HIDDENSPLAT: INJECTING VIEW-SPECIFIC CONTENT INTO 3D GAUSSIAN SPLATTING VIA "INVISIBLE" GAUSSIANS

Anonymous authors
Paper under double-blind review

ABSTRACT

3D Gaussian Splatting (3DGS) has rapidly emerged as a leading representation for photorealistic novel-view synthesis and is increasingly adopted in AR/VR, robotics, and digital content creation. As 3DGS scenes become widely distributed and reused as shareable assets, they introduce an underexplored security risk: an adversary can inject a set of structurally optimized Gaussians into a pretrained 3DGS scene such that a chosen target content (e.g., a sensitive image, logo, watermark, or hidden message) appears clearly when the scene is rendered from a specific target viewpoint while remaining visually imperceptible from other viewpoints. We term this attack HiddenSplat. Unlike prior approaches, HiddenSplat efficiently operates in both white-box and gray-box settings, enables the attacker to arbitrarily select the target viewpoint, and requires no auxiliary encoders or decoders. Extensive experiments across diverse scenes and target patterns demonstrate that HiddenSplat achieves high-fidelity target reconstruction at the attack viewpoint while keeping the injected content effectively "invisible" from non-target views. Our findings reveal a new, practical attack surface to 3DGS assets and highlight the need for efficient defenses against such vulnerabilities.

1 INTRODUCTION

3D Gaussian Splatting (3DGS) (Kerbl et al., 2023) has rapidly emerged as a leading representation for photorealistic 3D scene reconstruction and novel-view synthesis. By representing a scene as a collection of anisotropic Gaussian primitives with view-dependent appearance encoded using Spherical Harmonics (SH), 3DGS enables real-time rendering while achieving visual quality comparable to or surpassing that of Neural Radiance Fields (NeRFs) (Mildenhall et al., 2021). Its rendering efficiency and high visual fidelity have led to widespread adoption across AR/VR (Chen et al., 2025a; Zhai et al., 2025), robotics and embodied AI (Chen et al., 2025b; Wu et al., 2025), autonomous driving (Zhou et al., 2024; Hess et al., 2025), and digital content creation (Tang et al., 2024). Consequently, pretrained 3DGS scenes are increasingly distributed as standalone assets across platforms.

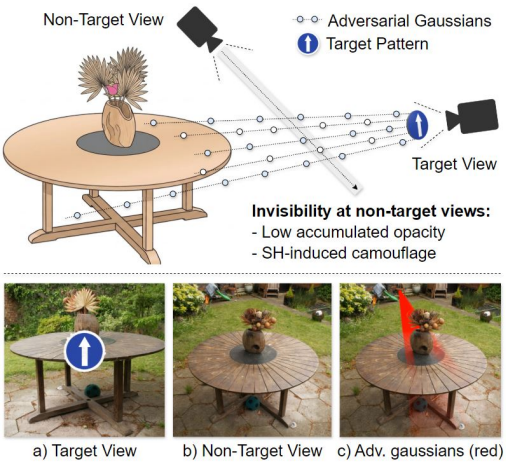


Figure 1: Illustration of HiddenSplat. Our method injects low-opacity Gaussians into 3DGS scenes such that the Gaussians accumulate into a chosen target content only at a specific view, while imperceptible at other views.

Despite these advantages, the widespread distribution of 3DGS assets introduces emerging security and integrity risks. Recent studies have revealed multiple attack surfaces, including rendering-cost attacks that significantly increase computational overhead (Li et al., 2026; Lu et al., 2025), camouflage attacks that conceal physical objects from neural perception systems (Lou et al.,

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.

2025), adversarial perturbations that degrade rendering quality or mislead downstream models (Jiang et al., 2025), and data poisoning attacks that compromise the training process of radiance-based representations (Ke et al., 2025). These findings demonstrate that 3DGS assets, like other learned representations, can be maliciously manipulated in ways that remain difficult for end users to detect.

In this work, we investigate a new post-training attack that targets the integrity of pretrained 3DGS assets. As illustrated in Figure 1, given a well-trained 3DGS scene, the attacker injects a set of low-opacity Gaussians arranged in a carefully designed spatial structure. The injected Gaussians are optimized such that they coherently accumulate along the rays of a designated target viewpoint, producing an attacker-chosen visual content with high fidelity. From all other viewpoints, the adversarial Gaussians become imperceptible owing to their low opacity and sparse spatial distribution. We refer to this attack as HiddenSplat.

Although viewpoint-dependent attacks have previously been explored for NeRFs (Jiang et al., 2024), those methods exploit properties of implicit MLP-based scene representations and cannot be directly transferred to 3DGS, whose primitive-based rasterization pipeline exhibits fundamentally different rendering behavior. The closest related work in the 3DGS literature is StealthAttack (Ke et al., 2025), which embeds a target content into viewpoints that are sparsely covered by the training cameras. Consequently, the target content is merely hidden in regions that observers are unlikely to visit, rather than being intrinsically invisible. Whenever a camera points in its direction, the target content remains directly observable. Moreover, the attacker cannot freely choose which viewpoint reveals the target content, limiting practicality in realistic deployment scenarios.

HiddenSplat adopts a fundamentally different perspective inspired by anamorphic art, in which a carefully designed 3D structure appears as a coherent content only when observed from a particular viewpoint. To achieve this behavior, we optimize the injected Gaussians using four complementary objectives. First, a target reconstruction loss, $\mathcal{L}_{\text{adv}}$, encourages faithful reconstruction of the attacker-specified content at the target viewpoint. Second, a non-target stealth loss, $\mathcal{L}_{\text{sth}}$, minimizes visual differences from the original scene across normal viewpoints. Third, a repulsion loss, $\mathcal{L}_{\text{rep}}$, prevents adversarial Gaussians from collapsing into dense, noticeable clusters by maintaining an appropriate spatial distribution. Finally, a color-spreading loss, $\mathcal{L}_{\text{col}}$, encourages neighboring adversarial Gaussians to adopt diverse colors, reducing the chance that they overlap into coherent colored structures when observed from non-target viewpoints. Together, these objectives enable HiddenSplat to achieve high-fidelity target reconstruction while remaining imperceptible from normal views.

2 Related Work

2.1 Safety and Security of 3DGS

As 3DGS has been increasingly adopted in real-world applications, its safety and security have emerged as important research topics. Poison-Splat (Lu et al., 2025) significantly increases the memory and computational cost of training 3DGS by poisoning the input images. To defend against such computational cost attacks, RemedyGS (Li et al., 2026) introduces a detector to identify poisoned images and a purifier to recover clean images from them. AdvSplat (Qiao et al., 2026) presents an adversarial attack against feed-forward 3DGS that degrades 3D reconstruction quality by injecting adversarial perturbations into the input images. GS-Hider (Zhang et al., 2024) enables steganography by embedding hidden messages into 3DGS through a scene encoder and a message decoder. Similarly, GuardSplat (Chen et al., 2025c) embeds watermarks into 3DGS using a message embedding module and retrieves them with a message decoder.

2.2 View-Specific Attacks

ComplicitSplat (Hull et al., 2025) exploits the view-dependent nature of spherical harmonics to perform camouflage attacks on 3DGS. Specifically, it replaces a subset of the training images with adversarial counterparts to conceal an object from specific viewpoints, causing object detection models to misclassify the target. Rather than hiding existing objects in a scene, IPA-NeRF (Jiang et al., 2024) injects a view-specific object that is visible only from a designated viewpoint while remaining invisible from all others. Although its objective is similar to ours, IPA-NeRF is designed specifically for NeRF and cannot be directly applied to 3DGS due to the fundamental differences between NeRF and 3DGS. For 3DGS, StealthAttack (Ke et al., 2025) injects target content into

sparsely observed regions of the scene. However, the injected object remains visible whenever the camera is directed toward it, and the attacker cannot arbitrarily choose the target viewpoint.

3 Problem Formulation

3.1 Preliminaries

Given a dataset $\mathcal{D} = \{(I_k, P_k)\}_{k=1}^{N}$ of N image-camera pose pairs, 3DGS reconstructs the scene as a set of n Gaussian primitives $\mathcal{G} = \{G_i\}_{i=1}^{n}$. Each Gaussian G_i is parameterized by its mean $\mu_i \in \mathbb{R}^3$ (position), covariance Σ_i (scale and rotation), base opacity o_i, and view-dependent color c_i represented by SH coefficients. Given a camera pose P_k, the Gaussian primitives are projected onto the image plane. The effective opacity that Gaussian G_i contributes at pixel p is its base opacity attenuated by the projected footprint:

$$\alpha_i(p) = o_i \, g_i(p), \qquad g_i(p) = \exp\Big(-\tfrac{1}{2}\,(p-\hat{\mu}_i)^\top \hat{\Sigma}_i^{-1}\,(p-\hat{\mu}_i)\Big), \tag{1}$$

where $g_i(p)$ is the Gaussian falloff evaluated at p, and $\hat{\mu}_i \in \mathbb{R}^2$ and $\hat{\Sigma}_i \in \mathbb{R}^{2\times 2}$ are respectively the projected mean and projected covariance of G_i on the image plane. The color $C_k(p)$ of pixel p is then computed by front-to-back alpha compositing over the Gaussians overlapping that pixel:

$$C_k(p) = \sum_{i=1}^{|\mathcal{N}_k(p)|} c_i \, \alpha_i(p) \prod_{j=1}^{i-1} \big(1-\alpha_j(p)\big), \tag{2}$$

where $\mathcal{N}_k(p) \subset \mathcal{G}$ denotes the set of Gaussians overlapping pixel p at view k, ordered from front to back by depth. Computing $C_k(p)$ over all pixels yields the rendered image $\hat{I}_k$. During training, the Gaussian parameters are optimized such that the rendered images $\{\hat{I}_k\}_{k=1}^{N}$ faithfully reconstruct the corresponding training images $\{I_k\}_{k=1}^{N}$.

