CAGE-GS: LOW-RESOLUTION PRIOR-GUIDED GAUSSIAN EXPANSION CONTROL AGAINST COMPUTATION COST ATTACKS ON 3D GAUSSIAN SPLATTING

Anonymous authors
Paper under double-blind review

ABSTRACT

Computational cost attacks against 3D Gaussian Splatting (3DGS) can substantially increase the memory and training overhead of 3DGS optimization, potentially causing denial-of-service (DoS). Existing defenses mainly rely on image purification or frequency filtering. However, because adversarial perturbations overlap with legitimate high-frequency scene details, aggressive input processing can suppress valid textures while leaving abnormal Gaussian geometry and spatial congestion insufficiently constrained. Motivated by the reduced effectiveness of such attacks at low resolution, we propose CAGE-GS, a coarse-to-fine defense that combines mild wavelet denoising with low-resolution reconstruction to derive structural and density priors. During full-resolution optimization, these priors reject structurally implausible Gaussian candidates and suppress excessive local expansion. Extensive experiments across three benchmark datasets show that CAGE-GS restores computational overhead toward clean-training levels while preserving reconstruction fidelity, and achieves better clean-input reconstruction quality than existing defense methods.

1 INTRODUCTION

Recovering 3D geometry and appearance from multi-view images is a fundamental computer vision problem with applications in novel-view synthesis, autonomous systems, robotics, and immersive media Schonberger & Frahm (2016); Yao et al. (2018); Mildenhall et al. (2021); Kerbl et al. (2023). Neural Radiance Fields (NeRFs) Mildenhall et al. (2021) achieve high-quality neural rendering but require expensive dense ray sampling and network inference. 3D Gaussian Splatting (3DGS) Kerbl et al. (2023) instead uses explicit anisotropic Gaussian primitives, enabling efficient optimization and real-time high-fidelity rendering.

A key capability of 3DGS is adaptive densification, which clones or splits Gaussians in under-reconstructed regions and prunes low-contribution primitives. While this mechanism improves reconstruction fidelity, it also creates a computational attack surface. Poison-Splat Lu et al. (2025) injects imperceptible perturbations that repeatedly trigger unnecessary densification, causing excessive growth in Gaussian count, GPU memory, and training time and potentially threatening system availability.

Existing defenses mitigate such resource growth primarily through image purification or frequency filtering Li et al. (2026); Qiao et al. (2026). Although these approaches can effectively suppress the attack, their robustness relies mainly on processing the 2D inputs. Because adversarial perturbations may overlap with genuine high-frequency textures, strong input processing can remove useful scene details and degrade reconstruction quality on benign inputs. Moreover, image-centric defenses do not directly exploit the abnormal Gaussian attributes and spatial congestion through which the attack ultimately manifests.

Prior work has shown that image resizing can mitigate adversarial perturbations in 2D vision Xie et al. (2018). Similarly, we find that reducing the input resolution substantially narrows the structural and density discrepancies between poisoned and clean 3DGS reconstructions. As illustrated in Figure 1, poisoned Gaussian fields at full resolution deviate from their clean counterparts, particularly in Gaussian count and geometric attributes such as position, size, and shape. At half resolution,

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.

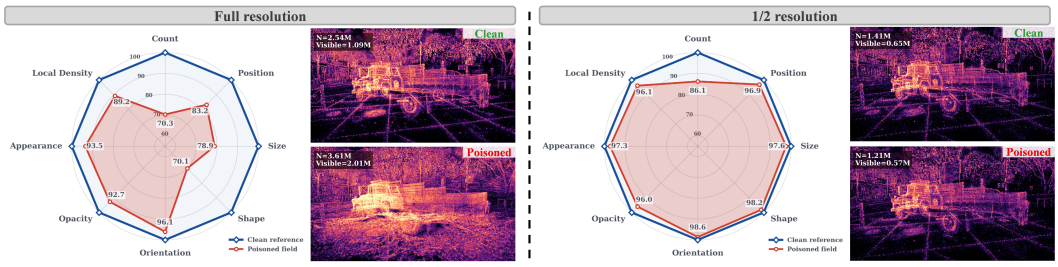

Figure 1: Motivating observation on the *Truck* scene from Tanks & Temples. At full resolution (left), poisoning causes substantial deviations from the clean Gaussian field in Gaussian count and geometric and density statistics. At half resolution, however, these discrepancies are substantially reduced: the similarity scores for position, size, shape, and local density all exceed 96%, while the count similarity increases from 70.3% to 86.1%.

however, these discrepancies are significantly reduced: the similarity scores for position, size, shape, and local density all exceed (96%), while the Gaussian count and spatial distribution also approach those of the clean reconstruction. This resolution-dependent behavior suggests that low-resolution reconstruction retains two complementary forms of scene-specific knowledge: a *structural prior* describing plausible Gaussian attributes and a *density prior* describing normal spatial occupancy; additional scene-level evidence is provided in Appendix C.

Motivated by this observation, we propose **CAGE-GS**, a coarse-to-fine defense that combines mild wavelet denoising with low-resolution structural and density priors. During full-resolution optimization, these priors jointly reject structurally implausible candidates and regulate local Gaussian congestion, suppressing attack-induced expansion while preserving benign reconstruction fidelity.

Our main contributions are:

- We identify that poisoned 3DGS reconstructions become substantially more clean-like at low resolution, providing reliable scene-specific structural and density priors.
- We propose CAGE-GS, a coarse-to-fine defense that uses these priors to regulate Gaussian expansion directly in the 3D representation.
- Experiments on three benchmarks show that CAGE-GS restores computational costs toward clean levels while preserving better clean-input reconstruction quality than the compared defenses.

2 Related Work

2.1 3D Gaussian Splatting and Applications

3D Gaussian Splatting (3DGS) Kerbl et al. (2023) has rapidly developed from a novel-view synthesis representation into a framework for 3D scene modeling. Its explicit primitives and efficient rasterization have supported dynamic scene reconstruction Wu et al. (2024a); Yang et al. (2024); Wang et al. (2025); Wu et al. (2025), large-scale and autonomous-driving scene modeling Lin et al. (2024); Zhou et al. (2024); Liu et al. (2026), and dense visual SLAM Keetha et al. (2024); Matsuki et al. (2024); Su et al. (2026). Gaussian representations have also been extended to language-aware scenes understanding Qin et al. (2024), interactive and controllable editing Huang et al. (2024); Wu et al. (2024b); Wen et al. (2025), human avatar reconstruction Hu et al. (2024), and efficient 3D content generation Tang et al. (2024). Their increasing adoption across interactive, large-scale, and safety-related applications makes the security and computational reliability of Gaussian-based systems increasingly important.

2.2 Security Threats and Defenses for 3DGS

Beyond conventional integrity attacks that corrupt model predictions or outputs, computation-cost attacks threaten system availability by amplifying resource consumption. Representative examples

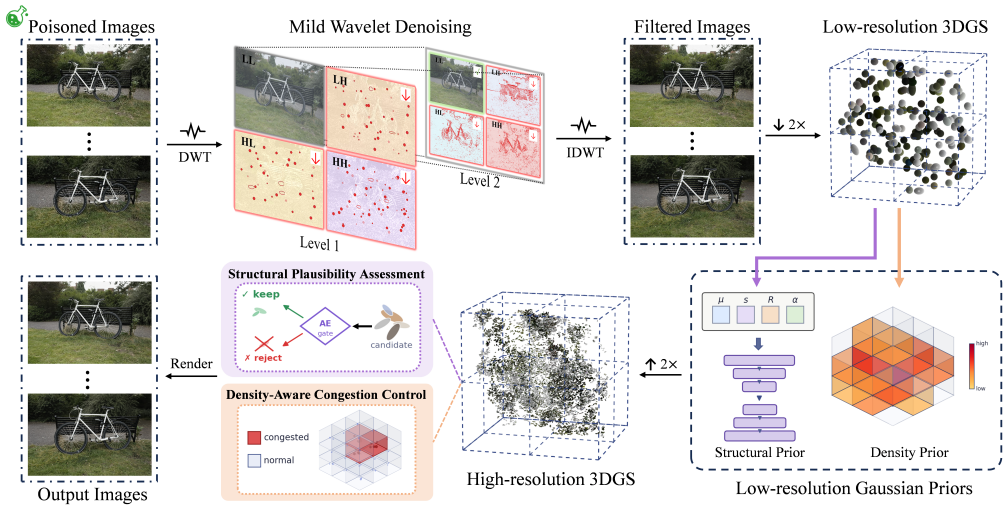

Figure 2: Overview of CAGE-GS. Mild wavelet denoising first attenuates image-space perturbations, followed by low-resolution reconstruction to obtain Gaussian structure and density priors. During full-resolution optimization, structural plausibility assessment rejects anomalous candidates, while density-aware congestion control suppresses excessive local expansion.

include Sponge Examples Shumailov et al. (2021), DeepSloth Hong et al. (2021), and Phantom Sponges Shapira et al. (2023), which increase inference cost or overload post-processing and may ultimately cause DoS.

Recent studies have exposed similar security risks in Gaussian-based reconstruction. CLOAK and DAGGER Hull et al. (2025) manipulate view-dependent appearance or Gaussian parameters to deceive downstream perception models, while Gaussian Splitting Attack Meng et al. (2025) generates multi-view adversarial perturbations through Gaussian manipulation. Other poisoning attacks target reconstructed content, geometry, or initialization Hong et al. (2025); Ke et al. (2025); Wadekar et al. (2026); Wang et al. (2026). Unlike these integrity-oriented attacks, Poison-Splat Lu et al. (2025) compromises computational availability by inducing excessive densification, thereby increasing Gaussian count, GPU memory consumption, and training time, potentially leading to denial of service (DoS) under severe resource exhaustion.

To mitigate this threat, RemedyGS Li et al. (2026) detects and purifies suspicious training views, whereas DefenseSplat Qiao et al. (2026) applies frequency-aware filtering before Gaussian optimization. Although these image-space defenses suppress malicious resource growth, input purification may also remove legitimate textures and fine structures. Balancing computational robustness with benign reconstruction fidelity therefore remains an open challenge.

3 Preliminaries

3.1 3D Gaussian Splatting.

3D Gaussian Splatting (3DGS) Kerbl et al. (2023) represents a scene using a set of anisotropic Gaussian primitives $\mathcal{G} = \{g_i\}_{i=1}^{N}$, with each Gaussian parameterized as $g_i = (\boldsymbol{\mu}_i, \mathbf{s}_i, \mathbf{q}_i, \alpha_i, \mathbf{c}_i)$. Here, $\boldsymbol{\mu}_i$, $\mathbf{s}_i$, and $\mathbf{q}_i$ denote its center, scale, and rotation, respectively, while α_i and $\mathbf{c}_i$ denote its opacity and spherical-harmonic appearance coefficients. Given a camera view, the Gaussians are projected onto the image plane and sorted by depth. The color of pixel $\mathbf{p}$ is computed through front-to-back alpha compositing:

$$C(\mathbf{p}) = \sum_{i=1}^{K} T_i(\mathbf{p})\, \alpha_i(\mathbf{p})\, \mathbf{c}_i(\mathbf{d}), \;\; T_i(\mathbf{p}) = \prod_{j=1}^{i-1} \left(1 - \alpha_j(\mathbf{p})\right). \tag{1}$$

where $\alpha_i(\mathbf{p})$ is the effective opacity of Gaussian i at $\mathbf{p}$, and $\mathbf{c}_i(\mathbf{d})$ is its view-dependent color. The Gaussian parameters are optimized by minimizing a weighted combination of the photometric L_1

loss and the D-SSIM loss between the rendered image $\hat{\mathbf{I}}$ and its ground-truth image $\mathbf{I}$:

$$\mathcal{L} = (1-\lambda)\,\mathcal{L}_1(\mathbf{I},\hat{\mathbf{I}}) + \lambda\,\mathcal{L}_{\text{D-SSIM}}(\mathbf{I},\hat{\mathbf{I}}), \tag{2}$$

where λ balances pixel-level fidelity and structural similarity. During optimization, 3DGS adaptively clones or splits Gaussians with large view-space positional gradients and prunes low-contribution primitives. This densification mechanism improves reconstruction fidelity but also creates an attack surface for malicious Gaussian expansion.

3.2 Poison-Splat Attack.

Poison-Splat Lu et al. (2025) is a computation cost attack that injects imperceptible perturbations into multi-view training images to induce excessive Gaussian densification. Let $\mathcal{V}_{\text{poi}}$ denote the poisoned multi-view training images constructed by the attacker. Poison-Splat can be formulated as the following bi-level optimization:

$$\mathcal{V}_{\text{poi}} = \arg\max_{\mathcal{V}_{\text{poi}}} \mathcal{C}(\mathcal{G}^*),\ \text{s.t.}\ \mathcal{G}^* = \arg\min_{\mathcal{G}} \mathcal{L}(\mathcal{V}_{\text{poi}}), \tag{3}$$

where $\mathcal{C}$ denotes the computational cost. Since resource consumption is not directly differentiable, Poison-Splat uses the number of optimized Gaussians as a surrogate objective. It increases image non-smoothness through a total-variation-based objective, thereby amplifying positional gradients and repeatedly triggering Gaussian cloning and splitting. The resulting expansion substantially increases GPU memory consumption and training time, and can eventually cause denial-of-service.

4 Method

Figure 2 presents an overview of CAGE-GS, which consists of three main stages: mild wavelet denoising, low-resolution prior construction, and prior-guided high-resolution Gaussian expansion control.

