0.690±8.9e-4	0.229±4.2e-4	214.3±5.3	0.266M	21.27±2.3e-2	**0.826±3.5e-4**	0.146±2.2e-4	197.2±13.9	**0.099M**
MGDC (Ours)	**21.05±5.4e-2**	**0.692±1.5e-4**	**0.227±8.8e-4**	**198.3±7.5**	0.277M	**21.36±3.3e-2**	**0.826±1.6e-2**	**0.144±3.7e-2**	**178.0±3.5**	0.115M

Effect of AAO. Figure 7 plots mask IoU over training epochs on three scenes of NeRF On-the-go dataset. The IoU curves for models with and without AAO follow nearly identical convergence trajectories. Across all scenes, the w/ AAO curve stays slightly above the baseline w/o AAO. This observation suggests that asynchronous alternating optimization does not degrade mask prediction quality, nor slow down convergence. Instead, decoupling mask updates from Gaussian optimization mitigates mutual optimization noise between the two modules. Although the final saturated IoU values are comparable, AAO substantially cuts training time, as quantified in Tab. 4 .

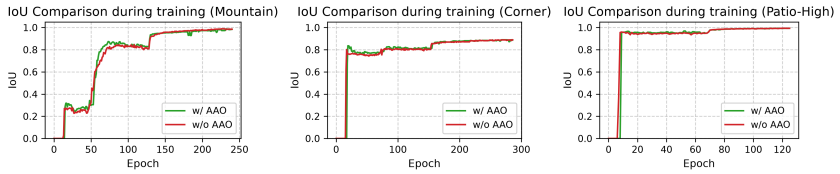


Figure 7: **Mask IoU curves over training epochs on three NeRF On-the-go scenes.** Green: w/ AAO; Red: w/o AAO. The curves closely track each other across all scenes, showing that AAO preserves mask quality while reducing training computation. Minor consistent gains in IoU can be observed for w/ AAO, indicating that asynchronous optimization alleviates mutual interference between Gaussian optimization and mask learning.

Table 6: Ablation study comparing two feature extractors, ResNet18 and DINOv2 ViT-S/14, on three scenes from the NeRF On-the-go dataset. The best results are highlighted in bold.

Settings	Mountain					Fountain					Patio				
	PSNR ↑	SSIM ↑	LPIPS ↓	Time ↓	N_{GS} ↓	PSNR ↑	SSIM ↑	LPIPS ↓	Time ↓	N_{GS} ↓	PSNR ↑	SSIM ↑	LPIPS ↓	Time ↓	N_{GS} ↓
ResNet18 (He et al., 2016)	21.28	0.723	0.216	**3min10s**	**0.22M**	20.87	0.686	0.231	**2min51s**	**0.27M**	21.25	0.824	0.146	**2min39s**	**0.11M**
DINOv2 ViT-S/14 (Ours)	**21.36**	**0.728**	**0.212**	3min15s	0.23M	**21.05**	**0.692**	**0.227**	3min18s	0.28M	**21.36**	**0.826**	**0.144**	2min58s	0.12M

Effect of Feature Extractor. We further conduct ablations to compare the performance of ResNet18 (He et al., 2016) and DINOv2 ViT-S/14 (Oquab et al., 2023) as our feature extraction backbone. As shown in Tab. 6, although ResNet18 achieves slightly faster training speed and fewer Gaussian primitives, DINOv2 consistently yields superior reconstruction performance across all scenes in PSNR, SSIM, and LPIPS.

B.2 MORE EVALUTIONS

Breakdown of Computational Overhead. Compared with vanilla FastGS (Ren et al., 2025), T-FastGS incurs extra runtime overhead from feature extraction and MLP training for transient mask prediction. To identify the sources of this overhead, we decompose the total training time and quantify the time consumed by these two components separately.

As shown in Tab. 7, vanilla FastGS achieves the fastest training speed. Equipping FastGS with DINOv2 feature extraction introduces approximately **30** seconds of extra runtime across different scenes, and the incorporated MLP mask predictor further brings around **14** seconds of additional overhead, even with our AAO scheme. These results indicate that feature extraction and mask inference constitute the major computational bottlenecks of T-FastGS. This motivates us to explore highly sparse feature extraction and optimized mask update schedules to further enhance training efficiency in future work.

Table 7: Runtime breakdown of feature extraction and transient mask prediction. All models are trained with 30k iterations. Note that both FastGS + DINOv2 and our T-FastGS adopt the proposed AAO strategy.