3.2 Threat Model

We consider a practical threat model in which the attacker is either an asset creator or a third-party asset distributor who can modify and distribute pretrained 3DGS scenes. The attacker starts from a well-optimized 3DGS scene $\mathcal{G} = \{G_i\}_{i=1}^{n}$ and injects a set of adversarial Gaussians $\mathcal{G}_{\text{adv}} = \{G_{\text{adv},i}\}_{i=1}^{m}$, where $m \ll n$. The injected Gaussians are optimized by HiddenSplat to satisfy the desired view-specific behavior. The resulting poisoned scene is given by $\mathcal{G}^* = \mathcal{G} \cup \mathcal{G}_{\text{adv}}$, which is then distributed as a regular 3DGS asset. Victim users may unknowingly deploy the poisoned asset in downstream applications, such as autonomous driving, robotic navigation, and AR/VR systems.

For arbitrary non-target viewpoints, the poisoned scene $\mathcal{G}^*$ should preserve the original rendering behavior, producing images that are visually indistinguishable from those rendered by the clean scene. However, from an attacker-selected target viewpoint k_{tgt}, the injected Gaussians should reveal a designated target content I_{tgt}, such as a hidden message, logo, or malicious visual pattern. Formally, the desired rendering behavior at the target viewpoint can be described as:

$$\hat{I}^*_{k_{\text{tgt}}} = \mathcal{P}(I_{\text{tgt}}, \mathbf{M}) + (\mathbf{1}-\mathbf{M}) \odot \hat{I}_{k_{\text{tgt}}}, \tag{3}$$

where $\mathbf{M}$ denotes the spatial region occupied by the target content, $\odot$ represents element-wise multiplication, $\hat{I}_{k_{\text{tgt}}}$ denotes the clean rendering from the original scene, and $\mathcal{P}(I_{\text{tgt}}, \mathbf{M})$ denotes an operator that places the target pattern into the masked region of the target view. This formulation describes the attack objective: the target viewpoint should contain the injected content within the target region, while preserving the original scene outside that region.

Attack Settings. HiddenSplat supports both white-box and gray-box settings. In the white-box setting, the attacker has full access to the clean 3DGS scene $\mathcal{G}$ and directly optimizes $\mathcal{G}_{\text{adv}}$ to maximize target fidelity while maintaining visual stealthiness. In the gray-box setting, the attacker does not have access to the victim scene $\mathcal{G}$ during adversarial optimization. Instead, the adversarial Gaussians are optimized using an auxiliary scene $\mathcal{G}_{\text{aux}}$ and subsequently injected into the target scene.

Practicality. While existing poisoning and camouflage attacks (Lu et al., 2025; Hull et al., 2025) require influencing the asset creation process, HiddenSplat operates entirely post-training: the attacker only needs access to a released 3DGS asset without accessing the victim's training pipeline, training images, camera poses, or the rendering process. Moreover, the attack is purely additive: the original Gaussians remain frozen and unmodified, while the injected Gaussians are standard 3DGS primitives with no special parameters or auxiliary representation. The resulting poisoned asset can therefore be rendered by unmodified 3DGS viewers without an auxiliary decoder as in embedding-based approaches. Different practical use cases of HiddenSplat are presented in the appendix.

4 HIDDENSPLAT: METHODOLOGY

4.1 MOTIVATION: COVERAGE-STEALTH ANALYSIS

Before presenting the framework, we establish two analytical results that motivate HiddenSplat's design. First, the *target reconstruction capability* theorem shows that multiple low-opacity Gaussians along the same ray can approximate arbitrary target colors with controllable error. Second, the *non-target perturbation bound* theorem shows that when these Gaussians are spatially dispersed in non-target views, their rendering perturbation remains bounded. Together, they provide insights and guide our choice of Gaussian budget and stealth objectives.

Target Reconstruction Capability. Consider a target pixel p in the target view k_{tgt}. To reconstruct this pixel, the attacker can place $K(p)$ adversarial Gaussians along the ray passing through p, then optimize these Gaussians. The resulting rendered pixel color is:

$$C^*_{k_{\mathrm{tgt}}}(p) = \sum_{i=1}^{K(p)} c_i\, \alpha_i(p) \prod_{j=1}^{i-1}(1-\alpha_j(p)) + T\, C_{\mathrm{bg}}(p), \tag{4}$$

where $C_{\mathrm{bg}}(p)$ denotes the clean-scene color behind the adversarial Gaussians, and $T = \prod_{j=1}^{K(p)}(1-\alpha_j(p))$ is the remaining transmittance after compositing the injected Gaussians.

During the optimization process, the position of these adversarial Gaussians can change. If a Gaussian falls off its ray, it no longer contributes to the reconstruction of its target pixel (i.e., $g_i(p) \to 0$). Thus, we proactively enforce a lower bound on this attenuation: every adversarial Gaussian may change its position only within a limit such that it retains at least a fraction $\gamma \in (0, 1]$ of its base opacity, i.e., $g_i(p) \geq \gamma$ and hence $\alpha_i(p) \geq \gamma\, o_i$. We call γ the *opacity retention factor*.

Theorem 1 (Target reconstruction capability)**.** *Consider a target pixel p from the target view k_{tgt}. Suppose each adversarial Gaussian has base opacity at least $o_{\min}$ and retention factor γ. By placing $K(p)$ Gaussians along the ray passing through p and assigning them the target pixel's value, the rendered pixel can approximate the desired target color $c_{\mathrm{tgt}}(p)$ with error at most ϵ, provided that*

$$K(p) \;\geq\; \frac{\ln(\Delta_C/\epsilon)}{\ln\!\big(1/(1-\gamma\, o_{\min})\big)}, \tag{5}$$

where $\Delta_C = \big|C_{\mathrm{bg}}(p) - c_{\mathrm{tgt}}(p)\big|$ is the distance between the background and target colors.

The intuition behind this result is that although each adversarial Gaussian has limited individual influence due to its low opacity, alpha compositing accumulates the stacked Gaussians's opacity along the ray toward full coverage, allowing the Gaussians to approximate any target pixel color with arbitrarily small error. Applying this construction over all pixels in the target view enables the adversarial Gaussians to reconstruct an arbitrary target image with controllable precision.

Non-Target Stealthiness. When the camera moves away from the target viewpoint, the projected locations of these Gaussians become dispersed, causing only a small subset of Gaussians to contribute to each affected pixel. We characterize this behavior by indexing a non-target view according to its angular offset θ from k_{tgt}, and denote by $K'(\theta)$ the number of adversarial Gaussians overlapping a pixel p at that view. The next result bounds the Gaussian perturbation at normal views.

Theorem 2 (Non-target perturbation bound)**.** *Suppose every adversarial Gaussian has base opacity at most $o_{\max}$. For a non-target view k with angular offset θ, the rendering deviation introduced by the adversarial Gaussians satisfies*

$$\big|C^*_k(p) - C_k(p)\big| \;\leq\; K'(\theta)\, o_{\max}. \tag{6}$$

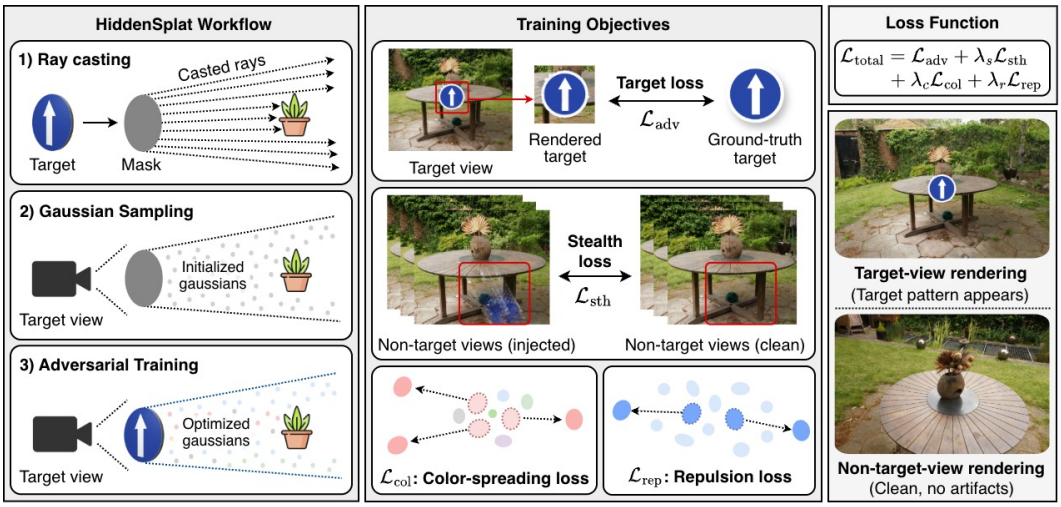


Figure 2: Overview of HiddenSplat. For each pixel within the target mask, HiddenSplat casts a ray and samples K adversarial Gaussians along the ray. These Gaussians are jointly optimized with four complementary objectives to achieve high-fidelity target reconstruction while preserving visual consistency across non-target views.

The bound remains small because $o_{\max}$ is constrained, while $K'(\theta)$ rapidly decreases as the camera view deviates from the target view: the injected Gaussians that are aligned along target-view rays become spatially dispersed under viewpoint changes, reducing the number of adversarial Gaussians that overlap the same non-target pixel. An empirical analysis of $K'(\theta)$ is given in the appendix.

Remark 1. *Theorem 1 and Theorem 2 motivate the design and training objectives of HiddenSplat. Theorem 1 characterizes the number of adversarial Gaussians required along each target ray to reconstruct the target color within a chosen error tolerance, which we use to set the per-ray Gaussian budget $K(p)$ at initialization. Theorem 2 bounds the perturbation at a non-target view by $K'(\theta)\, o_{\max}$, identifying the two quantities: (i) the max opacity of each adversarial Gaussian, which we constrain directly to $o_{\max}$, and (ii) the number $K'(\theta)$ of adversarial Gaussians a non-target ray traverses. We reduce the latter through the repulsion loss $\mathcal{L}_{\text{rep}}$, which spatially separates non-negligible Gaussians and thereby limits their overlap in non-target views. Since this bound is worst-case and depends on opacity alone, we further suppress the residual footprint through the stealth loss $\mathcal{L}_{\text{sth}}$, and reduce the color coherence it does not capture through the color-spreading loss $\mathcal{L}_{\text{col}}$.*

We note that Theorem 1 and Theorem 2 are only worst-case characterizations under the stated assumptions. In practice, the adversarial Gaussians can perform considerably better than these worst-case bounds after optimization. The budget $K(p)$ derived from Theorem 1 therefore serves as a conservative initialization; optionally, adaptive pruning presented in the appendix can further reduce the number of adversarial Gaussians at a modest cost of 3–5 dB in target PSNR. Section 5 shows that the optimized configurations achieve high target fidelity with low non-target perturbation, consistent with the derived bounds.