4.1 Mild Wavelet Denoising

To attenuate image-space perturbations, we apply a two-level 2D DWT to each input image, yielding a second-level approximation subband and six detail subbands. Let $\mathbf{L}$ denote the approximation subband and $\mathbf{H}_\ell$ denote the ℓ-th detail subband, where $\ell \in \{1,\dots,6\}$. We apply element-wise soft thresholding to each detail subband:

$$\widetilde{\mathbf{H}}_\ell = \text{sign}(\mathbf{H}_\ell)\max\left(|\mathbf{H}_\ell| - \tau_\ell, 0\right), \tag{4}$$

where

$$\tau_\ell = \rho\frac{Q_{0.5}\left(|\mathbf{H}_\ell|\right)}{0.6745}\sqrt{2\log N_\ell}. \tag{5}$$

Here, $Q_p(\cdot)$ denotes the empirical p-quantile, so $Q_{0.5}(|\mathbf{H}_\ell|)$ is the median absolute coefficient in $\mathbf{H}_\ell$, N_ℓ is the number of coefficients in $\mathbf{H}_\ell$, and ρ controls the denoising strength. Finally, we apply the inverse DWT to $\mathbf{L}$ and the thresholded detail subbands $\{\widetilde{\mathbf{H}}_\ell\}_{\ell=1}^{6}$ to reconstruct the denoised image.

4.2 Low-Resolution Gaussian Priors

Using the wavelet-denoised images, we first perform Gaussian reconstruction at half resolution by optimizing a coarse Gaussian set $\mathcal{G}_z = \{g_z\}_{z=1}^{N}$. The resulting coarse Gaussian representation is then used to construct the structural and density priors.

4.2.1 Structural Prior.

Since attack-induced densification is more directly reflected in Gaussian geometry, we construct a structural feature vector for each coarse Gaussian:

$$\mathbf{x}_z = \left[\frac{\boldsymbol{\mu}_z - \mathbf{o}}{r},\ \log\mathbf{s}_z,\ \bar{\mathbf{q}}_z,\ \text{logit}(\alpha_z),\ \eta_z\right] \in \mathbb{R}^{12}, \tag{6}$$

Table 1: Training statistics under clean, poisoned, and defended settings on selected scenes from Tanks and Temples (TT), NeRF-Synthetic (NS), and Mip-NeRF 360 (MIP). Red and blue subscripts denote poisoned-to-clean increases and poisoned-to-defended reductions, respectively.

	Max. Gaussians (M) ↓			Peak GPU Memory (MB) ↓			Training Time (min) ↓		
Scene	**Clean**	**Poisoned**	**Ours**	**Clean**	**Poisoned**	**Ours**	**Clean**	**Poisoned**	**Ours**
TT-Auditorium	0.716	$2.679_{3.74\times\uparrow}$	$0.625_{4.29\times\downarrow}$	4447	$14821_{3.33\times\uparrow}$	$6441_{2.30\times\downarrow}$	12.4	$23.2_{1.86\times\uparrow}$	$12.8_{1.81\times\downarrow}$
TT-Courtroom	2.945	$5.454_{1.85\times\uparrow}$	$2.208_{2.47\times\downarrow}$	9015	$14119_{1.57\times\uparrow}$	$6887_{2.05\times\downarrow}$	21.9	$33.5_{1.53\times\uparrow}$	$19.3_{1.73\times\downarrow}$
TT-Truck	2.539	$3.614_{1.42\times\uparrow}$	$1.522_{2.37\times\downarrow}$	7533	$9775_{1.30\times\uparrow}$	$5333_{1.83\times\downarrow}$	19.5	$27.8_{1.42\times\uparrow}$	$15.8_{1.76\times\downarrow}$
TT Avg.	**1.842**	$\mathbf{3.115}_{1.69\times\uparrow}$	$\mathbf{1.270}_{2.45\times\downarrow}$	**6977**	$\mathbf{10536}_{1.51\times\uparrow}$	$\mathbf{5827}_{1.81\times\downarrow}$	**17.7**	$\mathbf{24.5}_{1.39\times\uparrow}$	$\mathbf{15.9}_{1.55\times\downarrow}$
NS-drums	0.385	$0.692_{1.80\times\uparrow}$	$0.328_{2.11\times\downarrow}$	4259	$7037_{1.65\times\uparrow}$	$3503_{2.01\times\downarrow}$	8.1	$12.3_{1.52\times\uparrow}$	$9.7_{1.26\times\downarrow}$
NS-hotdog	0.184	$1.143_{6.21\times\uparrow}$	$0.274_{4.17\times\downarrow}$	4171	$29181_{7.00\times\uparrow}$	$4157_{7.02\times\downarrow}$	7.2	$19.9_{2.76\times\uparrow}$	$9.4_{2.11\times\downarrow}$
NS-ship	0.273	$1.066_{3.90\times\uparrow}$	$0.367_{2.90\times\downarrow}$	4297	$15953_{3.71\times\uparrow}$	$4925_{3.24\times\downarrow}$	8.2	$18.8_{2.29\times\uparrow}$	$11.2_{1.68\times\downarrow}$
NS Avg.	**0.290**	$\mathbf{0.717}_{2.47\times\uparrow}$	$\mathbf{0.289}_{2.48\times\downarrow}$	**4224**	$\mathbf{10354}_{2.45\times\uparrow}$	$\mathbf{3950}_{2.62\times\downarrow}$	**7.5**	$\mathbf{13.3}_{1.77\times\uparrow}$	$\mathbf{9.5}_{1.40\times\downarrow}$
MIP-bicycle	5.917	$10.248_{1.73\times\uparrow}$	$4.557_{2.25\times\downarrow}$	16449	$26735_{1.63\times\uparrow}$	$14705_{1.82\times\downarrow}$	45.4	$63.6_{1.40\times\uparrow}$	$37.7_{1.69\times\downarrow}$
MIP-bonsai	1.296	$6.104_{4.71\times\uparrow}$	$1.077_{5.67\times\downarrow}$	9097	$19975_{2.20\times\uparrow}$	$9367_{2.13\times\downarrow}$	22.8	$43.4_{1.90\times\uparrow}$	$21.4_{2.02\times\downarrow}$
MIP-garden	6.481	$7.835_{1.21\times\uparrow}$	$3.347_{2.34\times\downarrow}$	18091	$21305_{1.18\times\uparrow}$	$11065_{1.93\times\downarrow}$	49.1	$60.2_{1.23\times\uparrow}$	$33.1_{1.82\times\downarrow}$
MIP Avg.	**3.356**	$\mathbf{7.073}_{2.11\times\uparrow}$	$\mathbf{2.631}_{2.69\times\downarrow}$	**12553**	$\mathbf{22798}_{1.82\times\uparrow}$	$\mathbf{11633}_{1.96\times\downarrow}$	**33.6**	$\mathbf{53.1}_{1.58\times\uparrow}$	$\mathbf{30.2}_{1.76\times\downarrow}$

where $\mathbf{o}$ and r denote the center of the scene and the extent of the scene estimated from the coarse Gaussian model, respectively. $\bar{\mathbf{q}}_z$ is the quaternion with a non-negative scalar component, the anisotropy term η_z is defined as:

$$\eta_z = \max_k \log s_{z,k} - \min_k \log s_{z,k}, \quad \text{s.t. } k = 1, 2, 3. \tag{7}$$

It measures the logarithmic scale anisotropy across the three principal axes. After standardizing each feature dimension, we train an autoencoder with encoder $\mathcal{E}$ and decoder $\mathcal{D}$ on the coarse Gaussian features using the reconstruction loss:

$$\mathcal{L}_{\text{AE}} = \frac{1}{N} \sum_{z=1}^{N} \|\mathbf{x}_z - \mathcal{D}(\mathcal{E}(\mathbf{x}_z))\|_2^2 . \tag{8}$$

The reconstruction error of each coarse Gaussian is computed as:

$$r_z = \|\mathbf{x}_z - \mathcal{D}(\mathcal{E}(\mathbf{x}_z))\|_2^2 . \tag{9}$$

We define the structural acceptance threshold as:

$$\tau_{\text{str}} = Q_{0.85}\left(\{r_z\}_{z=1}^{N}\right). \tag{10}$$

The trained autoencoder and the associated threshold τ_{str} serve as a fixed coarse structural prior, providing a reference for assessing candidate Gaussians during subsequent high-resolution densification.

4.2.2 Density Prior.

To characterize the spatial density distribution of the coarse reconstruction, we partition the scene into uniform voxels with side length v. Each Gaussian center is assigned to a discrete voxel index:

$$\mathbf{h}_z = \text{round}\left(\frac{\boldsymbol{\mu}_z}{v}\right) \in \mathbb{Z}^3, \tag{11}$$

where $\boldsymbol{\mu}_z$ is the center of the z-th Gaussian. Let M denote the number of occupied voxels, indexed by $m \in \{1, \ldots, M\}$, and let o_m denote the occupancy of the m-th voxel, it can be defined as:

$$o_m = |\{z : \mathbf{h}_z = \mathbf{j}_m\}| . \tag{12}$$

where $\mathbf{j}_m \in \mathbb{Z}^3$ denotes the discrete spatial coordinate of the m-th voxel. We define the occupancy prior factor for voxel m as:

$$a_m = \min\left(\frac{o_m}{Q_{0.5}\left(\{o_m\}_{m=1}^{M}\right)}, 1\right). \tag{13}$$

Table 2: Generalization of CAGE-GS to Mip-Splatting Yu et al. (2024) on selected NeRF-Synthetic scenes. Metrics and colored subscripts follow Table 1.

	Max. Gaussians (M) ↓			Peak GPU Memory (MB) ↓			Training Time (min) ↓		
Scene	**Clean**	**Poisoned**	**Ours**	**Clean**	**Poisoned**	**Ours**	**Clean**	**Poisoned**	**Ours**
NS-chair	0.381	$1.154_{3.03\times\uparrow}$	$0.323_{3.57\times\downarrow}$	4230	$32342_{7.65\times\uparrow}$	$5962_{5.42\times\downarrow}$	14.9	$18.7_{1.26\times\uparrow}$	$10.8_{1.73\times\downarrow}$
NS-drums	0.411	$0.736_{1.79\times\uparrow}$	$0.305_{2.41\times\downarrow}$	5084	$17798_{3.50\times\uparrow}$	$4986_{3.57\times\downarrow}$	10.5	$15.5_{1.48\times\uparrow}$	$10.5_{1.48\times\downarrow}$
NS-ficus	0.215	$0.486_{2.26\times\uparrow}$	$0.148_{3.28\times\downarrow}$	3562	$18836_{5.29\times\uparrow}$	$3172_{5.94\times\downarrow}$	7.9	$13.2_{1.67\times\uparrow}$	$9.4_{1.40\times\downarrow}$
NS-materials	0.168	$0.505_{3.01\times\uparrow}$	$0.188_{2.68\times\downarrow}$	2814	$9888_{3.51\times\uparrow}$	$3608_{2.74\times\downarrow}$	8.3	$12.1_{1.46\times\uparrow}$	$9.0_{1.34\times\downarrow}$
NS-mic	0.273	$0.595_{2.18\times\uparrow}$	$0.220_{2.70\times\downarrow}$	3454	$24810_{7.18\times\uparrow}$	$5276_{4.70\times\downarrow}$	10.9	$16.0_{1.47\times\uparrow}$	$10.8_{1.49\times\downarrow}$
NS Avg.	**0.290**	$\mathbf{0.695}_{2.40\times\uparrow}$	$\mathbf{0.237}_{2.93\times\downarrow}$	**3829**	$\mathbf{20735}_{5.42\times\uparrow}$	$\mathbf{4601}_{4.51\times\downarrow}$	**10.5**	$\mathbf{15.1}_{1.44\times\uparrow}$	$\mathbf{10.1}_{1.50\times\downarrow}$

Table 3: Clean-input reconstruction quality. Bold and underlined entries denote the best and second-best results among defenses, respectively. Original 3DGS is shown as an undefended reference and excluded from the ranking.

		Mip-NeRF 360			NeRF-Synthetic			Tanks & Temples		
Method	**Source**	**PSNR ↑**	**SSIM ↑**	**LPIPS ↓**	**PSNR ↑**	**SSIM ↑**	**LPIPS ↓**	**PSNR ↑**	**SSIM ↑**	**LPIPS ↓**
3DGS	Original	29.51	0.863	0.196	36.98	0.982	0.024	24.77	0.857	0.176
Gaussian Blur	Naive	<u>28.30</u>	<u>0.798</u>	<u>0.260</u>	<u>32.21</u>	<u>0.960</u>	0.063	24.05	<u>0.806</u>	<u>0.259</u>
DefenseSplat	ECCV 2026	27.11	0.733	0.327	27.64	0.912	0.098	23.09	0.745	0.335
RemedyGS	CVPR 2026	27.21	0.767	0.274	28.12	0.928	<u>0.051</u>	<u>24.10</u>	0.803	0.262
CAGE-GS	**Ours**	**28.83**	**0.839**	**0.229**	**36.80**	**0.981**	**0.024**	**24.35**	**0.842**	**0.206**

The factor a_m measures the coarse occupancy of voxel m relative to the median occupancy. A larger a_m assigns a larger Gaussian occupancy budget to voxel m during full-resolution densification. The corresponding voxel-wise occupancy budget is:

$$L_m = \begin{cases} b, & o_m = 0, \\ o_m + \kappa\sqrt{o_m + 1} + a_m o_m + b & o_m > 0, \end{cases} \tag{14}$$

where b is the base capacity and κ controls the tolerance for local density variation. The resulting voxel-wise budgets L_m serve as a fixed coarse density prior, providing spatial references for regulating Gaussian densification during subsequent high-resolution optimization.

4.3 Prior-Guided High-Resolution Gaussian Expansion Control

After the coarse stage, we retain the Gaussian parameters and continue optimizing the same Gaussian set using full-resolution wavelet-denoised views, while keeping the learned autoencoder fixed. During high-resolution optimization, each clone or split candidate Gaussian $\hat{g}$ is assigned structural and density anomaly scores with respect to the coarse priors, measuring the plausibility of its Gaussian attributes and the excess occupancy of its corresponding voxel, respectively.