Settings	Mountain	Corner	Spot
FastGS (Ren et al., 2025)	146s	143s	151s
FastGS + DINOv2	181s	160s	183s
FastGS + DINOv2 + MLP (T-FastGS)	195s	183s	197s

More Evaluations on Two Benchmark Datasets. Table 8 presents quantitative results on three additional scenes from the NeRF On-the-go dataset. Table 9 further reports the overall average metrics of our method and baseline approaches across both NeRF On-the-go and RobustNeRF datasets. Extensive qualitative comparisons are also provided for comprehensive visual validation. As show in Fig. 8 and Fig. 9, T-FastGS produces transient-robust reconstructions with fine details.

Table 8: Quantitative comparison on three additional NeRF On-the-go scenes. The best results are highlighted in bold, and the second in underline.

Method	Fountain					Patio					Patio-High				
	PSNR ↑	SSIM ↑	LPIPS ↓	Time ↓	N_{GS} ↓	PSNR ↑	SSIM ↑	LPIPS ↓	Time ↓	N_{GS} ↓	PSNR ↑	SSIM ↑	LPIPS ↓	Time ↓	N_{GS} ↓
3DGS (Kerbl et al., 2023)	19.58	0.677	0.216	12min22s	2.29M	17.50	0.715	0.230	7min09s	1.05M	17.09	0.655	0.317	9min46s	1.99M
FastGS (Ren et al., 2025)	20.67	0.680	0.235	2min31s	0.28M	17.86	0.729	0.227	2min38s	0.13M	19.32	0.728	0.259	2min28s	0.27M
SpotLessSplats (Sabour et al., 2025)	20.68	0.640	0.223	25min12s	1.56M	21.33	0.823	0.171	20min51s	0.66M	21.69	0.743	0.237	25min10s	1.01M
T-3DGS (Markin et al., 2024)	20.75	0.681	0.212	34min23s	3.51M	21.17	**0.827**	0.143	27min01s	1.01M	22.13	0.820	0.172	28min59s	1.78M
RobustSplat (Fu et al., 2025)	21.02	**0.702**	0.199	13min30s	1.87M	**21.61**	**0.827**	0.140	9min55s	0.58M	22.71	0.836	0.145	11min12s	1.05M
PDF-GS (Seo et al., 2026)	20.96	0.683	0.208	13min46s	1.75M	21.46	0.824	0.145	9min13s	0.53M	22.61	0.828	0.157	11min02	0.93M
T-FastGS (Ours)	21.05	0.692	0.227	**3min18s**	**0.28M**	21.36	0.826	0.144	**2min58s**	**0.12M**	**22.77**	0.835	0.151	**3min27s**	**0.27M**
T-FastGS-Big (Ours)	**21.09**	0.700	**0.197**	4min02s	0.58M	21.24	**0.827**	**0.132**	3min19s	0.20M	22.72	**0.837**	**0.143**	3min53s	0.48M

Table 9: Quantitative average comparison on NeRF On-the-go and RobustNeRF datasets. The best results are highlighted in bold, and the second in underline.

Method	NeRF On-the-go					RobustNeRF				
	PSNR ↑	SSIM ↑	LPIPS ↓	Time ↓	N_{GS} ↓	PSNR ↑	SSIM ↑	LPIPS ↓	Time ↓	N_{GS} ↓
3DGS (Kerbl et al., 2023)	19.14	0.716	0.246	9min15s	1.64M	25.88	0.862	0.168	6min03s	0.82M
FastGS (Ren et al., 2025)	20.82	0.754	0.220	2min30s	0.20M	27.98	0.884	0.155	2min32s	0.13M
SpotLessSplats (Sabour et al., 2025)	22.36	0.754	0.206	23min42s	0.85M	28.90	0.875	0.162	25min24s	0.45M
T-3DGS (Markin et al., 2024)	22.34	0.800	0.167	29min05s	1.74M	27.95	0.884	0.155	26min48s	0.80M
RobustSplat (Fu et al., 2025)	23.05	**0.817**	0.149	11min08s	0.98M	**29.41**	**0.895**	0.137	11min04s	0.52M
PDF-GS (Seo et al., 2026)	23.02	0.810	0.157	10min54s	0.92M	29.32	0.893	0.135	9min52s	0.53M
T-FastGS (Ours)	**23.14**	0.813	0.158	**3min13s**	**0.20M**	29.34	0.894	0.142	**3min37s**	**0.13M**
T-FastGS-Big (Ours)	23.04	0.816	**0.143**	3min49s	0.36M	29.27	0.894	**0.132**	3min51s	0.22M