4.2 Framework Overview

As illustrated in Figure 2, HiddenSplat takes a pretrained 3DGS scene ($\mathcal{G}$ in white-box, or an auxiliary $\mathcal{G}_{\text{aux}}$ in gray-box), a target content I_{tgt}, and a target view k_{tgt}. It outputs a set of adversarial Gaussians $\mathcal{G}_{\text{adv}}$ that inject the target content I_{tgt} into the target viewpoint k_{tgt} while preserving non-target-view appearance. The framework consists of three stages as presented below.

Ray Casting. Given the target viewpoint, we first define a target mask $\mathbf{M}$ specifying the image region occupied by the target content I_{tgt}. For each pixel p inside the mask, we cast a ray from the target camera through the pixel. The free-space region before the first surface intersection is identified as the valid region for placing adversarial Gaussians. This ensures that the injected Gaussians can influence the target view without being occluded by the original scene geometry. Since scene depths vary across pixels, the available free-space interval can differ for different rays.

Gaussian Sampling. Along the ray of each target pixel p, we initialize $K(p)$ adversarial Gaussians at different depths within the available free space, where $K(p)$ is calculated based on Theorem 1. Their positions are determined using a Halton low-discrepancy sequence (Halton, 1960), which provides an approximately uniform distribution along each ray. Each Gaussian is initialized as a small, low-opacity spherical primitive with its target pixel color. During optimization, their positions, scales, opacities, and view-dependent colors are jointly optimized to achieve the desired target-view reconstruction while maintaining non-target stealthiness.

Adversarial Training. The Gaussians are optimized using four objectives described in Section 4.3. During optimization, Gaussian opacities are constrained within $[o_{\min}, o_{\max}]$, ensuring consistency with the analyses in Section 4.1. Then, the adversarial Gaussians are added to the original scene: $\mathcal{G}^* = \mathcal{G} \cup \mathcal{G}_{\text{adv}}$. This poisoned 3DGS scene reveals the target content at k_{tgt}, while preserving clean rendering from other views.

4.3 Training Objectives

Target Reconstruction Loss. The primary objective of HiddenSplat is to accurately reproduce the target content from the designated viewpoint. Therefore, we optimize the adversarial Gaussians by minimizing the discrepancy between the rendered image of the poisoned scene $\hat{I}^*_{k_{\text{tgt}}}$ and the desired target content I_{tgt} within the target region. Since pixels outside the target region should preserve the original scene appearance and do not contribute to target reconstruction, we compute the loss only within the target mask $\mathbf{M}$ to save computation:

$$\mathcal{L}_{\text{adv}} = \left\| \mathbf{M} \odot (\hat{I}^*_{k_{\text{tgt}}} - \mathcal{P}(I_{\text{tgt}}, \mathbf{M})) \right\|. \tag{7}$$

Non-Target Stealth Loss. While the theoretical analysis shows that low-opacity and geometric dispersion limit the influence of adversarial Gaussians from non-target viewpoints, we further suppress their remaining visual footprint. Specifically, for a set $\mathcal{V}$ of non-target views, we minimize the difference between the poisoned rendering $\hat{I}^*_k$ and the clean rendering $\hat{I}_k$ over only the pixels affected by adversarial Gaussians. Let $\mathbf{M}^{\text{adv}}_k$ denote the binary mask of pixels influenced by the projected adversarial Gaussians at view k. The non-target stealth loss is defined as:

$$\mathcal{L}_{\text{sth}} = \frac{1}{|\mathcal{V}|} \sum_{k \in \mathcal{V}} \left\| \mathbf{M}^{\text{adv}}_k \odot (\hat{I}^*_k - \hat{I}_k) \right\|_1. \tag{8}$$

We sample $\mathcal{V}$ not only from the original training views but also from interpolated viewpoints around the target view k_{tgt}. Enforcing stealthiness on these near-target views prevents the injected content from leaking into neighboring viewpoints, ensuring that it becomes visible only within a narrow angular region around the designated target viewpoint.

Repulsion Loss. According to Theorem 2, the non-target perturbation at a pixel depends on the accumulated opacity of adversarial Gaussians overlap with it. Therefore, clustering many adversarial Gaussians in a small spatial region increases the likelihood that non-target rays overlap multiple Gaussians, causing larger perturbation. To mitigate this effect, we introduce a repulsion loss that encourages adversarial Gaussians with non-negligible opacity to remain spatially separated:

$$\mathcal{L}_{\text{rep}} = \frac{1}{\kappa \, |\mathcal{G}_{\text{adv}}|} \sum_i \sum_{j \in \mathcal{N}_i} \frac{o_i \, o_j}{\|\mu_i - \mu_j\| + \delta}, \tag{9}$$

where $\mathcal{N}_i$ denotes the set of κ nearest neighboring adversarial Gaussians of Gaussian i, and δ is a small constant for numerical stability.

Color-Spreading Loss. Although Theorem 2 bounds the worst-case perturbation using opacity alone, the perceptual visibility of the perturbation also depends on the colors of the overlapping adversarial Gaussians. In particular, neighboring Gaussians with similar colors are more likely to reinforce one another and form coherent visual artifacts when projected onto the same pixels. To discourage such behavior, we introduce the following color-spreading loss:

$$\mathcal{L}_{\text{col}} = \frac{1}{\kappa \, |\mathcal{G}_{\text{adv}}|} \sum_i \sum_{j \in \mathcal{N}_i} \frac{\text{sim}(c_i, c_j)}{\|\mu_i - \mu_j\| + \delta}, \tag{10}$$

where $\mathrm{sim}(c_i, c_j) = \max\big(0,\ 1 - \|c_i - c_j\|\big)$ measures the color similarity between two normalized Gaussian colors, and $\mathcal{N}_i$ denotes the κ nearest neighboring adversarial Gaussians of Gaussian i.

In effect, this loss discourages nearby Gaussians from sharing a color, so similar colors tend to spread across different depths along the rays. A ray from a non-target view is then less likely to pass through Gaussians of similar color, reducing the chance that their contributions reinforce one another into visible artifacts.

4.4 Overall Training Process

Combining the four objectives, the overall optimization loss is:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{adv}} + \lambda_{\text{s}}\,\mathcal{L}_{\text{sth}} + \lambda_{\text{c}}\,\mathcal{L}_{\text{col}} + \lambda_{\text{r}}\,\mathcal{L}_{\text{rep}}, \tag{11}$$

where λ_{s}, λ_{c}, and λ_{r} balance the contributions of the three regularization terms. The target reconstruction loss $\mathcal{L}_{\text{adv}}$ serves as the primary optimization objective, while the remaining losses regularize the adversarial Gaussians to improve their stealthiness.

During optimization, only the adversarial Gaussians $\mathcal{G}_{\text{adv}}$ are updated, while the original scene $\mathcal{G}$ are kept frozen. Besides the opacity constraints, we additionally constrain Gaussian scales to remain below a maximum radius, preventing the occurrence of excessively large Gaussians that could introduce visible artifacts.

5 Experiments

5.1 Experimental Setup

Datasets. We evaluate HiddenSplat on three widely used 3D scene reconstruction benchmarks: Mip-NeRF360 (Barron et al., 2022) containing unbounded indoor and outdoor scenes, Tanks & Temples (Knapitsch et al., 2017) with large-scale real-world scenes, and Free (Wang et al., 2023) containing forward-facing scenes with complex geometry.

Evaluation Metrics. We evaluate HiddenSplat along two complementary axes: **target fidelity**, which measures how accurately the hidden content is reconstructed at the target view, and **stealthiness**, which measures how little the poisoned asset deviates from the clean scene at non-target views. For target fidelity, we compare the rendered target region against the target pattern using PSNR, SSIM, LPIPS, and CLIPs (Radford et al., 2021) similarity. For stealthiness, we compare poisoned and clean renderings over the affected region using PSNR, SSIM, and LPIPS. We further report the maximum and mean of the adversarial opacity accumulated in each affected pixel (α_{max} and α_{mean}), capturing the worst-case and average of adversarial perturbation at non-target views. Because only a small footprint of pixels is perturbed at non-target views, we compute the stealth metrics within the adversarial-footprint mask rather than over the full frame, preventing the vast majority of unaffected pixels from dominating the score and inflating apparent stealth.

Baselines. We compare HiddenSplat against four representative baselines that align with our threat model: (1) IPA-NeRF (Jiang et al., 2024), a view-specific attack originally designed for NeRF; (2) StealthAttack (Ke et al., 2025), which conceals adversarial Gaussians by placing them in low-density regions of the scene; (3) StealthAttack-Adapt, a version adapted by embedding target content at the same viewpoint with other frameworks instead of concealing them in low-density regions; and (4) DenseLayer, a variant that places adversarial Gaussians into $L = 3$ dense layers near the scene surface instead of distributing them sparsely throughout the free space. Other frameworks like GS-Hider (Zhang et al., 2024) and GuardSplat (Chen et al., 2025c) employ totally different threat models in which watermarks or hidden messages are embedded into 3DGS via additional encoder and decoder modules; thus, they are not directly comparable to HiddenSplat.