4.3.1 Structural Plausibility Assessment.

For every clone or split candidate $\hat{g}$, we construct its standardized feature vector $\mathbf{x}(\hat{g})$ using the same representation as Equation 6. The structural anomaly score is computed as the autoencoder reconstruction error:

$$r_{\text{str}}(\hat{g}) = \|\mathbf{x}(\hat{g}) - \mathcal{D}(\mathcal{E}(\mathbf{x}(\hat{g})))\|_2^2 . \tag{15}$$

Candidates with large reconstruction errors deviate from the distribution of coarse Gaussian attributes and are therefore regarded as structurally implausible.

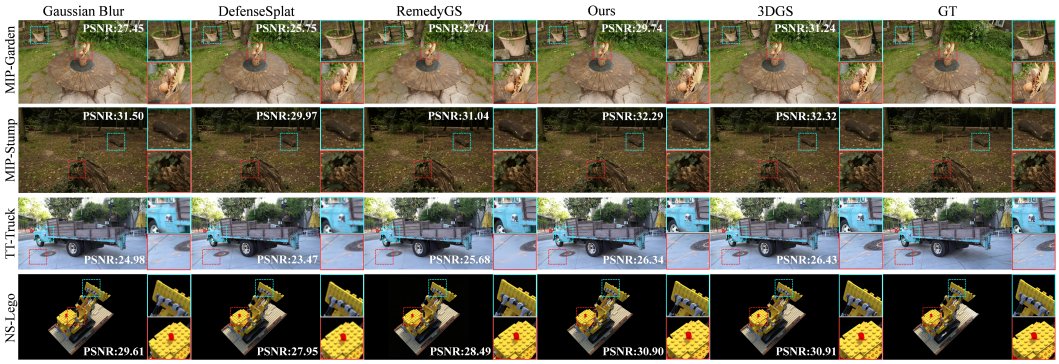

Figure 3: Qualitative comparison under clean inputs. Compared with existing defenses, CAGE-GS better preserves fine textures and structural details, producing reconstructions visually closer to the original 3DGS results and the ground truth across both real and synthetic scenes.

4.3.2 Density-Aware Congestion Control.

Let n_m denote the current number of high-resolution Gaussians in voxel m. For a candidate Gaussian $\hat{g}$ located in voxel m, we measure the density anomaly score as:

$$r_{\text{den}}(\hat{g}) = \max\left(\frac{n_m}{L_m} - 1, 0\right). \tag{16}$$

To adapt the strength of density regulation to different scene layouts, we further define a scene-adaptive density weight:

$$w_{den} = \text{clamp}\left(\frac{2Q_{0.5}\left(\{o_m\}_{m=1}^{M}\right)}{Q_{0.9}\left(\{o_m\}_{m=1}^{M}\right)}, 0.5, 1.5\right). \tag{17}$$

Scenes with more uneven occupancy distributions yield a smaller w_{den} and therefore weaker density regulation, whereas more uniformly occupied scenes yield a larger w_{den} and stronger suppression of unnecessary Gaussian accumulation. The resulting weight modulates the density anomaly score in the subsequent joint candidate acceptance criterion.

4.3.3 Joint Candidate Acceptance.

The structural and density anomaly scores are jointly evaluated as:

$$r(\hat{g}) = r_{\text{str}}(\hat{g}) + \gamma w_{den} r_{\text{den}}(\hat{g}), \tag{18}$$

where γ balances the structural and density terms. A candidate Gaussian is rejected when:

$$r(\hat{g}) > \tau_{\text{str}}\left(1 + 0.5a_m\right). \tag{19}$$

In practice, candidates located in voxels whose current occupancy does not exceed L_m receive no density penalty, whereas candidates in over-budget voxels are increasingly penalized as local congestion grows. Voxels satisfying $n_m > L_m$ are further subjected to stronger opacity-based pruning, which preferentially removes low-contribution Gaussians while retaining informative ones.

5 Experiments

5.1 Experiment Settings

5.1.1 Baselines.

We compare CAGE-GS with standard 3DGS Kerbl et al. (2023), Gaussian Blur, DefenseSplat Qiao et al. (2026), and RemedyGS Li et al. (2026). Gaussian Blur applies isotropic smoothing before reconstruction; DefenseSplat removes designated high-frequency wavelet components; and RemedyGS uses a detector–purifier pipeline for suspicious views. Detailed baseline configurations are provided in Appendix E.3.

5.1.2 DATASETS AND METRICS.

We evaluate on Mip-NeRF 360 Barron et al. (2022), Tanks & Temples Knapitsch et al. (2017), and NeRF-Synthetic Mildenhall et al. (2021), using all nine, eight, and eight scenes, respectively. For poisoned inputs, we report maximum Gaussian count, peak GPU memory, and training time; for clean inputs, we report PSNR, SSIM Wang et al. (2004), and LPIPS Zhang et al. (2018).

5.1.3 IMPLEMENTATION DETAILS.

All poisoned-input experiments use $\epsilon = 16/255$. CAGE-GS is optimized for 30,000 iterations, using half-resolution views for the first 7,000. Wavelet preprocessing uses a two-level Daubechies-4 transform with $\rho = 0.15$, and congestion control uses $b = 60$, $\kappa = 2.0$, and $\gamma = 0.5$. Each run uses a single NVIDIA V100 GPU with 32 GB memory. Further implementation details are provided in Appendix D.

5.2 EXPERIMENT RESULTS

5.2.1 POISONED INPUTS.

Table 1 reports the computational overhead under poisoned inputs. Across all three datasets, poisoned training consistently induces excessive Gaussian densification and substantially increases training costs. Compared with clean training, poisoning increases the average Gaussian count by $1.69\times$, $2.47\times$, and $2.11\times$, and peak GPU memory by $1.51\times$, $2.45\times$, and $1.82\times$ on Tanks and Temples, NeRF-Synthetic, and Mip-NeRF 360, respectively.

CAGE-GS effectively suppresses this attack-induced expansion. Relative to poisoned training, it reduces the average Gaussian count by $2.45\times$, $2.48\times$, and $2.69\times$, peak GPU memory by $1.81\times$, $2.62\times$, and $1.96\times$, and training time by $1.55\times$, $1.40\times$, and $1.76\times$, respectively. Across all three benchmarks, the defended models recover computational costs close to those of clean training, indicating that CAGE-GS effectively prevents runaway Gaussian growth caused by poisoned inputs.

We further evaluate CAGE-GS on the Mip-Splatting Yu et al. (2024) framework. As shown in Table 2, poisoned inputs increase the average Gaussian count, peak GPU memory, and training time by $2.40\times$, $5.42\times$, and $1.44\times$, respectively. Notably, the GPU memory overhead is substantially larger than that observed on the original 3DGS pipeline, suggesting that computation cost attacks can become even more severe on related Gaussian-based rendering frameworks. Nevertheless, CAGE-GS reduces these three metrics by $2.93\times$, $4.51\times$, and $1.50\times$, respectively, relative to poisoned training. In several scenes, such as NS-ficus and NS-drums, the defended models even require computational resources comparable to or lower than those of clean training, indicating that CAGE-GS generalizes effectively to related Gaussian-based reconstruction frameworks.

5.2.2 CLEAN INPUTS.

Table 3 reports reconstruction quality under clean inputs. An effective defense should preserve reconstruction fidelity while mitigating computation cost attacks. Across all three benchmarks, CAGE-GS consistently achieves the best performance among all defense methods and remains closest to the original 3DGS baseline.

The advantage is particularly evident on NeRF-Synthetic, where CAGE-GS achieves 36.80 dB PSNR, 0.981 SSIM, and 0.024 LPIPS, nearly matching the original 3DGS (36.98/0.982/0.024). Similar trends are observed on Mip-NeRF 360 and Tanks & Temples. Compared with the competing defenses, the improvements are especially pronounced in SSIM and LPIPS, suggesting that CAGE-GS more effectively preserves structural consistency and fine perceptual details rather than merely improving pixel-wise fidelity.

Figure 3 provides qualitative comparisons. Existing defenses noticeably over-smooth textures and blur fine structures, whereas CAGE-GS preserves sharp boundaries and high-frequency details that remain visually close to the original 3DGS and the ground truth. These results are consistent with our design: by combining mild wavelet denoising with prior-guided 3D Gaussian expansion control, CAGE-GS preserves clean scene details while maintaining strong robustness against computation cost attacks. Although existing defenses may achieve stronger resource suppression in some cases,

Table 4: Resource ablation on poisoned inputs. Results are reported on Drums and Ficus (NS), and Ballroom and Truck (TT). Bold denotes the lowest resource use among defense variants; poisoned and clean 3DGS are included as references.

Method	Final Gaussians (M) ↓				Peak GPU Memory (GB) ↓				Training Time (min) ↓			
	Drums	Ficus	Ballroom	Truck	Drums	Ficus	Ballroom	Truck	Drums	Ficus	Ballroom	Truck
3DGS (Poisoned)	0.690	0.273	3.839	3.614	5.72	2.54	12.08	9.55	12.45	7.65	30.60	27.78
Resolution only	0.594	0.210	3.566	3.212	6.65	3.37	9.94	8.59	11.00	8.08	26.48	22.93
+ Mild Wavelet	0.478	0.185	2.455	1.812	4.22	2.96	7.80	6.06	9.98	7.92	22.73	16.18
+ AE Gate	**0.331**	**0.149**	**1.928**	**1.487**	**3.46**	**2.90**	**6.50**	**5.22**	**8.82**	**7.62**	**20.00**	**15.13**
+ Congestion Control (Full)	0.343	0.160	2.027	1.549	3.52	2.91	7.20	5.42	9.97	8.15	21.33	15.85
3DGS (Clean)	0.385	0.266	3.180	2.539	4.16	3.84	9.98	7.36	8.05	6.68	26.92	19.50

CAGE-GS maintains near-clean computational costs while preserving substantially better benign reconstruction quality. Additional poisoned-input comparisons are provided in Appendix F.2.

5.3 Ablations

Table 4 evaluates the contribution of each component under poisoned inputs. Resolution reduction alone provides limited protection, whereas our mild wavelet denoising substantially suppresses attack-induced Gaussian growth. Unlike DefenseSplat Qiao et al. (2026), which removes designated high-frequency wavelet components, our preprocessing retains the detail subbands and applies coefficient-wise soft thresholding. Adding the AE gate further reduces the Gaussian count, peak memory, and training time by 21.0%, 14.1%, and 9.2%, respectively, showing that the low-resolution structural prior effectively rejects implausible densification candidates.

Table 5: Effect of congestion control on clean-input quality, averaged over Drums/Ficus (NS) and Ballroom/Truck (TT).

Dataset	Method	PSNR ↑	SSIM ↑	LPIPS ↓
NeRF-Synth.	w/o Cong.	33.10	0.983	0.018
	Full	**33.54**	**0.994**	**0.012**
Tanks & Temples	w/o Cong.	24.81	0.843	0.185
	Full	**25.03**	**0.858**	**0.165**

Compared with the AE-gated variant, the full model increases Gaussian count, peak memory, and training time by only 4.7%, 5.4%, and 7.2%, respectively. This modest increase is expected because congestion control preserves structurally supported refinement within coarse-derived occupancy budgets, while increasingly penalizing expansion in over-congested regions. Thus, the full model avoids excessive suppression of legitimate Gaussians.

As shown in Table 5, this retained capacity consistently improves clean-input reconstruction quality. Meanwhile, relative to poisoned 3DGS, the full model still reduces the average Gaussian count, peak memory, and training time by 51.5%, 36.3%, and 29.5%, respectively, while keeping Gaussian count and peak memory below clean 3DGS in all evaluated scenes. These results demonstrate a favorable fidelity–efficiency trade-off. Additional parameter-sensitivity and attack-strength analyses are provided in Appendix F.

6 Conclusion

We presented CAGE-GS, a coarse-to-fine defense that uses low-resolution structural and density priors to regulate full-resolution Gaussian densification. Across three benchmarks and a related Gaussian-based pipeline, CAGE-GS restores poisoned-training costs toward clean levels while preserving reconstruction fidelity, demonstrating the effectiveness and transferability of low-resolution prior-guided expansion control.

7 AI USE STATEMENT

In this work, generative AI tools were used to assist with manuscript preparation, primarily for language editing, rephrasing, and improving the clarity and readability of the writing. Generative AI was not used to develop the research methodology, design the experiments, formulate mathematical claims, or interpret the experimental results. All AI-assisted text was reviewed and revised by the authors for accuracy and consistency with the underlying research. The authors take full responsibility for the final content of this work, including any text produced with the assistance of generative AI.

REFERENCES

Jonathan T Barron, Ben Mildenhall, Dor Verbin, Pratul P Srinivasan, and Peter Hedman. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 5470–5479, 2022.

Ingrid Daubechies. *Ten lectures on wavelets*. SIAM, 1992.

Jiaxin Hong, Sixu Chen, Shuoyang Sun, Hongyao Yu, Hao Fang, Yuqi Tan, Bin Chen, Shuhan Qi, and Jiawei Li. Gausstrap: Stealthy poisoning attacks on 3d gaussian splatting for targeted scene confusion. *arXiv preprint arXiv:2504.20829*, 2025.

Sanghyun Hong, Yigitcan Kaya, Ionut-Vlad Modoranu, and Tudor Dumitras. A panda? no, it's a sloth: Slowdown attacks on adaptive multi-exit neural network inference. In *International Conference on Learning Representations*, 2021.

Shoukang Hu, Tao Hu, and Ziwei Liu. Gauhuman: Articulated gaussian splatting from monocular human videos. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 20418–20431. IEEE, 2024.

Yi-Hua Huang, Yang-Tian Sun, Ziyi Yang, Xiaoyang Lyu, Yan-Pei Cao, and Xiaojuan Qi. Sc-gs: Sparse-controlled gaussian splatting for editable dynamic scenes. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4220–4230. IEEE, 2024.

Matthew Hull, Haoyang Yang, Pratham Mehta, Mansi Phute, Aeree Cho, Haoran Wang, Matthew Lau, Wenke Lee, Willian T Lunardi, Martin Andreoni, et al. 3d gaussian splat vulnerabilities. *arXiv preprint arXiv:2506.00280*, 2025.