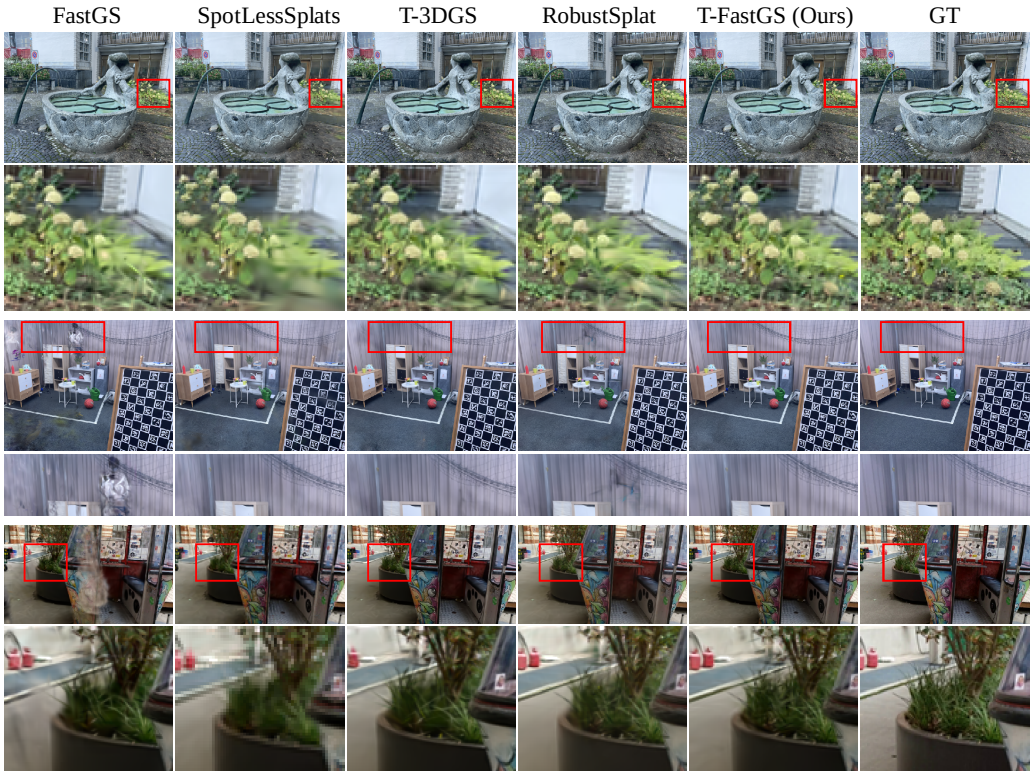


Figure 8: Qualitative results on other scenes from the NeRF On-the-go dataset.

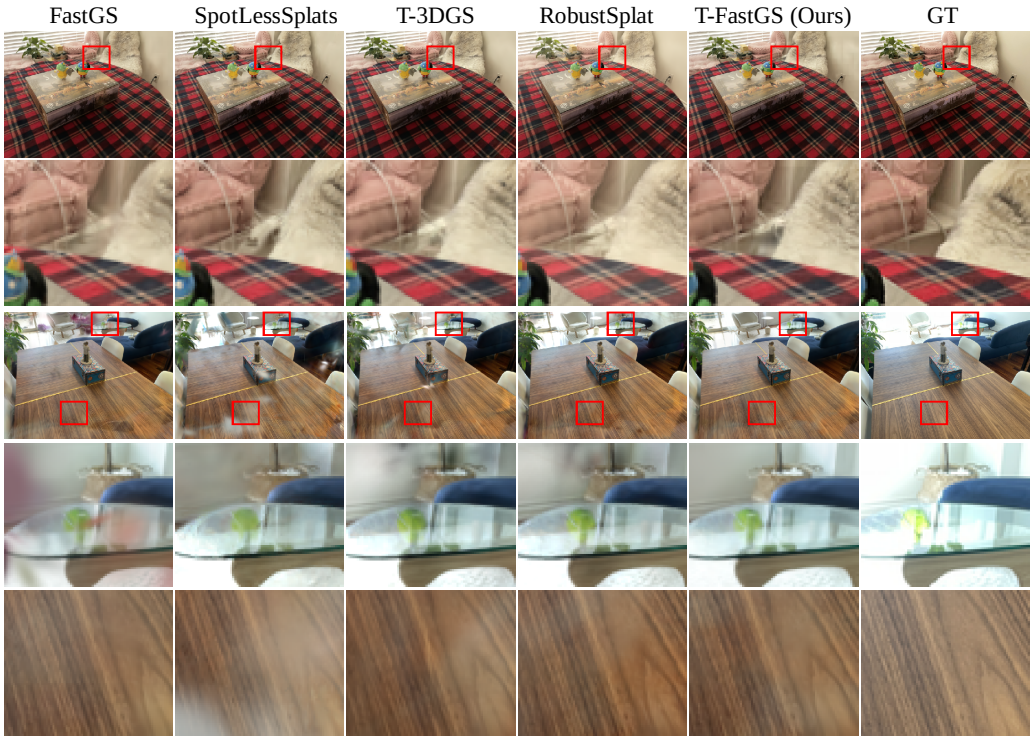


Figure 9: Qualitative results on other scenes from the RobustNeRF dataset.