Implementation Settings. Regarding Gaussian parameters, we set $o_{\max} = 0.006$, $o_{\min} = 0.005$, $\gamma = 0.85$, and the maximum Gaussian radius to 1 pixel. We use $\epsilon = 0.15$ to compute the number of adversarial Gaussians $K(p)$ for each pixel based on Theorem 1. During adversarial training, we optimize the adversarial Gaussians for 500 steps using Adam optimizer. We update Gaussian positions, scales, opacities, and colors with learning rates of 1.6e-4, 5e-3, 5e-3, and 5e-2, respectively. The loss weights are set to $\lambda_{\text{s}} = 1.0$, $\lambda_{\text{r}} = 1.5$, and $\lambda_{\text{c}} = 0.5$, which are derived based on standard empirical hyperparameter tuning. For repulsion and color-spreading losses, we consider $\kappa = 8$

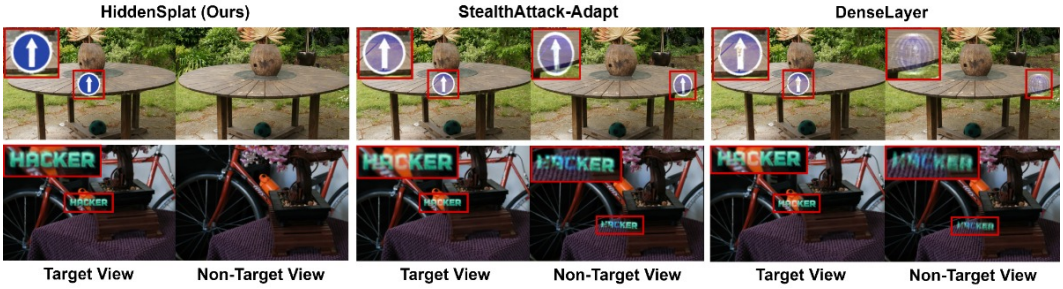


Figure 3: Qualitative comparison between HiddenSplat and 3DGS baselines. HiddenSplat successfully reconstructs the target pattern with high fidelity and effectively hides it at non-target views.

Table 1: Comparison of attack effectiveness between HiddenSplat and the baseline methods.

Dataset	Mip-NeRF 360				Tanks & Temples				Free			
Method	PSNR↑	SSIM↑	LPIPS↓	CLIP↑	PSNR↑	SSIM↑	LPIPS↓	CLIP↑	PSNR↑	SSIM↑	LPIPS↓	CLIP↑
IPA-NeRF	15.79	0.566	0.632	0.693	13.22	0.600	0.723	0.623	14.02	0.412	0.625	0.589
DenseLayer	23.35	0.962	**0.057**	0.836	21.65	0.821	0.265	0.765	22.68	0.812	0.098	0.812
StealthAttack	27.15	0.959	0.063	0.838	22.65	0.812	0.386	0.788	25.73	0.915	**0.082**	0.819
StealthAttack-Adapt	23.58	0.868	0.163	0.805	20.35	0.723	0.324	0.726	21.65	0.785	0.086	0.789
HiddenSplat (Ours)	**33.59**	**0.978**	0.061	**0.890**	**31.23**	**0.955**	**0.110**	**0.832**	**32.40**	**0.978**	0.083	**0.862**

nearest neighbors. Each training iteration samples 5 non-target views along with 3 near-target views. More details about our implementation and parameter choices are presented in the appendix.

5.2 Attack Effectiveness

Table 1 compares HiddenSplat with existing baselines on target-view reconstruction. HiddenSplat consistently achieves the strongest attack performance across all three datasets. For example, on Mip-NeRF360, HiddenSplat achieves 33.59 dB PSNR and 0.978 SSIM, substantially outperforming the baseline methods. Similar improvements are observed on Tanks & Temples and Free, demonstrating that HiddenSplat generalizes well across diverse 3D scenes. Notably, StealthAttack-Adapt performs worse than the original StealthAttack: because the target content is embedded in a normal-density region rather than a low-density one, the optimization is challenging and attack quality drops. DenseLayer likewise attains low attack capability, indicating that concentrating adversarial Gaussians in a few dense layers is insufficient to reconstruct the target content faithfully. Some generation results are visualized in Figure 3, confirming our observations.

5.3 Stealth Preservation

Table 2 compares the stealthiness of HiddenSplat against the baselines on both training and test views. HiddenSplat consistently achieves the highest visual quality, with PSNR above 42 dB, SSIM exceeding 0.99, and LPIPS below 0.02 on both view sets, indicating that the poisoned scene is nearly indistinguishable from the clean scene. Moreover, HiddenSplat produces the lowest maximum and

Table 2: Comparison of stealthiness between HiddenSplat and the baselines.

	Training Views					Test Views				
Method	PSNR↑	SSIM↑	LPIPS↓	$\alpha_{\max}$↓	α_{mean}↓	PSNR↑	SSIM↑	LPIPS↓	$\alpha_{\max}$↓	α_{mean}↓
IPA-NeRF	22.13	0.826	0.112	N/A	N/A	23.66	0.786	0.126	N/A	N/A
DenseLayer	16.71	0.741	0.164	0.169	0.101	15.23	0.714	0.166	0.148	0.121
StealthAttack	30.21	0.863	0.201	0.100	0.089	28.96	0.814	0.233	0.115	0.105
StealthAttack-Adapt	18.82	0.763	0.269	0.128	0.113	17.65	0.726	0.288	0.122	0.156
HiddenSplat (Ours)	**42.33**	**0.995**	**0.015**	**0.082**	**0.017**	**41.23**	**0.993**	**0.016**	**0.078**	**0.019**

Table 3: Cross-scene generalization (gray-box setting). T&T denotes Tanks & Temples, while 360 denotes Mip-NeRF360.

Scenes		Target View - Attack Effectiveness				Test Views - Stealthiness				
Source	Target	PSNR↑	SSIM↑	LPIPS↓	CLIP↑	PSNR↑	SSIM↑	LPIPS↓	$\alpha_{\max}$↓	α_{mean}↓
Garden (360)	Bicycle (360)	29.148	0.964	0.082	0.812	36.055	0.988	0.025	0.095	0.028
Kitchen (360)	Bonsai (360)	28.650	0.921	0.091	0.841	34.858	0.991	0.026	0.070	0.020
Room (360)	Lab (Free)	28.289	0.950	0.056	0.906	34.097	0.996	0.021	0.041	0.013
Road (Free)	Truck (T&T)	27.524	0.952	0.094	0.782	32.456	0.988	0.040	0.099	0.019

mean adversarial opacity, confirming that the injected Gaussians introduce only negligible perturbations at non-target viewpoints. In contrast, the baselines exhibit substantially lower rendering quality and higher opacity leakage, resulting in noticeable visual artifacts (illustrated in Figure 3). DenseLayer and StealthAttack already use degree-3 SH under an explicit non-target stealth objective, and both fail to hide the content, indicating the default SH alone is not enough to ensure stealthiness. HiddenSplat achieves stealth geometrically instead, via dispersion of the adversarial Gaussians when the viewpoint moves away from the target view. These results demonstrate that HiddenSplat effectively preserves the appearance of the original 3DGS scene.

5.4 Cross-Scene Generalization – Gray-Box Setting

Table 3 evaluates HiddenSplat in the gray-box optimization setting, where the adversarial Gaussians are optimized on an auxiliary source scene and directly transferred to a different target scene (implementation details are in the appendix). Despite having no access to the target scene during optimization, HiddenSplat consistently achieves strong attack effectiveness, with PSNR up to 29 dB, SSIM exceeding 0.92, and CLIP similarity up to 0.91 across all scene pairs. At the same time, the transferred Gaussians remain highly stealthy, yielding PSNR above 32 dB, SSIM around 0.99, LPIPS below 0.04, and low opacity leakage on the target scenes. These results demonstrate that HiddenSplat generalizes well across diverse scenes and remains effective even in the more practical gray-box settings.

5.5 Ablation Study

Table 4 reports the contribution of each loss term. Removing $\mathcal{L}_{\text{sth}}$ causes the largest drop in stealth, reducing the stealth PSNR from 42.33 dB to 30.58 dB. Removing either $\mathcal{L}_{\text{rep}}$ or $\mathcal{L}_{\text{col}}$ also degrades both target fidelity and stealth, confirming that these two regularizers improve attack quality while suppressing non-target artifacts. Overall, all proposed loss terms contribute importantly to the overall effectiveness of HiddenSplat. Detailed sensitivity analysis of these losses are provided in the appendix.

Table 4: Ablation study on loss terms.

Settings	Target Fidelity		Stealth	
	PSNR↑	SSIM↑	PSNR↑	SSIM↑
Full	33.59	0.978	42.33	0.995
W/o $\mathcal{L}_{\text{sth}}$	30.23	0.953	30.58	0.965
W/o $\mathcal{L}_{\text{rep}}$	27.89	0.964	38.56	0.982
W/o $\mathcal{L}_{\text{col}}$	27.55	0.963	32.48	0.992

6 Conclusion

We presented HiddenSplat, a novel framework for injecting view-specific visual content into pretrained 3D Gaussian Splatting scenes through a set of optimized "invisible" Gaussians. By exploiting the geometric accumulation of low-opacity Gaussians at a designated viewpoint while minimizing their influence elsewhere, HiddenSplat achieves high-fidelity target reconstruction without perceptible degradation from non-target views. Extensive experiments demonstrate its effectiveness in both white-box and gray-box settings across diverse scenes and target patterns. Beyond exposing a new integrity threat for shareable 3DGS assets, our work also highlights opportunities for controlled view-specific content embedding, such as watermarking and covert communication. We hope HiddenSplat motivates future research on securing 3DGS assets against post-training manipulation and on developing robust defenses for trustworthy 3D scene representations.

AI USE STATEMENT

We used LLMs solely to polish grammar and improve the readability of selected sentences. All technical content, ideas, experiments, and analyses were developed by the authors. The authors take full responsibility for the content of this paper.

ETHICS STATEMENT

This paper reveals a novel attack surface in 3DGS assets through which arbitrary content can be embedded into 3DGS scenes while remaining imperceptible from non-target viewpoints. Although the attack could be misused to compromise the integrity of shared or externally sourced 3DGS assets, we believe that systematically exposing such vulnerabilities is important for motivating the development of effective defenses. Our work aims to raise awareness of this emerging attack surface and facilitate future research on securing 3DGS-based systems.

REPRODUCIBILITY STATEMENT

Our method is fully specified through the training objectives in Section 4.3, with complete proofs of Theorems 1 and 2 provided in Appendix B. All implementation details required to reproduce our experiments are provided in Section 5 and Appendix A. All three datasets used in our experiments are publicly available.

REFERENCES

Jonathan T Barron, Ben Mildenhall, Dor Verbin, Pratul P Srinivasan, and Peter Hedman. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5460–5469. IEEE, 2022.