Bo-Hsu Ke, You-Zhe Xie, Yu-Lun Liu, and Wei-Chen Chiu. Stealthattack: Robust 3d gaussian splatting poisoning via density-guided illusions. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 27400–27411. IEEE, 2025.

Nikhil Keetha, Jay Karhade, Krishna Murthy Jatavallabhula, Gengshan Yang, Sebastian Scherer, Deva Ramanan, and Jonathon Luiten. Splatam: Splat, track & map 3d gaussians for dense rgb-d slam. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 21357–21366. IEEE, 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.

Arno Knapitsch, Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. Tanks and temples: Benchmarking large-scale scene reconstruction. *ACM Transactions on Graphics (ToG)*, 36(4):1–13, 2017.

Yanping Li, Zhening Liu, Zijian Li, Zehong Lin, and Jun Zhang. Remedygs: Defend 3d gaussian splatting against computation cost attacks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 33227–33236, 2026.

Jiaqi Lin, Zhihao Li, Xiao Tang, Jianzhuang Liu, Shiyong Liu, Jiayue Liu, Yangdi Lu, Xiaofei Wu, Songcen Xu, Youliang Yan, et al. Vastgaussian: Vast 3d gaussians for large scene reconstruction. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5166–5175. IEEE, 2024.

Shuai Liu, Quanmin Liang, Zefeng Li, Boyang Li, and Kai Huang. Gaussianfusion: Gaussian-based multi-sensor fusion for end-to-end autonomous driving. *Advances in Neural Information Processing Systems*, 38:48062–48084, 2026.

Jiahao Lu, Yifan Zhang, Qiuhong Shen, Xinchao Wang, and Shuicheng Yan. Poison-splat: Computation cost attack on 3d gaussian splatting. In *International Conference on Learning Representations*, volume 2025, pp. 98599–98630, 2025.

Hidenobu Matsuki, Riku Murai, Paul HJ Kelly, and Andrew J Davison. Gaussian splatting slam. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 18039–18048. IEEE, 2024.

Lingzhuang Meng, Mingwen Shao, Yuanjian Qiao, Wenjie Liu, and Xiang Lv. Gaussian splitting attack: Gaussian splatting-based multi-view 3d adversarial attack. *Pattern Recognition*, pp. 112466, 2025.

Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106, 2021.

Yiran Qiao, Yiren Lu, Yunlai Zhou, Rui Yang, Linlin Hou, Yu Yin, and Jing Ma. Defensesplat: Enhancing the robustness of 3d gaussian splatting via frequency-aware filtering. *arXiv preprint arXiv:2602.19323*, 2026.

Minghan Qin, Wanhua Li, Jiawei Zhou, Haoqian Wang, and Hanspeter Pfister. Langsplat: 3d language gaussian splatting. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 20051–20060. IEEE, 2024.

Johannes L Schonberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4104–4113, 2016.

Avishag Shapira, Alon Zolfi, Luca Demetrio, Battista Biggio, and Asaf Shabtai. Phantom sponges: Exploiting non-maximum suppression to attack deep object detectors. In *2023 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pp. 4560–4569. IEEE, 2023.

Ilia Shumailov, Yiren Zhao, Daniel Bates, Nicolas Papernot, Robert Mullins, and Ross Anderson. Sponge examples: Energy-latency attacks on neural networks. In *2021 IEEE European symposium on security and privacy (EuroS&P)*, pp. 212–231. IEEE, 2021.

Yongxin Su, Chen Lin, Kaiting Zhang, Ziping Yu, Chenfeng Hou, and Zhongliang Zhao. Gaus-slam: Dense rgb-d slam with gaussian surfels. *IEEE Robotics and Automation Letters*, 2026.

Jiaxiang Tang, Jiawei Ren, Hang Zhou, Ziwei Liu, and Gang Zeng. Dreamgaussian: Generative gaussian splatting for efficient 3d content creation. In *International Conference on Learning Representations*, volume 2024, pp. 33879–33896, 2024.

Prajas Wadekar, Venkata Sai Pranav Bachina, Kunal Bhosikar, Ankit Gangwal, and Charu Sharma. Patchpoison: Poisoning multi-view datasets to degrade 3d reconstruction. *arXiv preprint arXiv:2604.13153*, 2026.

Weijie Wang, Songlong Xing, Zhengyu Zhao, Nicu Sebe, and Bruno Lepri. Poinit-of-view: Poisoning initialization of views transfers across multiple 3d reconstruction systems. *arXiv preprint arXiv:2604.16540*, 2026.

Yifan Wang, Peishan Yang, Zhen Xu, Jiaming Sun, Zhanhua Zhang, Yong Chen, Hujun Bao, Sida Peng, and Xiaowei Zhou. Freetimegs: Free gaussian primitives at anytime anywhere for dynamic scene reconstruction. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 21750–21760. IEEE, 2025.

Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.

Minghao Wen, Shengjie Wu, Kangkan Wang, and Dong Liang. Intergsedit: Interactive 3d gaussian splatting editing with 3d geometry-consistent attention prior. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 26136–26145. IEEE, 2025.

Guanjun Wu, Taoran Yi, Jiemin Fang, Lingxi Xie, Xiaopeng Zhang, Wei Wei, Wenyu Liu, Qi Tian, and Xinggang Wang. 4d gaussian splatting for real-time dynamic scene rendering. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 20310–20320. IEEE, 2024a.

Jiahao Wu, Rui Peng, Zhiyan Wang, Lu Xiao, Luyang Tang, Jinbo Yan, Kaiqiang Xiong, and Ronggang Wang. Swift4d: Adaptive divide-and-conquer gaussian splatting for compact and efficient reconstruction of dynamic scene. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=c1RhJVTPwT.

Jing Wu, Jia-Wang Bian, Xinghui Li, Guangrun Wang, Ian Reid, Philip Torr, and Victor Adrian Prisacariu. Gaussctrl: Multi-view consistent text-driven 3d gaussian splatting editing. In *European conference on computer vision*, pp. 55–71. Springer, 2024b.

Cihang Xie, Jianyu Wang, Zhishuai Zhang, Zhou Ren, and Alan L. Yuille. Mitigating adversarial effects through randomization. In *International Conference on Learning Representations*, 2018.

Ziyi Yang, Xinyu Gao, Wen Zhou, Shaohui Jiao, Yuqing Zhang, and Xiaogang Jin. Deformable 3d gaussians for high-fidelity monocular dynamic scene reconstruction. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 20331–20341. IEEE, 2024.

Yao Yao, Zixin Luo, Shiwei Li, Tian Fang, and Long Quan. Mvsnet: Depth inference for unstructured multi-view stereo. In *European conference on computer vision*, pp. 785–801. Springer, 2018.

Zehao Yu, Anpei Chen, Binbin Huang, Torsten Sattler, and Andreas Geiger. Mip-splatting: Alias-free 3d gaussian splatting. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 19447–19456, 2024.

Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *2018 IEEE/CVF conference on computer vision and pattern recognition*, pp. 586–595. IEEE, 2018.

Xiaoyu Zhou, Zhiwei Lin, Xiaojun Shan, Yongtao Wang, Deqing Sun, and Ming-Hsuan Yang. Drivinggaussian: Composite gaussian splatting for surrounding dynamic autonomous driving scenes. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 21634–21643. IEEE, 2024.

A Appendix

This appendix provides additional discussion, implementation details, and experimental results that complement the main paper. We first discuss the limitations of CAGE-GS and provide further scene-level evidence for the resolution-dependent robustness phenomenon that motivates our coarse-to-fine design. We then describe the complete training procedure, wavelet preprocessing, structural autoencoder, and voxel-based congestion control in greater detail. Finally, we report the experimental environment and reproducibility settings, followed by extended evaluations on hyperparameter sensitivity, robustness to different perturbation strengths, and additional quantitative and qualitative comparisons.

B Discussion and Limitations

CAGE-GS approaches computation cost attacks from the perspective of Gaussian representation rather than relying solely on aggressive input purification. Its key observation is that low-resolution optimization retains more stable Gaussian structure and spatial density than full-resolution poisoned reconstruction, allowing the coarse model to provide scene-specific references for distinguishing legitimate refinement from attack-induced expansion. Mild wavelet denoising attenuates perturbations while preserving useful high-frequency content, the structural prior filters geometrically implausible candidates, and congestion control suppresses excessive local accumulation without completely disabling refinement. This design therefore balances computational robustness and reconstruction fidelity instead of minimizing Gaussian count unconditionally. Its consistent performance across multiple datasets and the Mip-Splatting Yu et al. (2024) pipeline also suggests that low-resolution prior-guided expansion control may generalize to related Gaussian reconstruction frameworks.

CAGE-GS still has several limitations. Its effectiveness depends on the low-resolution reconstruction remaining sufficiently robust; an adaptive attack that survives downsampling or directly corrupts the structural and density priors may weaken the defense. The current evaluation also focuses mainly on Poison-Splat-style Lu et al. (2025) computation cost attacks in static scenes and does not cover geometry corruption, appearance manipulation, or attacks jointly targeting preprocessing and expansion control. In addition, coarse reconstruction, scene-specific autoencoder training, and voxel-prior construction introduce extra implementation complexity. Extending the method to dynamic scenes, online reconstruction, or changing Gaussian distributions may require temporal or incremental prior updates. Future work will investigate stronger adaptive attacks, broader Gaussian-based systems, and dynamically updated priors.

C Resolution-Dependent Robustness on Mip-NeRF 360

This section provides scene-level evidence for the resolution-dependent robustness phenomenon underlying CAGE-GS. For each scene, we train standard 3DGS Kerbl et al. (2023) under four settings: clean and poisoned inputs at full and half resolution. We evaluate six representative scenes from Mip-NeRF 360 Barron et al. (2022), including three unbounded outdoor scenes (*Bicycle*, *Garden*, and *Stump*) and three bounded indoor scenes (*Bonsai*, *Counter*, and *Room*). Together, these scenes cover diverse scene structures and a broad range of attack-induced Gaussian inflation.

C.1 Gaussian-Field Similarity.

We measure the similarity between each clean–poisoned pair using eight Gaussian-field statistics. Count is defined as $\min(N_c, N_p)/\max(N_c, N_p)$. Position, Size, and Orientation are computed as the mean histogram overlap of the per-axis Gaussian centers, logarithmic scales, and canonicalized quaternion components, respectively. Shape measures the overlap of logarithmic scale anisotropy; Opacity and Appearance measure the overlap of opacity and DC luminance; and Local Density measures the overlap of normalized eight-nearest-neighbor distances. All scores lie in $[0, 1]$, with larger values indicating greater similarity to the corresponding clean reconstruction.

As shown in Figure 4, half-resolution training improves all eight similarity measures for all six scenes, resulting in 48 consistent improvements out of 48 comparisons. On average, Count increases from 42.8% to 91.2%, Shape from 48.4% to 83.3%, Size from 66.5% to 89.5%, and Local Density

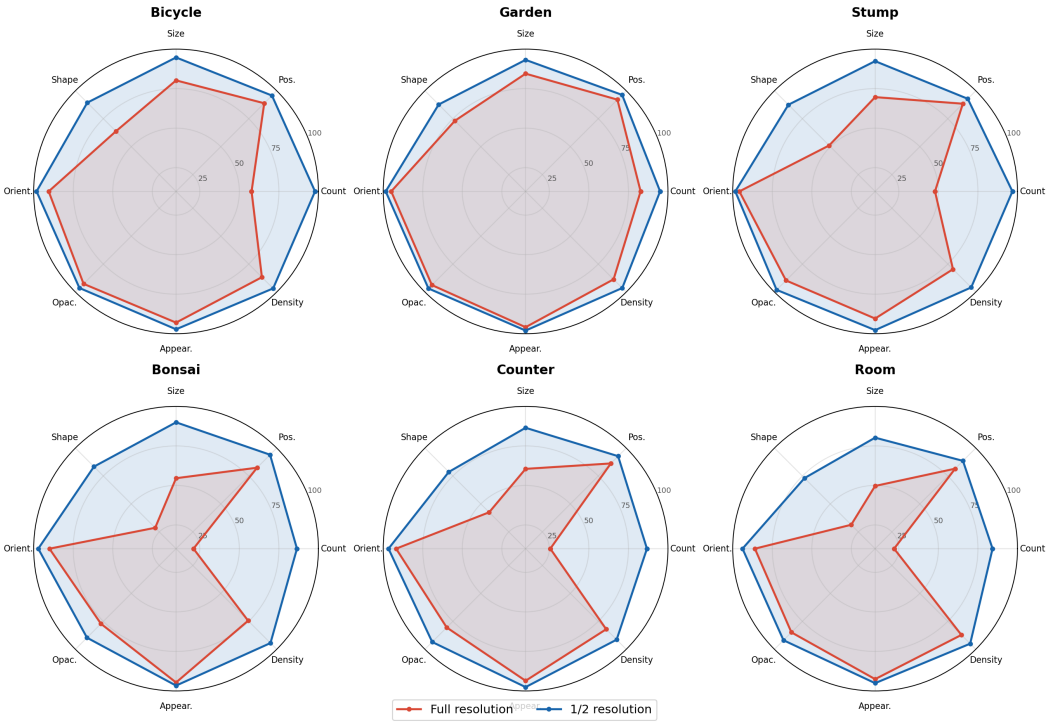

Figure 4: Poisoned-to-clean Gaussian-field similarity on six Mip-NeRF 360 scenes. Each axis reports histogram-overlap similarity, except Count, which is the ratio of the smaller to the larger Gaussian count. Red and blue polygons denote full- and half-resolution training, respectively. Half-resolution training consistently yields higher similarity across all scenes and statistics.

from 83.1% to 95.0%. Position, Orientation, Opacity, and Appearance increase from 86.6%, 91.5%, 86.4%, and 93.3% to 93.4%, 97.1%, 94.4%, and 97.0%, respectively. The largest full-resolution distortions occur in Count, Shape, and Size, which is consistent with an attack that repeatedly triggers unnecessary Gaussian cloning and splitting. These geometric attributes also benefit most from resolution reduction, improving by 48.4, 34.9, and 23.0 percentage points, respectively. By contrast, Appearance improves by only 3.7 points. This result supports using geometric attributes, rather than view-dependent color coefficients, to construct the structural prior.

C.2 Spatial Density.

Figure 5 further shows where the additional Gaussians accumulate. At full resolution, poisoned reconstruction produces visibly brighter and more concentrated density regions. Relative to clean training, the number of Gaussians projected into the shared camera view increases by $1.24\times$–$4.70\times$. At half resolution, this range decreases to $0.99\times$–$1.31\times$, and the clean and poisoned density maps become visually similar.

These results indicate that the low-resolution reconstruction preserves not only plausible attributes of individual Gaussians but also the normal spatial occupancy pattern of the scene. It therefore provides a meaningful reference for constructing both the structural prior and the voxel-wise density budget used by CAGE-GS.

C.3 Computational Overhead.

The improved Gaussian-field similarity is also reflected in computational cost. Averaged over the six scenes, full-resolution poisoning increases the maximum Gaussian count from 3.57M to 7.73M, peak GPU memory from $13{,}131$MB to $25{,}594$MB, and training time from 34.39 to 55.85 minutes. These changes correspond to $2.16\times$, $1.95\times$, and $1.62\times$ inflation, respectively.

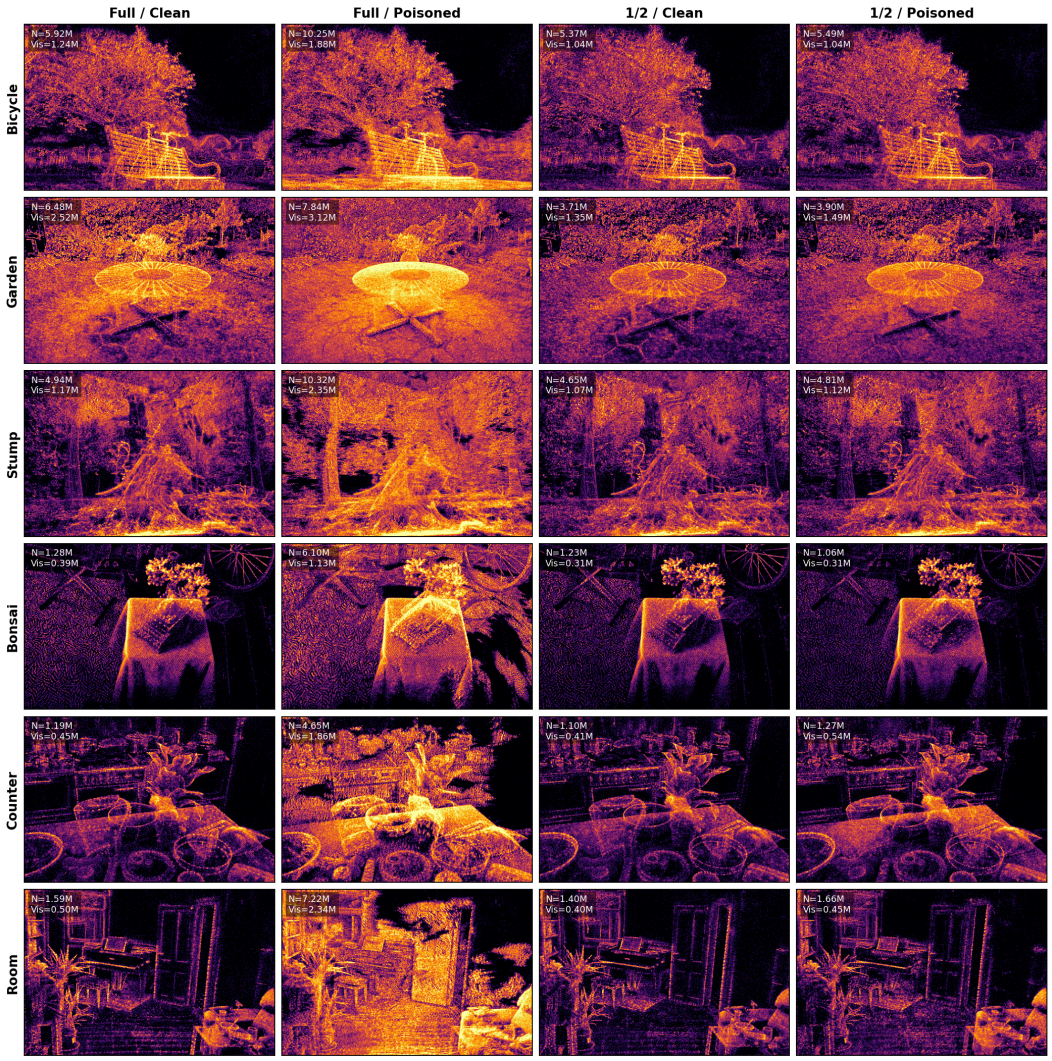

Figure 5: Projected Gaussian density on the same six scenes. Columns show full-resolution clean, full-resolution poisoned, half-resolution clean, and half-resolution poisoned reconstructions. Panels within each row share the same color scale. N and Vis denote the total and visible Gaussian counts, respectively.

At half resolution, the corresponding poisoned-to-clean ratios decrease to $1.04\times$, $1.03\times$, and $1.04\times$. At the scene level, Gaussian inflation ranges from $1.21\times$ to $4.71\times$ under full-resolution training, but only from $0.87\times$ to $1.19\times$ at half resolution. Thus, even scenes that are severely affected at full resolution exhibit near-clean computational behavior after resolution reduction.

C.4 Implications for CAGE-GS.

Taken together, these observations justify using half-resolution reconstruction as a scene-specific reference for benign Gaussian structure. The coarse stage is substantially less affected by poisoning and requires approximately 19 minutes on average, compared with 34 minutes for clean full-resolution training. It provides a structural prior over plausible Gaussian geometry and a density prior over normal spatial occupancy, which are subsequently used to regulate full-resolution densification.

The coarse reconstruction is not used as the final output, because half-resolution optimization alone cannot recover all fine-scale scene details. Instead, it serves only as a robust intermediate repre-

Algorithm 1 Overall Training Procedure of CAGE-GS

Input: Training views $\mathcal{V}$; total iterations T; transition iteration T_c; wavelet strength ρ; structural quantile q; voxel size v; density parameters b, κ, and γ
Output: Defended Gaussian representation $\mathcal{G}$

Apply mild wavelet denoising: $\widetilde{\mathcal{V}} \leftarrow \mathrm{WaveletDenoise}(\mathcal{V}, \rho)$
Construct half-resolution views $\widetilde{\mathcal{V}}_{\mathrm{low}}$ and retain full-resolution views $\widetilde{\mathcal{V}}_{\mathrm{full}}$
Initialize $\mathcal{G}$ from SfM points
for $t = 1$ to T_c **do**
  Sample a view from $\widetilde{\mathcal{V}}_{\mathrm{low}}$
  Render $\mathcal{G}$ and minimize the standard 3DGS loss
  if t is a densification iteration **then**
    Apply standard 3DGS cloning, splitting, and pruning
  end if
end for
Extract and standardize structural features from the coarse Gaussian field
Train autoencoder $(\mathcal{E}, \mathcal{D})$ and set τ_{str} to the q-quantile of the coarse reconstruction errors
Construct coarse voxel occupancies $\{o_m\}$ and budgets $\{L_m\}$
Compute the scene-adaptive density weight w_{den} and freeze the structural and density priors
for $t = T_c + 1$ to T **do**
  Sample a view from $\widetilde{\mathcal{V}}_{\mathrm{full}}$
  Render $\mathcal{G}$ and compute the standard 3DGS loss
  Update $\mathcal{G}$ by backpropagation
  if t is a densification iteration **then**
    Generate clone and split candidates $\widehat{\mathcal{G}}$ using the standard 3DGS gradient criterion
    for each $\hat{g} \in \widehat{\mathcal{G}}$ **do**
      Compute the structural anomaly score $r_{\mathrm{str}}(\hat{g})$
      Locate voxel m and query o_m, n_m, L_m, and a_m
      Compute the density anomaly score $r_{\mathrm{den}}(\hat{g}) \leftarrow \max(n_m/L_m - 1, 0)$
      Compute the joint anomaly score $r(\hat{g}) \leftarrow r_{\mathrm{str}}(\hat{g}) + \gamma w_{\mathrm{den}} r_{\mathrm{den}}(\hat{g})$
      if $r(\hat{g}) \le \tau_{\mathrm{str}}(1 + 0.5a_m)$ **then**
        Accept and insert $\hat{g}$ into $\mathcal{G}$
        Update the current voxel occupancy: $n_m \leftarrow n_m + 1$
      else
        Reject $\hat{g}$
      end if
    end for
    Update voxel occupancies and mark voxels satisfying $n_m > L_m$ as congested
    Apply standard global pruning and stronger opacity-based pruning in congested voxels
  end if
end for
return $\mathcal{G}$

sentation from which the two priors are extracted. Full-resolution optimization is then continued under prior-guided expansion control, preserving reconstruction capacity while suppressing attack-induced Gaussian growth.

D Additional Implementation Details

D.1 Overall Training Algorithm

Algorithm 1 summarizes the complete training procedure of CAGE-GS. The method follows a coarse-to-fine optimization schedule. We first apply mild wavelet denoising and optimize 3DGS on half-resolution views. The resulting coarse Gaussian field is then used to construct a structural normality model and a voxel-wise density prior. During full-resolution optimization, these fixed priors jointly regulate clone and split candidates generated by standard 3DGS densification.

CAGE-GS preserves the original gradient-based candidate generation mechanism. The structural prior evaluates whether a candidate follows the geometric attribute distribution of the coarse Gaussian field, while the density prior measures whether its local region has expanded beyond the coarse-derived occupancy budget. Importantly, this budget is not a hard capacity constraint. Regions with strong coarse occupancy support are allocated greater refinement capacity, and densification remains enabled even after a voxel becomes congested. In that case, CAGE-GS progressively strengthens candidate filtering and opacity-based pruning, allowing informative Gaussians to be retained while preferentially suppressing low-contribution primitives.

D.2 Wavelet Preprocessing

The wavelet denoising procedure is applied independently to each color channel using a two-level Daubechies-4 DWT Daubechies (1992). The threshold is estimated independently for each detail subband; different subbands generally use different thresholds. After the inverse DWT, the reconstructed output is cropped to the original image dimensions, clipped to the valid intensity range, and converted back to an 8-bit image.

D.3 Training Schedule and Resolution Transition

CAGE-GS follows the standard 30,000-iteration 3DGS optimization schedule. The first $T_c = 7{,}000$ iterations use wavelet-denoised half-resolution views. At the end of this coarse stage, we extract the Gaussian structural features, train the autoencoder, and construct the voxel occupancy prior. These operations are performed once, and all derived priors remain fixed thereafter.

The full-resolution stage continues directly from the coarse Gaussian representation without reinitializing its parameters or optimizer state. We replace the half-resolution camera views with their full-resolution counterparts and continue optimization until iteration 30,000. Standard 3DGS Kerbl et al. (2023) densification is used during the coarse stage. After the transition, gradient-based clone and split candidates are generated in the same manner as standard 3DGS, but are filtered by the fixed structural and density priors before insertion. Densification starts at iteration 500, is performed every 100 iterations, and ends at iteration 15,000; opacity is reset every 3,000 iterations, following the standard 3DGS schedule.

D.4 Structural Autoencoder

The structural prior models the distribution of plausible Gaussian geometry in the coarse reconstruction. For every coarse Gaussian, we construct the 12-dimensional feature vector defined in the main paper, consisting of the normalized 3D position, logarithmic scale, canonicalized rotation quaternion, logit opacity, and logarithmic scale anisotropy. Appearance coefficients are excluded because computation cost attacks primarily manifest through abnormal geometric expansion rather than view-dependent color.

Before training the autoencoder, each feature dimension is standardized using the mean and standard deviation computed over all coarse Gaussians in the current scene. The structural autoencoder is a symmetric multilayer perceptron. Its encoder has dimensions

$$12 \to 64 \to 64 \to 4,$$

and its decoder mirrors this architecture:

$$4 \to 64 \to 64 \to 12.$$

ReLU activations are used after each hidden layer, while the latent and output layers are linear. The compact four-dimensional latent representation forces the network to capture the dominant structural patterns of the coarse Gaussian field rather than simply copying individual features.

We train one autoencoder independently for each scene using Adam with learning rate 10^{-3} for 1,000 epochs. At each epoch, at most 65,536 coarse Gaussians are randomly sampled to form the training batch. The optimization objective is the mean squared reconstruction error over the standardized features:

$$\mathcal{L}_{\mathrm{AE}} = \frac{1}{|\mathcal{B}|} \sum_{\mathbf{x} \in \mathcal{B}} \|\mathbf{x} - \mathcal{D}(\mathcal{E}(\mathbf{x}))\|_2^2, \tag{20}$$

where $\mathcal{B}$ denotes the sampled batch. After training, the reconstruction error of each coarse Gaussian is

$$r_z = \|\mathbf{x}_z - \mathcal{D}(\mathcal{E}(\mathbf{x}_z))\|_2^2 . \tag{21}$$

The structural threshold is set to the 0.85 quantile:

$$\tau_{\text{str}} = Q_{0.85}\left(\{r_z\}_{z=1}^{N}\right). \tag{22}$$

The autoencoder, feature normalization statistics, and τ_{str} are then frozen. During full-resolution densification, clone and split candidates are transformed using the same normalization statistics and evaluated by the fixed autoencoder. This scene-specific normality model accommodates legitimate variations in Gaussian geometry while assigning high reconstruction errors to candidates that deviate from the stable coarse representation.

D.5 Voxel Prior and Congestion Control

D.5.1 Coarse-Guided Refinement Capacity.