Jianchuan Chen, Jingchuan Hu, Gaige Wang, Zhonghua Jiang, Tiansong Zhou, Zhiwen Chen, and Chengfei Lv. Taoavatar: Real-time lifelike full-body talking avatars for augmented reality via 3d gaussian splatting. In *CVPR*, pp. 10723–10734, 2025a.

Timothy Chen, Ola Shorinwa, Joseph Bruno, Aiden Swann, Javier Yu, Weijia Zeng, Keiko Nagami, Philip Dames, and Mac Schwager. Splat-nav: Safe real-time robot navigation in gaussian splatting maps. *IEEE Transactions on Robotics*, 2025b.

Zixuan Chen, Guangcong Wang, Jiahao Zhu, Jianhuang Lai, and Xiaohua Xie. Guardsplat: efficient and robust watermarking for 3d gaussian splatting. In *CVPR*, pp. 16325–16335, 2025c.

John H Halton. On the efficiency of certain quasi-random sequences of points in evaluating multi-dimensional integrals. *Numerische Mathematik*, 2(1):84–90, 1960.

Georg Hess, Carl Lindström, Maryam Fatemi, Christoffer Petersson, and Lennart Svensson. Splatad: Real-time lidar and camera rendering with 3d gaussian splatting for autonomous driving. In *CVPR*, pp. 11982–11992, 2025.

Matthew Hull, Haoyang Yang, Pratham Mehta, Mansi Phute, Aeree Cho, Haorang Wang, Matthew Lau, Wenke Lee, Wilian Lunardi, Martin Andreoni, et al. Complicitsplat: Downstream models are vulnerable to blackbox attacks by 3d gaussian splat camouflages. *arXiv preprint arXiv:2508.11854*, 2025.

Wenxiang Jiang, Hanwei Zhang, Shuo Zhao, Zhongwen Guo, and Hao Wang. Ipa-nerf: Illusory poisoning attack against neural radiance fields. *arXiv preprint arXiv:2407.11921*, 2024.

Wenxiang Jiang, Hanwei Zhang, Weigang Wang, Zhongwen Guo, Tianao Zhang, and Hao Wang. Mpam-3dgs: Multi-parametric adversarial manipulation for 3d gaussian splatting. In *ICASSP*, pp. 1–5. IEEE, 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 *ICCV*, pp. 27400–27411, 2025.

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 *CVPR*, pp. 33227–33236, 2026.

Tianrui Lou, Xiaojun Jia, Siyuan Liang, Jiawei Liang, Ming Zhang, Yanjun Xiao, and Xiaochun Cao. 3d gaussian splatting driven multi-view robust physical adversarial camouflage generation. In *CVPR*, pp. 28752–28762, 2025.

Jiahao Lu, Yifan Zhang, Qiuhong Shen, Xinchao Wang, and Shuicheng Yan. Poison-splat: Computation cost attack on 3d gaussian splatting. In *ICLR*, volume 2025, pp. 98599–98630, 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. Advsplat: Adversarial attacks on feed-forward gaussian splatting models. *arXiv preprint arXiv:2603.23686*, 2026.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pp. 8748–8763. PmLR, 2021.

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

Peng Wang, Yuan Liu, Zhaoxi Chen, Lingjie Liu, Ziwei Liu, Taku Komura, Christian Theobalt, and Wenping Wang. F2-nerf: Fast neural radiance field training with free camera trajectories. In *CVPR*, pp. 4150–4159, 2023.

Yuxuan Wu, Lei Pan, Wenhua Wu, Guangming Wang, Yanzi Miao, Fan Xu, and Hesheng Wang. Rl-gsbridge: 3d gaussian splatting based real2sim2real method for robotic manipulation learning. In *ICRA*, pp. 192–198. IEEE, 2025.

Hongjia Zhai, Xiyu Zhang, Boming Zhao, Hai Li, Yijia He, Zhaopeng Cui, Hujun Bao, and Guofeng Zhang. Splatloc: 3d gaussian splatting-based visual localization for augmented reality. *IEEE Transactions on Visualization and Computer Graphics*, 2025.

Xuanyu Zhang, Jiarui Meng, Runyi Li, Zhipei Xu, Yongbing Zhang, and Jian Zhang. Gs-hider: Hiding messages into 3d gaussian splatting. *NeurIPS*, 37:49780–49805, 2024.

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 *CVPR*, pp. 21634–21643, 2024.

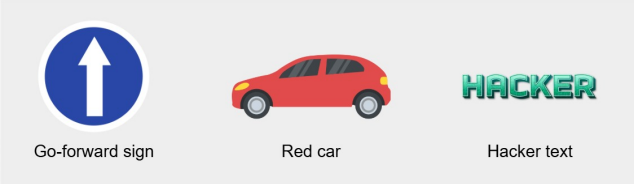


Figure 4: Three target images used in our experiments.

A Implementation Details

Target and Viewpoint Selection. Our experiments use three target patterns, shown in Figure 4, covering both visual and textual content. Since HiddenSplat allows arbitrary target-view selection, we choose the first validation view of each scene as the target view and exclude it from the evaluation set. For the multi-view attack, the target views are randomly sampled from the validation set. All reported results are averaged over three runs using the three target patterns. Experiments are conducted on an NVIDIA RTX 6000 Ada GPU.

Details of Gaussian Constraints. During optimization, we constrain the maximum Gaussian radius to 1 pixel to prevent adversarial Gaussians from growing excessively large and trivially covering the target region, which accelerates optimization at the beginning but ultimately degrades reconstruction quality. Based on the chosen opacity retention factor γ, we constrain the Gaussian positions to prevent them from drifting too far away from their assigned rays (which causes visible artifacts at non-target views, while not contributing to target reconstruction at the target view). Specifically, the selected $\gamma = 0.85$ allows each Gaussian to deviate up to $r = \sqrt{-2\ln\gamma} \approx 0.57$ standard deviations (Mahalanobis units) from the target ray. We also fix Gaussian rotations to avoid the formation of excessively thin, elongated Gaussians that can produce noticeable rendering artifacts.

Number of Adversarial Gaussians. As presented in Section 5.1, the number of adversarial Gaussians $K(p)$ for each pixel p is initialized based on the result in Theorem 1 and Eq. (5). In our experiments with $\epsilon = 0.15$, $\gamma = 0.85$, and $o_{\min} = 0.005$, we find that the average number of Gaussians per ray is approximately 120 (with an average Δ_C of 0.25), varying slightly among different target patterns and scenes. However, Theorem 1 only provides a conservative worst-case assessment based on the minimum allowed opacity $o_{\min}$. In practice, the opacity of these Gaussians are updated during optimization, which can yield a similar reconstruction quality with fewer adversarial Gaussians. With the adaptive pruning mechanism presented below, we can significantly reduce the number of Gaussians with an acceptable trade-off for target reconstruction quality.

Adaptive Pruning. Optionally, the number of adversarial Gaussians can be reduced during optimization by adaptive pruning, at a modest cost in attack effectiveness. Note that all reported results in Section 5 are obtained without pruning; we describe the variant here for completeness.

Compared with the standard adaptive density control (ADC) of 3DGS (Kerbl et al., 2023), we retain only the pruning operation and disable both cloning and splitting. Splitting is designed to resolve sharp surfaces by subdividing a primitive that under-fits the local geometry. Our adversarial Gaussians instead form a volumetric stack along each target ray, and splitting such a stack merely fragments it: the children are displaced off the ray, where they no longer satisfy the retention constraint $g_i(p) \geq \gamma$ and therefore stop contributing to their target pixel. Cloning is likewise unnecessary, since the per-pixel budget $K(p)$ is fixed at initialization rather than grown. Pruning, by contrast, is well matched to our setting. Because opacities are clamped to $[o_{\min}, o_{\max}]$, a Gaussian that the stealth loss would like to suppress further accumulates at the lower bound: convergence to $o_{\min}$ is a signal that the optimizer has already judged the primitive redundant. We therefore remove adversarial Gaussians whose opacity has settled at the floor ($o_i \leq o_{\min}$) every 100 iterations.

As reported in Appendix C.3, this reduces the injected primitive count significantly, at a cost of 3–5 dB in target PSNR. Pruning therefore offers a practical trade-off when minimizing the attack footprint and overhead matters more than maximizing target fidelity.

Gray-Box Attack Setting. In the gray-box setting where adversarial Gaussians are optimized on an auxiliary scene, the target view is an attacker-chosen camera pose in the recipient scene's coordinate

frame; the payload region is the image-centered bounding box at that pose. The injected Gaussians are not transplanted in absolute world coordinates but re-registered to this camera: their positions are obtained by casting rays through the chosen view and placing K Gaussians per pixel in the free space in front of the scene, where the free depth is measured per ray from the recipient scene's rendered geometry. Consequently the method adapts to occupancy automatically.

This gray-box setting is more practical as the computationally intensive adversarial optimization is performed without access to the victim scene. Only the subsequent injection step, which is quick and simple, requires access to the target scene. This separation not only simplifies the attack pipeline but also minimizes the attacker's exposure to users or defenders during optimization.

B PROOFS

B.1 PROOF OF THEOREM 1 (TARGET RECONSTRUCTION CAPABILITY)

Proof. Consider a target pixel p in view k_{tgt}, and assume that $K(p)$ adversarial Gaussians are placed along the corresponding ray in front of the clean scene. The alpha-compositing process can be decomposed into the contribution from the adversarial Gaussians and the remaining contribution from the original scene:

$$C^*_{k_{\text{tgt}}}(p) = \underbrace{\sum_{i=1}^{K(p)} c_i\, \alpha_i(p) \prod_{j=1}^{i-1}(1-\alpha_j(p))}_{C_{\text{adv}}} \; + \; \underbrace{T\, C_{\text{bg}}(p)}_{C_{\text{clean}}}, \tag{12}$$

where $T = \prod_{j=1}^{K(p)}(1-\alpha_j(p))$ denotes the residual transmittance after the adversarial Gaussians, and $C_{\text{bg}}(p) \in [0,1]$ is the color contributed by the clean scene behind the injected Gaussians.