The voxel prior distinguishes legitimate refinement in geometrically complex regions from attack-induced local accumulation. A voxel with high occupancy in the coarse reconstruction is likely to contain genuine geometric or textural complexity. Instead of imposing a uniform global limit, CAGE-GS allocates greater refinement capacity to such coarse-supported regions.

We partition the scene into uniform voxels. Let $\{o_j\}_{j=1}^{M}$ denote the occupancies of the M non-empty voxels in the coarse reconstruction. For any queried voxel m, we use $o_m = 0$ when it is absent from the coarse field. The local refinement factor is defined as

$$a_m = \min\left(\frac{o_m}{Q_{0.5}(\{o_j\}_{j=1}^{M})}, 1\right), \tag{23}$$

where the quantile is computed only over non-empty coarse voxels. The voxel-wise occupancy budget is then given by

$$L_m = \begin{cases} b, & o_m = 0, \\ o_m + \kappa\sqrt{o_m + 1} + a_m o_m + b, & o_m > 0. \end{cases} \tag{24}$$

We use $b = 60$ and $\kappa = 2.0$ in all experiments.

For a coarse-occupied voxel, the budget preserves its existing occupancy while allocating additional capacity according to its local structural support. The term $a_m o_m$ preferentially permits further refinement in regions that are already well represented at low resolution. For a voxel absent from the coarse representation, the base budget b still allows fine structures that become visible only at full resolution to emerge. Thus, the density prior does not confine the full-resolution reconstruction to the spatial support observed at the coarse stage.

D.5.2 Progressive Congestion-Aware Filtering.

Let n_m denote the current number of full-resolution Gaussians in voxel m. For a candidate $\hat{g}$ located in this voxel, we compute

$$r_{\text{den}}(\hat{g}) = \max\left(\frac{n_m}{L_m} - 1, 0\right). \tag{25}$$

When $n_m \le L_m$, the density score is zero, and candidate acceptance is determined primarily by structural plausibility. Gaussian growth therefore proceeds normally within the allocated refinement capacity. When $n_m > L_m$, the density score increases continuously with the degree of over-occupancy rather than immediately blocking subsequent growth.

To adapt density regulation to different scene layouts, we compute

$$w_{\text{den}} = \text{clamp}\left(\frac{2Q_{0.5}(\{o_j\}_{j=1}^{M})}{Q_{0.9}(\{o_j\}_{j=1}^{M})}, 0.5, 1.5\right). \tag{26}$$

We then combine the density score with the autoencoder reconstruction error:

$$r(\hat{g}) = r_{\text{str}}(\hat{g}) + \gamma w_{\text{den}} r_{\text{den}}(\hat{g}), \tag{27}$$

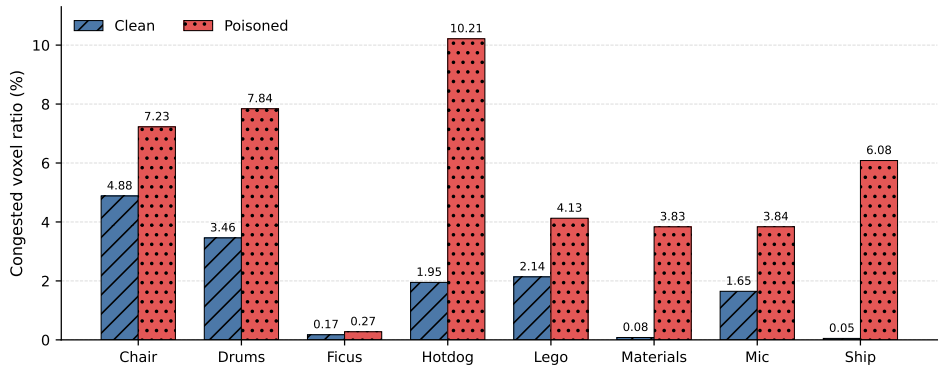

Figure 6: Congested voxel ratios under clean and poisoned inputs on all eight NeRF-Synthetic Mildenhall et al. (2021) scenes. We report the fraction of active voxels whose occupancy exceeds the coarse-derived budget after the final densification step. Poisoning increases the macro-average ratio from 1.80% to 5.43%.

where $\gamma = 0.5$ controls the density contribution. A candidate is accepted when

$$r(\hat{g}) \leq \tau_{\text{str}}(1 + 0.5a_m). \tag{28}$$

The factor $1 + 0.5a_m$ provides a more permissive structural threshold in regions supported by high coarse occupancy. Consequently, such regions can retain more structurally plausible candidates, while unsupported or over-expanded regions are regulated more strictly.

Figure 6 reports the fraction of active voxels whose current occupancy exceeds the coarse-derived budget:

$$R_{\text{cong}} = \frac{N_{\text{congested voxels}}}{N_{\text{active voxels}}} \times 100\%. \tag{29}$$

Because the voxel census is updated at each densification event, we measure this ratio immediately after the final densification step. Poisoned inputs yield a higher congestion ratio than clean inputs in every NeRF-Synthetic scene. The differences are particularly pronounced on *Hotdog*, *Materials*, and *Ship*, where the ratios increase from 1.95% to 10.21%, from 0.08% to 3.83%, and from 0.05% to 6.08%, respectively. Across all eight scenes, poisoning increases the macro-average congestion ratio from 1.80% to 5.43%. These results show that computation cost attacks more frequently induce local Gaussian accumulation beyond the scene-specific capacity estimated from the coarse reconstruction.

D.5.3 Continued Refinement in Congested Voxels.

Although Figure 6 shows that poisoned inputs activate congestion control more frequently, CAGE-GS does not freeze a voxel or disable its densification once $n_m > L_m$. A hard stopping rule could prevent the recovery of legitimate full-resolution details and cause irreversible quality degradation. Instead, congestion activates a stricter but adaptive update policy.

New candidates in congested voxels continue to be evaluated by the joint structural-density criterion. As local occupancy increases, r_{den} grows, progressively favoring candidates that are strongly supported by the structural prior. A candidate may therefore still be accepted when its structural plausibility is sufficient to compensate for the density penalty.

During pruning, congested voxels use a stricter opacity criterion, preferentially removing low-contribution Gaussians and releasing capacity for more informative candidates. Congestion control therefore forms a dynamic replacement process: local refinement remains active, while redundant Gaussians are preferentially removed.

E Additional Experimental Settings

E.1 Hardware and Software Environment

All experiments are conducted on a Linux server equipped with two Intel Xeon Gold 5218 CPUs at 2.30 GHz, providing 32 physical cores and 64 logical CPU threads in total. The server has 128 GB of system memory and two NVIDIA Tesla V100S GPUs, each with 32 GB of memory. Each individual training run uses only one GPU; the two GPUs are used solely to execute independent scene-level experiments in parallel. Thus, the resource measurements reported for each scene correspond to a single-GPU run and do not aggregate resources across devices.

Our implementation uses Python 3.11.15 and PyTorch 2.5.1 with CUDA 12.1 and cuDNN 9.1. The NVIDIA driver version is 550.120. We use the differentiable Gaussian rasterizer and CUDA-accelerated nearest-neighbor implementation distributed with the official 3DGS codebase Kerbl et al. (2023). Image loading and preprocessing use Pillow 12.2.0 and OpenCV 5.0.0.

E.2 Reproducibility and Random Seeds

Unless stated otherwise, all methods use the same initialization and optimization settings as the official 3DGS implementation. At the beginning of each run, we set the Python, NumPy, and PyTorch random seeds to 0. This seed controls camera sampling, Gaussian densification randomness, and the random subsampling used to train the structural autoencoder. Each scene and input condition is trained independently from its corresponding SfM initialization.

The same seed is used for clean and poisoned inputs and for all parameter settings. We do not select the best result from multiple random restarts. Consequently, all reported scene-level values correspond to one run under the fixed seed, following the standard evaluation protocol used by 3DGS. Although the fixed seed removes the principal sources of stochastic variation, minor numerical differences may remain because of parallel GPU operations in the differentiable rasterizer.

E.3 Baseline Configurations

For a controlled comparison, all image-preprocessing baselines use the same camera parameters, SfM initialization, 30,000-iteration optimization schedule, densification interval, opacity-reset interval, and rendering and evaluation code as standard 3DGS. A preprocessing method changes only the input images unless its original defense explicitly requires additional components.

For the **Gaussian Blur** baseline, every RGB training image is filtered independently using Pillow's isotropic Gaussian blur operator with radius 1 before 3DGS training. We apply the same filter to clean and poisoned inputs and keep all other 3DGS settings unchanged. The reference images used for evaluation are not blurred, so the reported reconstruction metrics remain measured against the original unprocessed views.

For **DefenseSplat Qiao et al. (2026)**, we follow its frequency-domain preprocessing protocol and remove the designated high-frequency wavelet detail components before reconstruction. For **RemedyGS Li et al. (2026)**, we use its released detector–purifier pipeline and apply purification only to views classified as suspicious by its detector. The resulting views are then passed to the standard 3DGS training pipeline. For both methods, we use their released models and recommended inference settings without scene-specific parameter tuning.

F Extended Experiments

F.1 Ablation Studies

F.1.1 Parameter Sensitivity

Tables 6 and 7 evaluate the sensitivity of CAGE-GS to its three principal hyperparameters. Overall, the method remains effective across all tested configurations. Under poisoned inputs, every setting reduces the maximum Gaussian count by at least $2.29\times$, peak GPU memory by at least $1.71\times$, and

Table 6: Parameter sensitivity of CAGE-GS under poisoned inputs on Tanks & Temples Knapitsch et al. (2017). We vary the structural threshold quantile q, base voxel budget b, and density weight γ one at a time while fixing the remaining parameters to the default values $q = 0.85$, $b = 60$, and $\gamma = 0.5$. Results are averaged over all eight scenes. Blue subscripts denote reductions relative to undefended 3DGS trained on poisoned inputs.

Setting	**Gaussians (M)** ↓	**Memory (MB)** ↓	**Time (min)** ↓
3DGS (Undefended)	3.115	10536	24.53
$q = 0.80$	$1.259_{2.47\times\downarrow}$	$5854_{1.80\times\downarrow}$	$15.85_{1.55\times\downarrow}$
$q = 0.85$ **(Default)**	$1.305_{2.39\times\downarrow}$	$5902_{1.78\times\downarrow}$	$16.04_{1.53\times\downarrow}$
$q = 0.90$	$1.362_{2.29\times\downarrow}$	$6178_{1.71\times\downarrow}$	$16.34_{1.50\times\downarrow}$
$b = 40$	$1.293_{2.41\times\downarrow}$	$6047_{1.74\times\downarrow}$	$15.99_{1.53\times\downarrow}$
$b = 60$ **(Default)**	$1.305_{2.39\times\downarrow}$	$5902_{1.78\times\downarrow}$	$16.04_{1.53\times\downarrow}$
$b = 80$	$1.312_{2.37\times\downarrow}$	$5937_{1.77\times\downarrow}$	$16.03_{1.53\times\downarrow}$
$\gamma = 0.25$	$1.298_{2.40\times\downarrow}$	$5976_{1.76\times\downarrow}$	$16.00_{1.53\times\downarrow}$
$\gamma = 0.50$ **(Default)**	$1.305_{2.39\times\downarrow}$	$5902_{1.78\times\downarrow}$	$16.04_{1.53\times\downarrow}$
$\gamma = 0.75$	$1.310_{2.38\times\downarrow}$	$6044_{1.74\times\downarrow}$	$16.01_{1.53\times\downarrow}$

Table 7: Parameter sensitivity of CAGE-GS under clean inputs on Tanks & Temples Knapitsch et al. (2017). We vary q, b, and γ one at a time while fixing the remaining parameters to the default values $q = 0.85$, $b = 60$, and $\gamma = 0.5$. Results are averaged over all eight scenes.

Setting	**Gaussians (M)** ↓	**Memory (MB)** ↓	**Time (min)** ↓	**PSNR** ↑	**SSIM** ↑	**LPIPS** ↓
3DGS (Undefended)	1.842	6976	17.67	24.77	0.857	0.176
$q = 0.80$	1.289	5826	15.83	24.28	0.841	0.208
$q = 0.85$ **(Default)**	1.346	5862	16.21	24.35	0.843	0.205
$q = 0.90$	1.408	6001	16.36	24.40	0.845	0.203
$b = 40$	1.335	5748	16.07	24.29	0.840	0.207
$b = 60$ **(Default)**	1.346	5862	16.21	24.35	0.843	0.205
$b = 80$	1.356	5971	16.11	24.37	0.844	0.205
$\gamma = 0.25$	1.349	5826	16.10	24.41	0.844	0.205
$\gamma = 0.50$ **(Default)**	1.346	5862	16.21	24.35	0.843	0.205
$\gamma = 0.75$	1.359	5886	16.11	24.32	0.844	0.203

training time by at least $1.50\times$ relative to undefended 3DGS. This consistent reduction indicates that the effectiveness of CAGE-GS does not depend on a narrowly tuned parameter configuration.

Among the three parameters, the structural quantile q produces the clearest efficiency–fidelity trade-off. Increasing q from 0.80 to 0.90 makes the structural acceptance threshold more permissive, allowing more candidate Gaussians to pass the structural gate. Consequently, under poisoned inputs, the maximum Gaussian count increases from 1.259M to 1.362M, peak GPU memory increases from 5854MB to 6178MB, and training time increases from 15.85 to 16.34 minutes. On clean inputs, however, the same change improves PSNR from 24.28 to 24.40 dB, SSIM from 0.841 to 0.845, and LPIPS from 0.208 to 0.203. A smaller q therefore provides stricter suppression, whereas a larger q preserves slightly more legitimate scene detail. The default value $q = 0.85$ provides an intermediate operating point between these two objectives.