For target-view analysis, the adversarial Gaussians along the ray p are assigned the target pixel color to reconstruct that pixel. Thus, we replace every c_i in C_{adv} with $c_{\text{tgt}}(p)$, resulting in:

$$C_{\text{adv}} = \sum_{i=1}^{K(p)} c_{\text{tgt}}(p)\alpha_i(p) \prod_{j=1}^{i-1}(1-\alpha_j(p)) = c_{\text{tgt}}(p) \sum_{i=1}^{K(p)} \alpha_i(p) \prod_{j=1}^{i-1}(1-\alpha_j(p)). \tag{13}$$

Using the standard telescoping identity of alpha compositing, we have:

$$\sum_{i=1}^{K(p)} \alpha_i(p) \prod_{j=1}^{i-1}(1-\alpha_j(p)) = 1 - \prod_{i=1}^{K(p)}(1-\alpha_i(p)) = 1 - T. \tag{14}$$

Substituting Eq. (14) into Eq. (13), we obtain:

$$C_{\text{adv}} = c_{\text{tgt}}(p)(1-T). \tag{15}$$

Substituting C_{adv} from Eq. (15) into Eq. (12), the reconstruction error is derived:

$$\underbrace{\left|C^*_{k_{\text{tgt}}}(p) - c_{\text{tgt}}(p)\right|}_{\text{reconstruction error}} = T\left|C_{\text{bg}}(p) - c_{\text{tgt}}(p)\right| = T\Delta_C. \tag{16}$$

Since each Gaussian satisfies the retention constraint $\alpha_i(p) \geq \gamma\, o_i$ and has base opacity $o_i \geq o_{\min}$, we have $\alpha_i(p) \geq \gamma\, o_{\min}$ for every i. Thus, we have:

$$T = \prod_{j=1}^{K(p)} (1-\alpha_j(p)) \leq (1-\gamma\, o_{\min})^{K(p)}. \tag{17}$$

Combining Eqs. (16) and (17), the reconstruction error is bounded:

$$\left|C^*_{k_{\text{tgt}}}(p) - c_{\text{tgt}}(p)\right| \leq \Delta_C (1-\gamma\, o_{\min})^{K(p)}. \tag{18}$$

To guarantee a reconstruction error no larger than ϵ, i.e., $\left|C^*_{k_{\text{tgt}}}(p) - c_{\text{tgt}}(p)\right| \leq \epsilon$, it is sufficient that $\Delta_C(1-\gamma\, o_{\min})^{K(p)} \leq \epsilon$. Taking logarithms on both sides gives:

$$K(p) \geq \frac{\ln(\Delta_C/\epsilon)}{\ln\big(1/(1-\gamma\, o_{\min})\big)}.$$

Hence, the target pixel can be reconstructed with arbitrary precision by adjusting the number of overlapped adversarial Gaussians. Since the bound applies independently to each target pixel, it demonstrates that overlapped adversarial Gaussians have sufficient capacity to reconstruct arbitrary target patterns at the target viewpoint. □

B.2 Proof of Theorem 2 (Non-Target Stealthiness)

Proof. Consider a pixel p from a non-target view k. Let $\mathcal{A}_k(p)$ denote the set of adversarial Gaussians overlapping this pixel, with $|\mathcal{A}_k(p)| = K'(\theta)$. We compare the clean rendering $C_k(p)$ with the poisoned rendering $C^*_k(p)$ by introducing the adversarial Gaussians one by one according to their depth order. When inserting a Gaussian $G_{\text{adv},i}$, it changes the current composite color by:

$$\Delta_i = T'_i\, \alpha_i(p)\, \big(c'_i - C_{\text{behind},i}(p)\big), \tag{19}$$

where $T'_i \leq 1$ is the transmittance before reaching this Gaussian, c'_i and $\alpha_i(p)$ are its color and effective opacity at p, and $C_{\text{behind},i}(p)$ is the accumulated color of the scene behind the Gaussian.

For simplicity, we analyze each color channel independently. Assuming normalized color values in $[0, 1]$, the maximum difference between two colors is bounded by 1:

$$\left|c'_i - C_{\text{behind},i}(p)\right| \leq 1. \tag{20}$$

Together with $T'_i \leq 1$, it yields:

$$|\Delta_i| = T'_i\, \alpha_i(p)\, \left|c'_i - C_{\text{behind},i}(p)\right| \leq \alpha_i(p). \tag{21}$$

The total difference between the poisoned and clean renderings is obtained by accumulating the perturbations introduced by all adversarial Gaussians:

$$C^*_k(p) - C_k(p) = \sum_{i \in \mathcal{A}_k(p)} \Delta_i, \tag{22}$$

where each Δ_i is measured with respect to the intermediate composite after inserting the previous Gaussians.

Applying the triangle inequality gives:

$$\left|C^*_k(p) - C_k(p)\right| \;\leq\; \sum_{i \in \mathcal{A}_k(p)} |\Delta_i|. \tag{23}$$

Combining Eqs. (21) and (23), we obtain:

$$\left|C^*_k(p) - C_k(p)\right| \;\leq\; \sum_{i \in \mathcal{A}_k(p)} \alpha_i(p). \tag{24}$$

Since $\alpha_i(p) \leq o_i \leq o_{\max}$ and $|\mathcal{A}_k(p)| = K'(\theta)$, the perturbation is bounded as:

$$\left|C^*_k(p) - C_k(p)\right| \;\leq\; K'(\theta)\, o_{\max}. \tag{25}$$

□

C Extended Experiments

C.1 Near-Target-View Stealthiness

Although HiddenSplat is designed to reveal the target content only at a designated viewpoint, a small amount of leakage may still occur from nearby viewpoints, potentially degrading clean-view

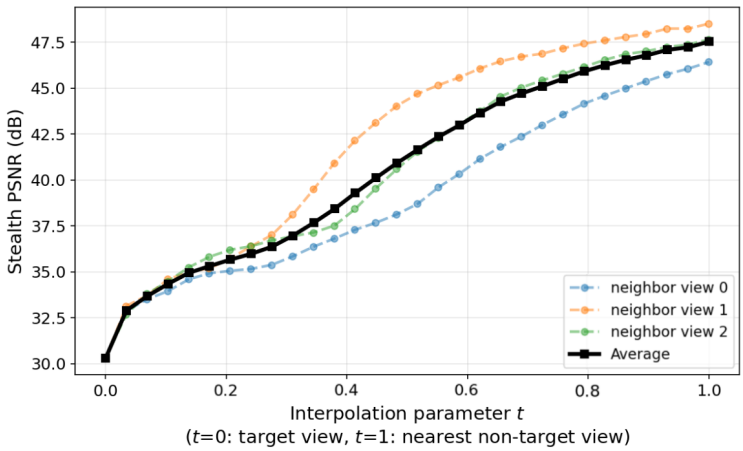


Figure 5: Stealth improvement from the target view towards the nearest non-target views.

rendering. To quantify this effect, we identify the training view nearest to the target view and uniformly interpolate intermediate viewpoints between them. Figure 5 plots the stealth PSNR along this trajectory. As the camera moves away from the target view, the stealth PSNR rapidly increases, indicating that the leaked content quickly disappears. This demonstrates that the influence of the adversarial Gaussians is confined to a very small neighborhood around the target viewpoint, further validating the stealthiness of HiddenSplat.

C.2 Number of Overlapped Gaussians at Non-Target Views

Theorem 2 bounds the non-target rendering deviation by $K'(\theta)\, o_{\max}$, where $K'(\theta)$ is the number of adversarial Gaussians a ray encounters at a view with angular offset θ from the target. We measure $K'(\theta)$ directly to show that it collapses as the camera moves away from the target view. In this test, a Gaussian is counted as overlapping a ray if its projected center lies within a Mahalanobis distance ρ of that ray, i.e. within a fraction ρ of the Gaussian's footprint size; we set $\rho = 0.5$ in our test. $K'(\theta)$ is then the average of this count over the pixels the adversarial cloud occupies.

Figure 6 shows a steep, monotone decay of $K'(\theta)$ as the angular offset θ grows. This count almost levels off at ≈ 5 beyond $\theta = 6^\circ$. Since $o_{\max}$ is fixed, the deviation budget $K'(\theta)\, o_{\max}$ shrinks in proportion, becoming negligible at the non-target viewpoints. This empirically confirms that the injected adversarial Gaussians disperse rapidly under viewpoint changes, weakening the perturbation everywhere except in the immediate neighborhood of the target view. With an average of about 5 adversarial Gaussians overlapping each ray at a non-target view, the resulting perturbation is bounded by $K'(\theta)\, o_{\max} = 5 \times 0.006 = 0.03$, i.e. at most 3% of the full $[0, 1]$ intensity range. This explains why the injected content remains imperceptible from non-target viewpoints. Note that the reported results are an average across the target images and scenes.

C.3 Overhead Analysis

In this experiment, we report the computational overhead introduced by HiddenSplat. Specifically, we choose the go-forward sign as the target image, and a target size of 100×100 pixels. Applying masking to the target image, it covers 5,565 pixels. As reported in Table 5, the attack injects $667,780$ adversarial Gaussians into a scene containing 5.04 million original Gaussians, increasing the total number of primitives by 13.23%.

With periodical adaptive pruning presented in Appendix A, the number of adversarial Gaussians reduces further to just 102,502 (2.03% of the original scene) with the cost of ~ 4 dB in target PSNR. The added Gaussians increase the total render time by 0.08 ms and yield an additional training cost of 1.28 minutes. These results show that HiddenSplat incurs only a reasonable computational overhead while remaining practical for real-time 3DGS rendering.

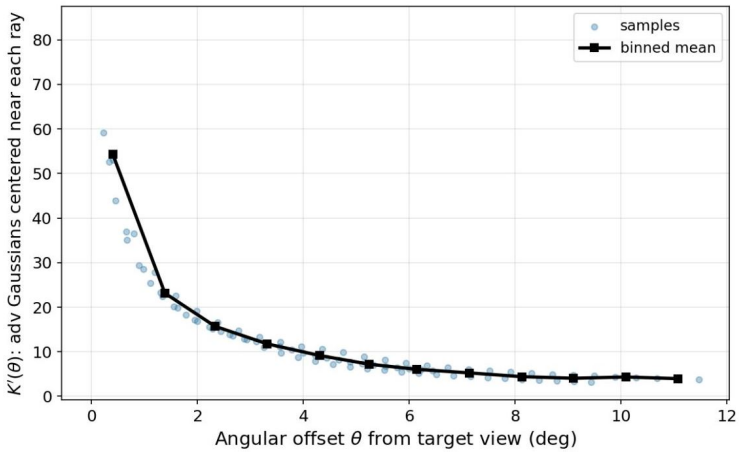


Figure 6: Adversarial ray occupancy $K'(\theta)$ versus angular offset θ from the target view.

Table 5: Overhead analysis of HiddenSplat with and without adaptive pruning.

	Original Scene	Adv. Gaussians	Pruning Applied
# Gaussians	5,044,801	667,780	102,502
Render time	4.55 ms	+0.27 ms	+0.08 ms
Training time	35.2 min	+3.3 min	+1.28 min
Target PSNR	N/A	33.59 dB	29.12 dB
Stealth PSNR	N/A	41.23 dB	41.86 dB

C.4 Multi-View Injection

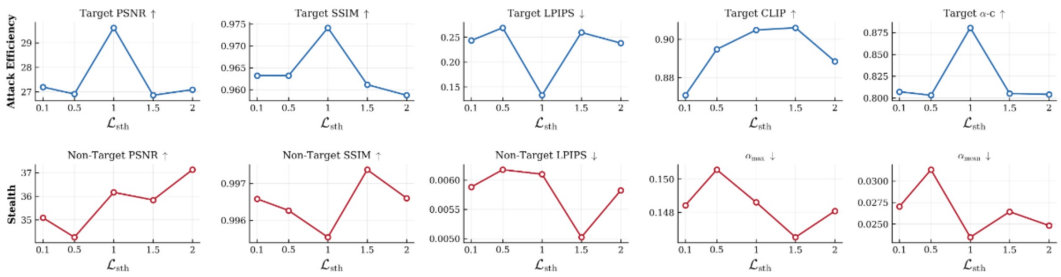
We further evaluate whether HiddenSplat can simultaneously inject target contents into multiple target viewpoints. As shown in Table 6, HiddenSplat consistently achieves high attack effectiveness and stealth when the number of target views increases from two to four. Across all settings, the target-view PSNR remains above 32 dB with SSIM exceeding 0.96, while the stealth PSNR stays around 40 dB with low opacity leakage. These results demonstrate that HiddenSplat naturally extends to multi-view injection, enabling the hidden contents to be revealed from multiple designated viewpoints without substantially compromising stealth.

C.5 Detailed Sensitivity Analysis

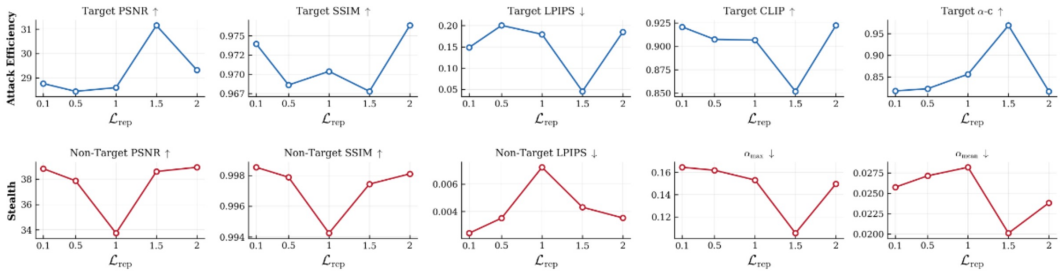
Figure 7 presents a detailed sensitivity analysis of the loss weights in terms of both attack effectiveness and non-target stealth. For the non-target stealth loss $\mathcal{L}_{\text{sth}}$, we observe that $\lambda_s = 1$ achieves the best overall trade-off, yielding the highest target fidelity while maintaining high stealth (high non-target PSNR and low accumulated opacity). Similarly, we find that $\lambda_r = 1.5$ and $\lambda_c = 0.5$ provide the best balance for the repulsion loss $\mathcal{L}_{\text{rep}}$ and the color-spreading loss $\mathcal{L}_{\text{col}}$, respectively. These results justify the loss weights used throughout our experiments.

Table 6: Evaluation on multi-view injection.

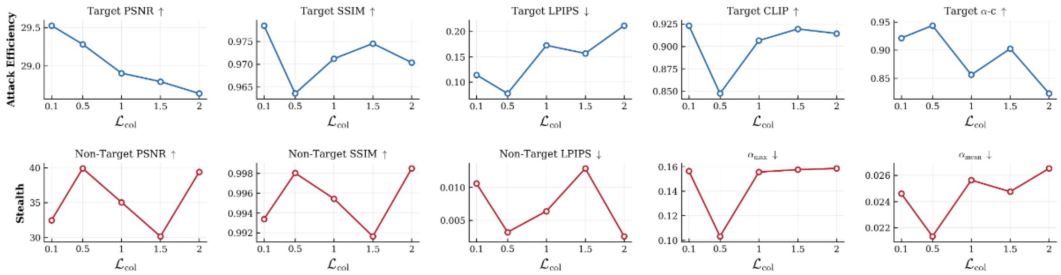
	Target View - Attack Effectiveness				Test Views - Stealthiness				
# of views	PSNR↑	SSIM↑	LPIPS↓	CLIP↑	PSNR↑	SSIM↑	LPIPS↓	α_{max}↓	α_{mean}↓
2	32.65	0.968	0.068	0.924	40.98	0.986	0.016	0.079	0.021
3	33.21	0.986	0.055	0.933	39.25	0.954	0.021	0.068	0.025
4	32.02	0.972	0.052	0.928	41.31	0.986	0.014	0.061	0.019



a) Sensitivity analysis for non-target stealth loss



b) Sensitivity analysis for repulsion loss



c) Sensitivity analysis for color-spreading loss

Figure 7: Detailed sensitivity analysis for different loss weights.

Interestingly, although $\mathcal{L}_{\text{sth}}$, $\mathcal{L}_{\text{rep}}$, and $\mathcal{L}_{\text{col}}$ are introduced primarily to preserve non-target stealth, moderate weights also improve attack effectiveness. We attribute this to the fact that these regularizers encourage a more favorable spatial and color distribution of the adversarial Gaussians, reducing redundant overlap and optimization conflicts, which facilitates more accurate target reconstruction.

D Potential Use Cases of HiddenSplat

The ability to inject view-specific content into 3DGS assets introduces potential security risks for downstream systems that rely on shared or externally sourced 3D representations. For example, an autonomous driving system using a poisoned 3DGS-based map could encounter attacker-controlled visual content from a designated viewpoint, which may in turn affect downstream perception or decision-making. Similarly, robots that use 3DGS scenes as persistent visual memory could receive inconsistent observations across viewpoints. We emphasize that these examples represent potential downstream implications of the proposed attack capability rather than end-to-end attacks on autonomous driving or robotic systems; evaluating such downstream effects is beyond the scope of this work.

Beyond security, the same view-specific injection capability could enable controlled content embedding. For example, view-specific Gaussians could be used to embed ownership signals or watermarks into 3D assets, with a designated viewpoint serving as a private verification mechanism. Similarly, the mechanism could support hidden message embedding, where information is revealed only from selected viewpoints. Note that these applications are likewise intended to illustrate the

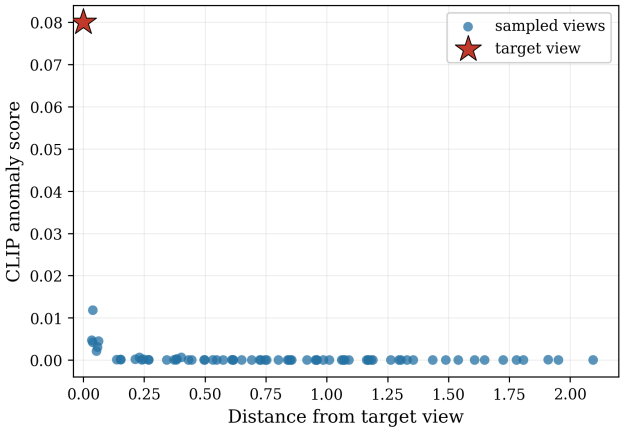


Figure 8: Anomaly-based detection defense.

broader potential of the proposed representation-level capability rather than to claim a complete watermarking or steganographic system. Exploring robustness, capacity, and security of such constructive applications is out of scope of this work.

E Potential Defenses

In this section, we investigate potential defense strategies against HiddenSplat. We assume a realistic setting in which the defender does not know whether a 3DGS asset has been poisoned, nor has access to the target viewpoint, target content, or the attack parameters. The defender's goal is either to remove the injected adversarial Gaussians or to detect the presence of the attack.

E.1 Visual Anomaly Detection

This defense detects anomalous rendering perturbation by sampling multiple viewpoints and analyzing the rendered images. To evaluate this strategy, we render a dense set of viewpoints from the poisoned scene and compute their CLIP similarity with the corresponding renders from the clean scene. As shown in Figure 8, the anomaly score remains close to zero for nearly all non-target viewpoints, increasing only within a very small neighborhood around the target viewpoint. This suggests that a detector based solely on rendered appearance is unlikely to identify the attack unless it samples viewpoints extremely close to the target. Moreover, the viewpoint space of a 3DGS scene is continuous, making it infeasible to exhaustively search for the unknown target viewpoint. Notably, our evaluation assumes access to the clean scene for comparison, which is unavailable in practice, making visual anomaly detection even more challenging. Consequently, a defender is unlikely to render the precise viewpoint at which the hidden content is revealed, further limiting the practicality of this defense.

E.2 Opacity-Based Gaussian Pruning

Since HiddenSplat constrains the opacity of every adversarial Gaussian to be below $o_{\max}$, a natural defense is to prune Gaussians whose opacity falls below a predefined threshold. However, because the defender does not know $o_{\max}$ and cannot distinguish adversarial Gaussians from legitimate low-opacity Gaussians, this strategy inevitably removes clean Gaussians as well, degrading the rendering quality. In practice, the defender can set a pruning threshold that satisfies a predefined acceptable quality degradation. However, Figure 9 shows that the clean scene quality deteriorates rapidly as the pruning threshold increases. If the attacker knows exactly the chosen $o_{\max}$ and that the scene is poisoned, he can sacrifice 4 dB of clean scene PSNR to remove the adversarial Gaussians. Nevertheless, both information is unknown. Thus, choosing a threshold that is too low fails to eliminate the attack, whereas a threshold that is too high severely damages the original scene.