CAGE-GS is substantially less sensitive to the base voxel budget b. Varying b from 40 to 80 changes the poisoned-input Gaussian count only from 1.293M to 1.312M, while training time remains approximately 16 minutes. On clean inputs, increasing b slightly improves reconstruction quality, because a larger base budget permits additional refinement in voxels with limited or no coarse-stage occupancy. Nevertheless, the differences remain small: PSNR varies from 24.29 to 24.37 dB and

Table 8: Robustness of CAGE-GS under different perturbation strengths of constrained Poison-Splat Lu et al. (2025) on NeRF-Synthetic. Red and blue subscripts indicate multiplicative increases and decreases, respectively. Ratios in the poisoned columns are measured relative to clean 3DGS, while those in the Ours columns are measured relative to poisoned 3DGS. Results for $\epsilon = 8/255$ and $16/255$ are averaged over all eight scenes; for $\epsilon = 32/255$, we report Ficus and Materials, for which complete poisoned datasets are available.

	Max. Gaussians (M) ↓			Peak GPU Memory (MB) ↓			Training Time (min) ↓		
Scene	**Clean**	**Poisoned**	**Ours**	**Clean**	**Poisoned**	**Ours**	**Clean**	**Poisoned**	**Ours**
					$\epsilon = 8/255$				
NS-chair	0.495	$0.672_{1.36\times\uparrow}$	$0.302_{2.23\times\downarrow}$	4541	$3231_{1.41\times\downarrow}$	$2333_{1.38\times\downarrow}$	9.17	$11.32_{1.23\times\uparrow}$	$9.22_{1.23\times\downarrow}$
NS-drums	0.385	$0.471_{1.22\times\uparrow}$	$0.274_{1.72\times\downarrow}$	4259	$2901_{1.47\times\downarrow}$	$2271_{1.28\times\downarrow}$	8.05	$10.35_{1.29\times\uparrow}$	$9.27_{1.12\times\downarrow}$
NS-ficus	0.266	$0.248_{1.07\times\downarrow}$	$0.153_{1.62\times\downarrow}$	3935	$2051_{1.92\times\downarrow}$	$1915_{1.07\times\downarrow}$	6.68	$8.08_{1.21\times\uparrow}$	$7.63_{1.06\times\downarrow}$
NS-hotdog	0.184	$0.602_{3.27\times\uparrow}$	$0.175_{3.44\times\downarrow}$	4171	$8603_{2.06\times\uparrow}$	$2485_{3.46\times\downarrow}$	7.20	$13.77_{1.91\times\uparrow}$	$9.12_{1.51\times\downarrow}$
NS-lego	0.347	$0.475_{1.37\times\uparrow}$	$0.300_{1.58\times\downarrow}$	4505	$3485_{1.29\times\downarrow}$	$2751_{1.27\times\downarrow}$	8.17	$10.42_{1.28\times\uparrow}$	$9.38_{1.11\times\downarrow}$
NS-materials	0.166	$0.227_{1.36\times\uparrow}$	$0.154_{1.47\times\downarrow}$	3969	$2317_{1.71\times\downarrow}$	$2111_{1.10\times\downarrow}$	5.97	$7.37_{1.23\times\uparrow}$	$7.70_{1.05\times\uparrow}$
NS-mic	0.203	$0.252_{1.24\times\uparrow}$	$0.150_{1.69\times\downarrow}$	4117	$2551_{1.61\times\downarrow}$	$2407_{1.06\times\downarrow}$	6.60	$8.23_{1.25\times\uparrow}$	$8.42_{1.02\times\uparrow}$
NS-ship	0.273	$0.513_{1.88\times\uparrow}$	$0.226_{2.27\times\downarrow}$	4297	$4099_{1.05\times\downarrow}$	$2763_{1.48\times\downarrow}$	8.18	$11.80_{1.44\times\uparrow}$	$9.70_{1.22\times\downarrow}$
Average	**0.290**	$\mathbf{0.433}_{1.49\times\uparrow}$	$\mathbf{0.217}_{2.00\times\downarrow}$	**4224**	$\mathbf{3655}_{1.16\times\downarrow}$	$\mathbf{2380}_{1.54\times\downarrow}$	**7.50**	$\mathbf{10.17}_{1.36\times\uparrow}$	$\mathbf{8.80}_{1.15\times\downarrow}$
					$\epsilon = 16/255$				
NS-chair	0.495	$0.960_{1.94\times\uparrow}$	$0.388_{2.47\times\downarrow}$	4541	$7503_{1.65\times\uparrow}$	$4151_{1.81\times\downarrow}$	9.17	$14.45_{1.58\times\uparrow}$	$10.47_{1.38\times\downarrow}$
NS-drums	0.385	$0.692_{1.80\times\uparrow}$	$0.328_{2.11\times\downarrow}$	4259	$7037_{1.65\times\uparrow}$	$3503_{2.01\times\downarrow}$	8.05	$12.27_{1.52\times\uparrow}$	$9.70_{1.26\times\downarrow}$
NS-ficus	0.266	$0.274_{1.03\times\uparrow}$	$0.168_{1.63\times\downarrow}$	3935	$3445_{1.14\times\downarrow}$	$3007_{1.15\times\downarrow}$	6.68	$7.65_{1.14\times\uparrow}$	$8.28_{1.08\times\uparrow}$
NS-hotdog	0.184	$1.143_{6.21\times\uparrow}$	$0.274_{4.18\times\downarrow}$	4171	$29181_{7.00\times\uparrow}$	$4157_{7.02\times\downarrow}$	7.20	$19.90_{2.76\times\uparrow}$	$9.43_{2.11\times\downarrow}$
NS-lego	0.347	$0.829_{2.39\times\uparrow}$	$0.391_{2.12\times\downarrow}$	4505	$9303_{2.07\times\uparrow}$	$4775_{1.95\times\downarrow}$	8.17	$13.97_{1.71\times\uparrow}$	$10.43_{1.34\times\downarrow}$
NS-materials	0.166	$0.411_{2.47\times\uparrow}$	$0.207_{1.99\times\downarrow}$	3969	$4519_{1.14\times\uparrow}$	$3421_{1.32\times\downarrow}$	5.97	$9.07_{1.52\times\uparrow}$	$7.93_{1.14\times\downarrow}$
NS-mic	0.203	$0.360_{1.77\times\uparrow}$	$0.186_{1.93\times\downarrow}$	4117	$5895_{1.43\times\uparrow}$	$3663_{1.61\times\downarrow}$	6.60	$10.33_{1.57\times\uparrow}$	$8.62_{1.20\times\downarrow}$
NS-ship	0.273	$1.066_{3.91\times\uparrow}$	$0.367_{2.90\times\downarrow}$	4297	$15953_{3.71\times\uparrow}$	$4925_{3.24\times\downarrow}$	8.18	$18.77_{2.29\times\uparrow}$	$11.17_{1.68\times\downarrow}$
Average	**0.290**	$\mathbf{0.717}_{2.47\times\uparrow}$	$\mathbf{0.289}_{2.48\times\downarrow}$	**4224**	$\mathbf{10354}_{2.45\times\uparrow}$	$\mathbf{3950}_{2.62\times\downarrow}$	**7.50**	$\mathbf{13.30}_{1.77\times\uparrow}$	$\mathbf{9.50}_{1.40\times\downarrow}$
					$\epsilon = 32/255$				
NS-ficus	0.266	$0.854_{3.21\times\uparrow}$	$0.193_{4.43\times\downarrow}$	3935	$23669_{6.01\times\uparrow}$	$2245_{10.54\times\downarrow}$	6.68	$15.32_{2.29\times\uparrow}$	$7.93_{1.93\times\downarrow}$
NS-materials	0.166	$0.821_{4.93\times\uparrow}$	$0.260_{3.15\times\downarrow}$	3969	$11639_{2.93\times\uparrow}$	$2833_{4.11\times\downarrow}$	5.97	$13.22_{2.22\times\uparrow}$	$8.95_{1.48\times\downarrow}$
Average	**0.216**	$\mathbf{0.837}_{3.87\times\uparrow}$	$\mathbf{0.227}_{3.70\times\downarrow}$	**3952**	$\mathbf{17654}_{4.47\times\uparrow}$	$\mathbf{2539}_{6.95\times\downarrow}$	**6.33**	$\mathbf{14.27}_{2.26\times\uparrow}$	$\mathbf{8.44}_{1.69\times\downarrow}$

SSIM from 0.840 to 0.844. These results show that the coarse occupancy prior, rather than the absolute base capacity alone, dominates the allocation of local refinement capacity.

The density weight γ also has only a limited influence within the evaluated range. Under poisoned inputs, changing γ from 0.25 to 0.75 results in maximum Gaussian counts between 1.298M and 1.310M, with nearly unchanged training times. Clean-input reconstruction metrics are similarly stable, with PSNR between 24.32 and 24.41 dB, SSIM between 0.843 and 0.844, and LPIPS between 0.203 and 0.205. This stability suggests that the structural prior and the adaptive voxel budget already provide the primary constraints, while the density term acts as a gradual penalty for locally over-occupied regions rather than a dominant hard constraint.

Across all tested parameter configurations, clean-input PSNR varies by only 0.13 dB, while SSIM and LPIPS each vary by no more than 0.005. Therefore, the remaining reconstruction-quality difference from undefended 3DGS is not primarily caused by the selection of q, b, or γ. Based on the combined poisoned- and clean-input results, the default configuration $q = 0.85$, $b = 60$, and $\gamma = 0.5$ is adopted as a balanced setting and is used consistently in the remaining experiments.

F.1.2 Robustness to Perturbation Strength

We further investigate whether CAGE-GS generalizes across poisoning attacks with different perturbation budgets. As shown in Table 8, CAGE-GS consistently suppresses poisoning-induced resource

Table 9: Quantitative comparison under poisoned inputs. Standard 3DGS trained on clean and poisoned inputs is included as reference, while all defenses are evaluated on the corresponding poisoned inputs. Blue subscripts denote reductions relative to poisoned 3DGS. Results are averaged over eight NeRF-Synthetic scenes, eight Tanks & Temples scenes, and nine Mip-NeRF 360 scenes.

Dataset	Method	Max. Gaussians (M) ↓	Peak Memory (MB) ↓	Time (min) ↓
NeRF-Synthetic	3DGS (Clean)	0.290	4224	7.50
	3DGS (Poisoned)	0.717	10354	13.30
	Gaussian Blur	$0.148_{4.84\times\downarrow}$	$2003_{5.17\times\downarrow}$	$7.06_{1.88\times\downarrow}$
	DefenseSplat Qiao et al. (2026)	$0.135_{5.31\times\downarrow}$	$1971_{5.25\times\downarrow}$	$6.68_{1.99\times\downarrow}$
	RemedyGS Li et al. (2026)	$0.348_{2.06\times\downarrow}$	$2662_{3.89\times\downarrow}$	$8.95_{1.49\times\downarrow}$
	CAGE-GS (Ours)	$0.289_{2.48\times\downarrow}$	$3950_{2.62\times\downarrow}$	$9.50_{1.40\times\downarrow}$
Tanks & Temples	3DGS (Clean)	1.842	6976	17.67
	3DGS (Poisoned)	3.115	10536	24.53
	Gaussian Blur	$0.924_{3.37\times\downarrow}$	$4114_{2.56\times\downarrow}$	$13.07_{1.88\times\downarrow}$
	DefenseSplat Qiao et al. (2026)	$0.825_{3.78\times\downarrow}$	$3873_{2.72\times\downarrow}$	$12.47_{1.97\times\downarrow}$
	RemedyGS Li et al. (2026)	$1.136_{2.74\times\downarrow}$	$4440_{2.37\times\downarrow}$	$14.12_{1.74\times\downarrow}$
	CAGE-GS (Ours)	$1.270_{2.45\times\downarrow}$	$5827_{1.81\times\downarrow}$	$15.87_{1.55\times\downarrow}$
Mip-NeRF 360	3DGS (Clean)	3.356	12553	33.62
	3DGS (Poisoned)	7.073	22798	53.06
	Gaussian Blur	$2.007_{3.52\times\downarrow}$	$9715_{2.35\times\downarrow}$	$26.71_{1.99\times\downarrow}$
	DefenseSplat Qiao et al. (2026)	$1.979_{3.57\times\downarrow}$	$9689_{2.35\times\downarrow}$	$26.15_{2.03\times\downarrow}$
	RemedyGS Li et al. (2026)	$2.548_{2.78\times\downarrow}$	$11243_{2.03\times\downarrow}$	$29.11_{1.82\times\downarrow}$
	CAGE-GS (Ours)	$2.631_{2.69\times\downarrow}$	$11633_{1.96\times\downarrow}$	$30.24_{1.75\times\downarrow}$

Table 10: Clean-input reconstruction quality. Bold and underlined entries denote the best and second-best results among defense methods, respectively. Original 3DGS is shown as an undefended reference and excluded from the ranking.

		Mip-NeRF 360			NeRF-Synthetic			Tanks & Temples		
Method	**Source**	**PSNR ↑**	**SSIM ↑**	**LPIPS ↓**	**PSNR ↑**	**SSIM ↑**	**LPIPS ↓**	**PSNR ↑**	**SSIM ↑**	**LPIPS ↓**
3DGS Kerbl et al. (2023)	Original	29.51	0.863	0.196	36.98	0.982	0.024	24.77	0.857	0.176
Gaussian Blur	Naive	<u>28.30</u>	<u>0.798</u>	<u>0.260</u>	<u>32.21</u>	<u>0.960</u>	0.063	24.05	<u>0.806</u>	<u>0.259</u>
DefenseSplat Qiao et al. (2026)	ECCV 2026	27.11	0.733	0.327	27.64	0.912	0.098	23.09	0.745	0.335
RemedyGS Li et al. (2026)	CVPR 2026	27.21	0.767	0.274	28.12	0.928	<u>0.051</u>	<u>24.10</u>	0.803	0.262
CAGE-GS	**Ours**	**28.83**	**0.839**	**0.229**	**36.80**	**0.981**	**0.024**	**24.35**	**0.842**	**0.206**

inflation for $\epsilon \in \{8/255, 16/255, 32/255\}$. Under the weak $\epsilon = 8/255$ attack, it reduces the average maximum Gaussian count, peak GPU memory, and training time by $2.00\times$, $1.54\times$, and $1.15\times$ relative to poisoned standard 3DGS. At the default $\epsilon = 16/255$ budget, the corresponding reductions increase to $2.48\times$, $2.62\times$, and $1.40\times$.

The benefit is most pronounced under the strongest evaluated attack. For $\epsilon = 32/255$, poisoned standard 3DGS reaches 0.837M Gaussians and 17654MB peak memory on average, whereas CAGE-GS requires only 0.227M Gaussians and 2539MB, producing $3.70\times$ and $6.95\times$ reductions, respectively. In addition, CAGE-GS reduces the training time from 14.27 to 8.44 minutes. While the resource consumption of standard 3DGS increases rapidly as the perturbation budget grows, CAGE-GS maintains comparatively stable resource usage. These results demonstrate that CAGE-GS does not rely on a specific attack strength and provides robust defense against weak, moderate, and strong poisoning attacks.

F.2 Additional Experimental Results

Tables 9, 10, and 11 jointly evaluate the effectiveness and fidelity of the considered defenses. The poisoned-input resource results show that all evaluated defenses can mitigate the abnormal Gaussian

Table 11: Clean-input resource consumption averaged over nine Mip-NeRF 360 scenes, eight NeRF-Synthetic scenes, and eight Tanks & Temples scenes. Lower resource consumption does not necessarily indicate better benign behavior, as aggressive preprocessing may suppress legitimate scene content.

Method	Source	Max. Gaussians (M) ↓	Peak Memory (MB) ↓	Time (min) ↓
Mip-NeRF 360				
3DGS	Original	3.356	12553	33.62
Gaussian Blur	Naive	2.136	10065	27.19
DefenseSplat Qiao et al. (2026)	ECCV 2026	2.026	9735	26.32
RemedyGS Li et al. (2026)	CVPR 2026	2.666	11006	29.92
CAGE-GS	**Ours**	2.524	11246	28.35
NeRF-Synthetic				
3DGS	Original	0.290	4224	7.50
Gaussian Blur	Naive	0.159	2000	6.29
DefenseSplat Qiao et al. (2026)	ECCV 2026	0.140	1976	6.12
RemedyGS Li et al. (2026)	CVPR 2026	0.229	2182	7.50
CAGE-GS	**Ours**	0.204	4149	7.58
Tanks & Temples				
3DGS	Original	1.842	6977	17.67
Gaussian Blur	Naive	0.982	4173	13.44
DefenseSplat Qiao et al. (2026)	ECCV 2026	0.858	3964	12.60
RemedyGS Li et al. (2026)	CVPR 2026	1.215	4570	14.58
CAGE-GS	**Ours**	1.312	5778	15.99

proliferation induced by the computation cost attack. Therefore, the main distinction among these methods is not simply whether they reduce the resource overhead, but whether they do so without unnecessarily suppressing legitimate scene content.

As shown in Table 9, standard 3DGS suffers substantial resource inflation under poisoned inputs on all three datasets. Its average maximum Gaussian count increases from 0.290M to 0.717M on NeRF-Synthetic, from 1.842M to 3.115M on Tanks & Temples, and from 3.356M to 7.073M on Mip-NeRF 360. Peak GPU memory and training time exhibit similar increases. All defense methods successfully suppress this attack-induced expansion. In particular, CAGE-GS reduces the poisoned Gaussian count by $2.48\times$, $2.45\times$, and $2.69\times$ on the three datasets, respectively. It also reduces peak memory by $2.62\times$, $1.81\times$, and $1.96\times$, demonstrating consistent computational robustness across synthetic, bounded real-world, and unbounded real-world scenes.

Gaussian Blur and DefenseSplat sometimes produce smaller Gaussian models and lower resource consumption than CAGE-GS. However, the clean-input results indicate that these smaller resource footprints are partly associated with aggressive suppression of legitimate image information. As shown in Table 11, DefenseSplat reduces the clean-input Gaussian count from 3.356M to 2.026M on Mip-NeRF 360, from 0.290M to 0.140M on NeRF-Synthetic, and from 1.842M to 0.858M on Tanks & Temples. Gaussian Blur exhibits a similar reduction. Such pronounced under-expansion can lower computational cost, but it also reduces the representation capacity available for reconstructing valid scene details.

This trade-off is reflected directly in the reconstruction-quality results of Table 10. Although Gaussian Blur and DefenseSplat achieve low clean-input resource consumption, they incur clear quality degradation. Relative to original 3DGS, DefenseSplat decreases PSNR from 29.51 to 27.11 on Mip-NeRF 360, from 36.98 to 27.64 on NeRF-Synthetic, and from 24.77 to 23.09 on Tanks & Temples. Gaussian Blur also reduces PSNR and SSIM and increases LPIPS across all three datasets. These results show that minimizing the number of Gaussians unconditionally is not equivalent to faithfully restoring the clean reconstruction process.

In contrast, CAGE-GS retains substantially higher clean-input reconstruction quality while still controlling resource consumption. On NeRF-Synthetic, CAGE-GS achieves 36.80 PSNR, 0.981 SSIM, and 0.024 LPIPS, nearly matching the original 3DGS results of 36.98, 0.982, and 0.024. It also achieves the best clean-input quality among the evaluated defenses on Mip-NeRF 360 and Tanks & Temples, reaching 28.83 and 24.35 PSNR, respectively. At the same time, its clean-input resource usage remains below or close to that of original 3DGS. For example, on NeRF-Synthetic, CAGE-GS uses 0.204M Gaussians and 4149MB peak memory, compared with 0.290M and 4224MB for original 3DGS. On Tanks & Temples, it reduces the clean-input Gaussian count from 1.842M to 1.312M while retaining substantially higher reconstruction quality than the alternative defenses.

These results clarify the objective of CAGE-GS. The method is not designed to produce the smallest possible Gaussian representation. Instead, it seeks to remove attack-induced redundant expansion while preserving the capacity required for legitimate scene reconstruction. Consequently, CAGE-GS may use more Gaussians than aggressive image-filtering defenses, but these additional Gaussians support valid scene structures rather than attack-induced proliferation. The combination of the three tables therefore demonstrates that CAGE-GS achieves a more favorable balance between computational robustness and reconstruction fidelity: it effectively defends against poisoned inputs while preserving clean-input behavior substantially better than methods that rely on direct and aggressive image suppression.

Figure 7 further visualizes the spatial effect of the computation cost attack and the corresponding recovery produced by CAGE-GS. For each scene, the three Gaussian fields are projected from the same camera pose using the same image resolution and a shared density color scale. Therefore, differences in brightness directly reflect differences in the number and spatial concentration of Gaussians projected into the same view. The poisoned 3DGS models consistently produce brighter and more congested density maps than their clean counterparts, confirming that the resource inflation reported in Table 9 corresponds to substantial redundant occupancy inside the effective reconstruction volume rather than merely to Gaussians outside the evaluated views.

The severity of this congestion varies across scenes. Relative to clean 3DGS, the number of visible Gaussians increases by $3.65\times$, $2.14\times$, and $1.75\times$ on *Auditorium*, *Meetingroom*, and *Courtroom*, respectively. The increase is even more pronounced on the selected NeRF-Synthetic scenes, reaching $6.20\times$ on *Hotdog*, $3.88\times$ on *Ship*, and $2.56\times$ on *Materials*. Averaged over the six scenes, poisoned training increases the visible Gaussian count to approximately $3.36\times$ that of clean training. The additional Gaussians are also spatially nonuniform: they accumulate in locally congested regions instead of uniformly improving the scene representation. This observation is consistent with the attack repeatedly triggering unnecessary Gaussian cloning and splitting.

CAGE-GS suppresses most of this attack-induced congestion. After defense, the visible Gaussian count relative to clean 3DGS becomes $0.88\times$, $0.83\times$, and $0.74\times$ on the three Tanks and Temples scenes, and $1.48\times$, $1.34\times$, and $1.28\times$ on the three NeRF-Synthetic scenes. Equivalently, CAGE-GS removes between 50.1% and 76.2% of the visible Gaussians present in the poisoned models. Across all six scenes, the average visible-Gaussian ratio decreases from $3.36\times$ before defense to $1.09\times$ after defense. The resulting density maps consequently recover both the overall intensity and the spatial organization of the corresponding clean Gaussian fields.

The defended Gaussian count does not exactly match the clean count in every scene. In the selected Tanks and Temples scenes, CAGE-GS produces slightly fewer visible Gaussians than clean 3DGS, whereas in the NeRF-Synthetic scenes it retains moderately more Gaussians. This scene-dependent behavior is expected because the defense does not impose a fixed global Gaussian budget. Instead, the coarse reconstruction estimates the structural complexity and normal spatial occupancy of each scene, allowing the subsequent full-resolution stage to retain the capacity required by legitimate geometry. For synthetic objects with sharp silhouettes and high-frequency structures, the defended model can therefore preserve more Gaussians when they are supported by the scene-specific prior.

The visualization also explains why global resource limiting alone is insufficient. The attack changes not only the total Gaussian number but also where the redundant Gaussians are created. A fixed Gaussian cap may reduce the overall model size while still permitting severe local congestion or, conversely, suppressing Gaussians required for valid scene details. By constraining voxel-wise occupancy with the coarse density reference, CAGE-GS targets abnormal local expansion while preserving scene-dependent representation capacity. This mechanism is consistent with the quanti-

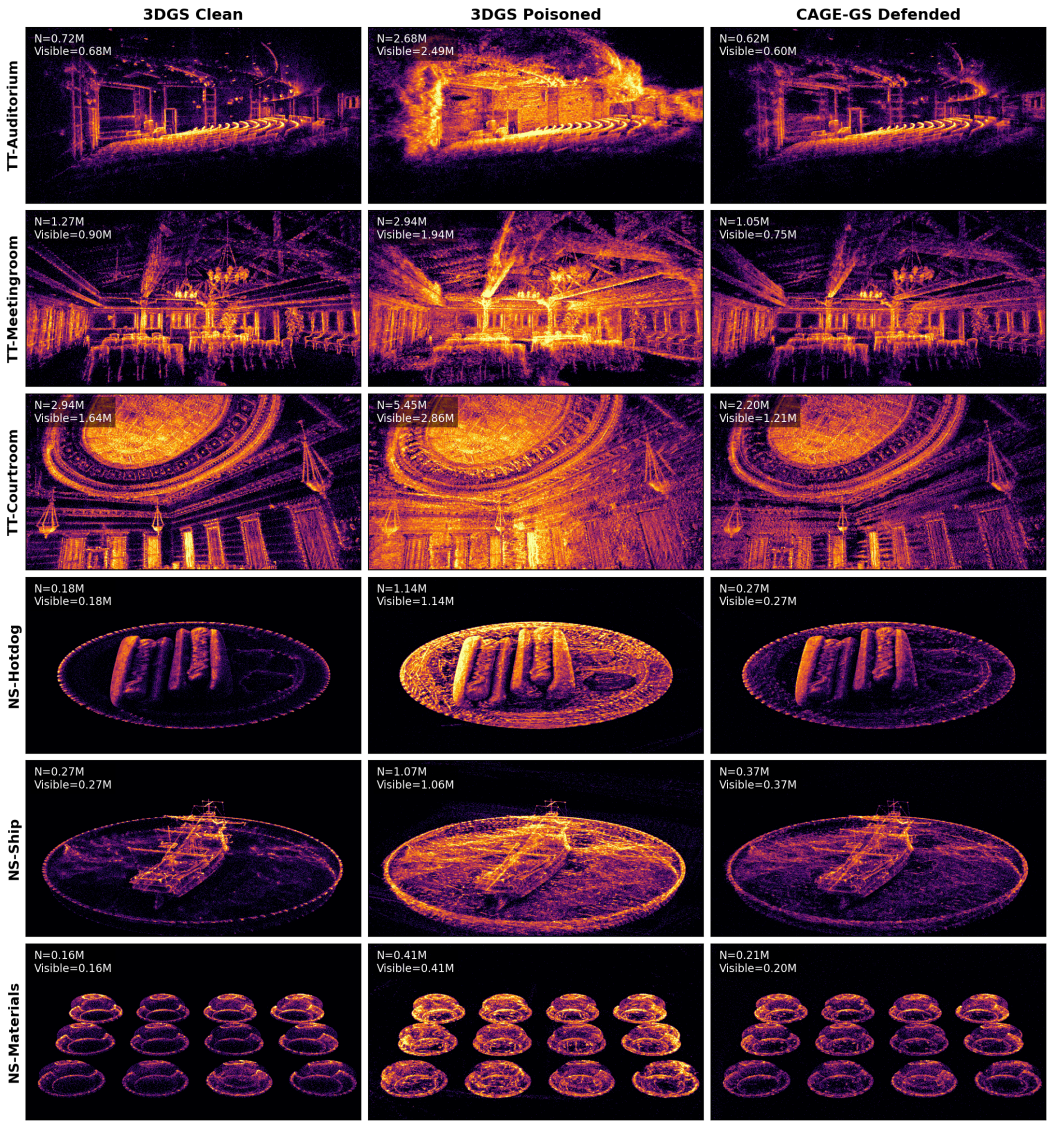

Figure 7: Projected Gaussian-density comparison on representative Tanks & Temples (TT) and NeRF-Synthetic (NS) scenes. Each row uses the same camera pose and color scale for clean, poisoned, and defended reconstructions. Poisoning induces strong local Gaussian congestion, whereas CAGE-GS largely restores the spatial-density pattern toward the clean reconstruction. N and Visible denote the total and visible Gaussian counts, respectively.

tative results above: CAGE-GS may retain more Gaussians than aggressive filtering methods, but it simultaneously achieves substantially better clean-input reconstruction quality.

Together with Tables 9, 10, and 11, Figure 7 demonstrates that the effectiveness of CAGE-GS does not arise from indiscriminately shrinking the Gaussian field. Instead, the method removes attack-induced redundant Gaussians and restores the field to a scene-dependent benign density range. This selective recovery accounts for its favorable balance between computational robustness and reconstruction fidelity.