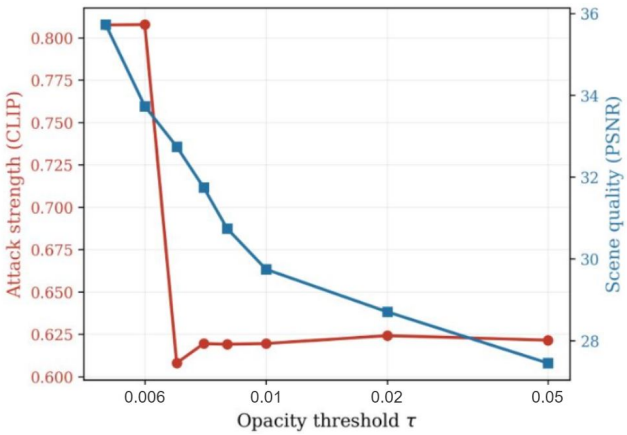


Figure 9: Gaussian-pruning defense – Attack removal vs. Scene stealthiness.

We emphasize that these experiments should be viewed as preliminary defense probes rather than a comprehensive evaluation of defenses. The results suggest that neither strategy can fully mitigate the HiddenSplat attack without introducing additional costs or compromising rendering quality. Developing robust and efficient defenses against HiddenSplat therefore remains an important direction for future work. In this paper, our primary contribution is not a defense mechanism, but the identification of an underexplored content-injection surface that can enable both view-dependent attacks and benign applications, such as watermarking and hidden-message embedding for 3DGS.

F Visualization Results

Figure 10 presents qualitative results on different pretrained 3DGS scenes and target patterns. HiddenSplat consistently reconstructs the target content with high fidelity at the designated target view while leaving the scene visually unchanged from non-target viewpoints.

To illustrate the placement of the adversarial Gaussians, Figure 11 provides an additional visualization. The left column shows renderings of the poisoned 3DGS scene, while the right column shows the bounding box (square, before masking) in which the adversarial Gaussians can be placed, rendered with $100\times$ larger density and red color to make them visible. This shows that despite being explicitly embedded in the 3DGS asset, the adversarial Gaussians remain effectively imperceptible.

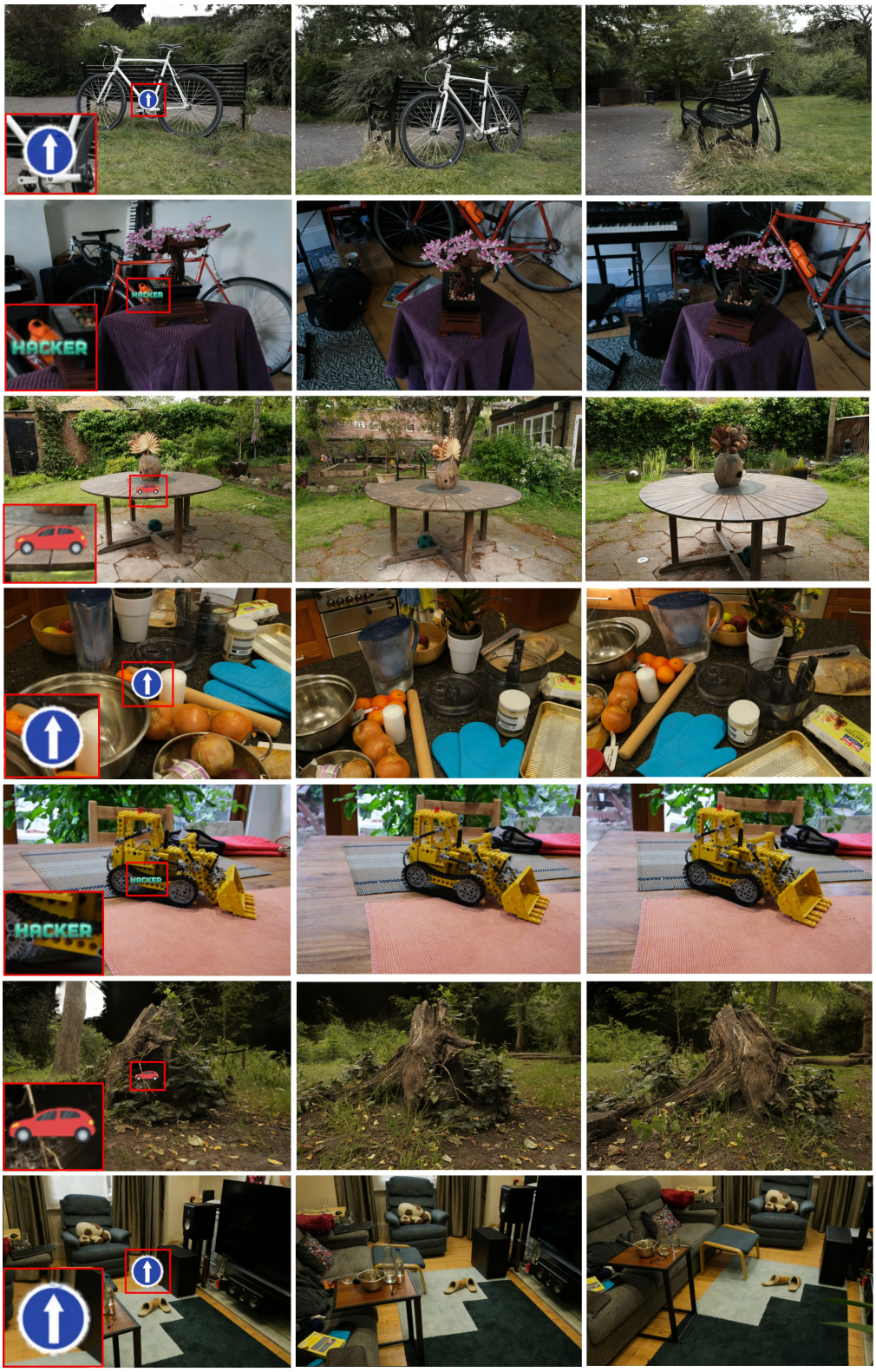


Figure 10: Examples of HiddenSplat rendering.

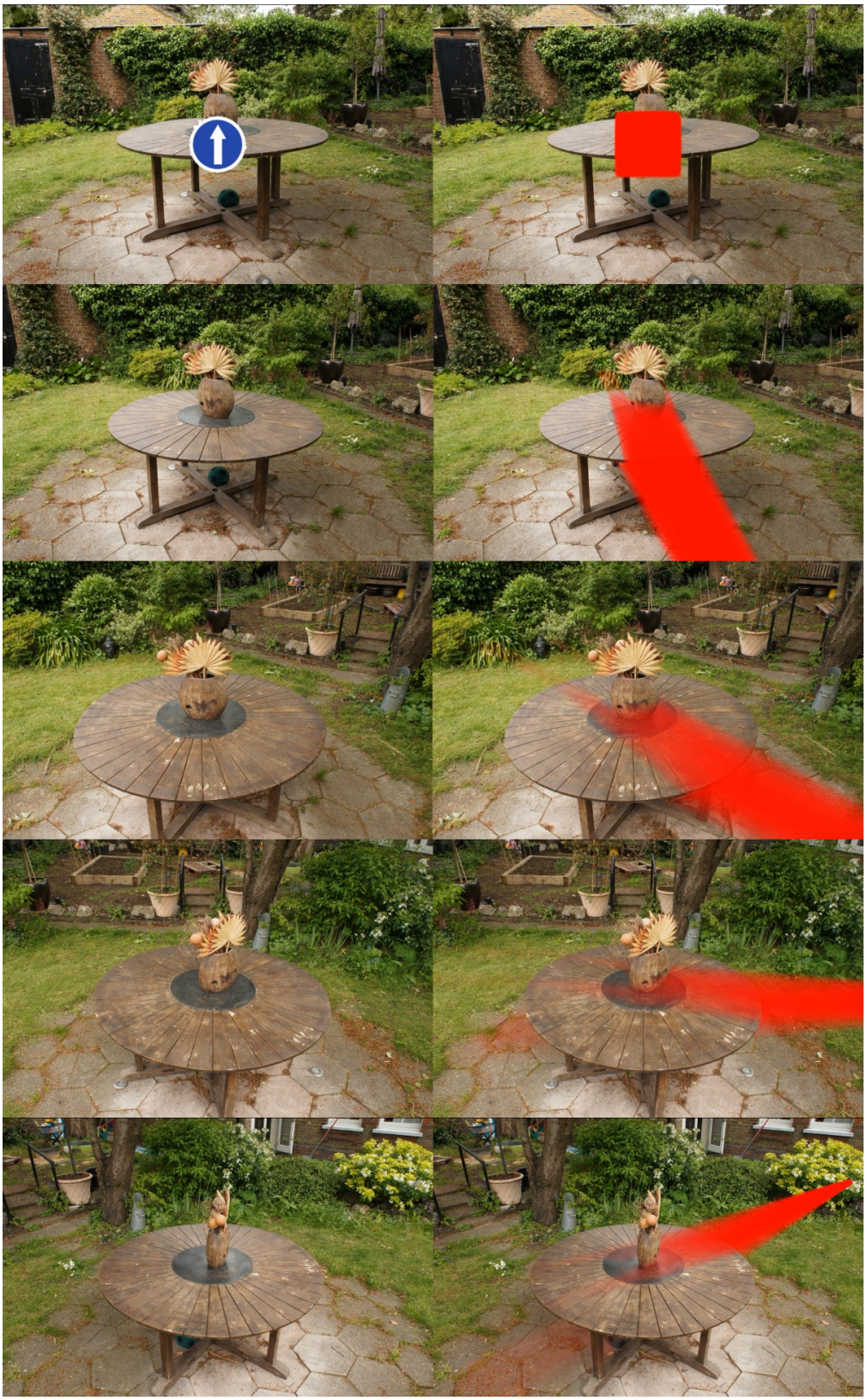


Figure 11: **Left column**: HiddenSplat normal rendering. **Right column**: Adversarial Gaussians (initial bounding box) are highlighted in red with $100\times$ density.