# Splats in Splats: Robust and Effective 3D Steganography Towards Gaussian Splatting

**Yijia Guo [1*], Wenkai Huang [2,3*], Yang Li [1], Gaolei Li [2,3†], Hang Zhang [4]**
**Liwen Hu [1], Jianhua Li [2,3], Tiejun Huang [1], Lei Ma [1,5†]**

[1]National Key Laboratory for Multimedia Information Processing, Peking University
[2]School of Computer Science, Shanghai Jiao Tong University
[3]Shanghai Key Laboratory of Integrated Administration Technologies for Information Security, Shanghai Jiao Tong University
[4]Cornell University
[5]National Biomedical Imaging Center, Peking University
{2301112015, ly0376, liwenhu, tjhuang, leima}@stu.pku.edu.cn, {sjtuhwk, gaolei_li, lijh888}@sjtu.edu.cn, hz459@cornell.edu

## Abstract

3D Gaussian splatting (3DGS) has demonstrated impressive 3D reconstruction performance with explicit scene representations. Given the widespread application of 3DGS in 3D reconstruction and generation tasks, there is an urgent need to protect the copyright of 3DGS assets. However, existing copyright protection techniques for 3DGS overlook the usability of 3D assets, posing challenges for practical deployment. Here we describe ***splats in splats***, the first 3DGS steganography framework that embeds 3D content in 3DGS itself without modifying any attributes. To achieve this, we take a deep insight into spherical harmonics (SH) and devise an importance-graded SH coefficient encryption strategy to embed the hidden SH coefficients. Furthermore, we employ a convolutional autoencoder to establish a mapping between the original Gaussian primitives' opacity and the hidden Gaussian primitives' opacity. Extensive experiments indicate that our method significantly outperforms existing 3D steganography techniques, with **5.31%** higher scene fidelity and **3**× faster rendering speed, while ensuring security, robustness, and user experience.

## Introduction

Building on the success of utilizing a discrete 3D Gaussian representation for scenes, 3D Gaussian Splatting (3DGS) (Kerbl et al. 2023) significantly accelerates the training and rendering speed of radiance fields with explicit scene representations. Given the widespread application of 3DGS in 3D reconstruction and generation tasks (Yi et al. 2023, 2024), the identification and attribution of 3DGS assets are crucial to ensure their secure usage and protect against copyright infringement. Steganography can play a key role in verifying the provenance of 3DGS assets and preventing both accidental and deliberate misuse. In addition to ensuring security, fidelity, and robustness as observed in traditional steganography, the steganography technique for 3DGS must fulfill the following two criteria:

**a) Protecting the 3DGS asset itself rather than the rendered image.** A straightforward approach for 3DGS steganography is embedding specific information directly into the rendered images from the 3DGS (Huang et al. 2024). However, it solely safeguards the copyright of the rendered images from specific viewpoints instead of the copyright of the 3DGS itself. Malicious users may generate new samples employing different rendering strategies after stealing the core model, circumventing the external steganography anticipated by the creators.

**b) Ensuring the usability of 3DGS assets.** Steganography techniques must not disrupt normal usage of 3DGS assets. Digital steganography for 2D images (Baluja 2019; Xu et al. 2022), videos (Mou et al. 2023; Luo et al. 2023) and Neural Radiance Fields (NeRF) (Li et al. 2023; Jang et al. 2024) have all ensured this aspect. However, existing 3DGS steganography techniques such as GS-Hider (Zhang et al. 2024) and SecureGS (Zhang et al. 2025) modify the attributes and rendering pipeline of vanilla 3DGS, as shown in Fig. 1. These methods greatly affect users' standard utilization since the modified 3DGS asset poses challenges for practical deployment in vanilla 3DGS rendering engines and downstream tasks. The fundamental solution is to keep the attributes of vanilla 3DGS. **Is there a solution that can embed hidden information in 3DGS itself without modifying any attributes of the vanilla 3DGS?**

To fulfill the above demands, we propose an effective, flexible and robust steganography framework named ***splats in splats***. To the best of our knowledge, splats in splats is the first framework that embeds 3D content into the vanilla 3DGS while fully preserving its attributes. To achieve this, we take a deep insight into spherical harmonics (SH) and devise an importance-graded SH coefficient encryption/decryption strategy to embed the hidden SH coefficients. Furthermore, we employ a convolutional autoencoder to establish a mapping between the original Gaussian primitives' opacity and the hidden Gaussian primitives' opacity. Splats in splats can also embed other types of digital content, like images. Our main contributions are as follows:

*These authors contributed equally.
†Corresponding authors.

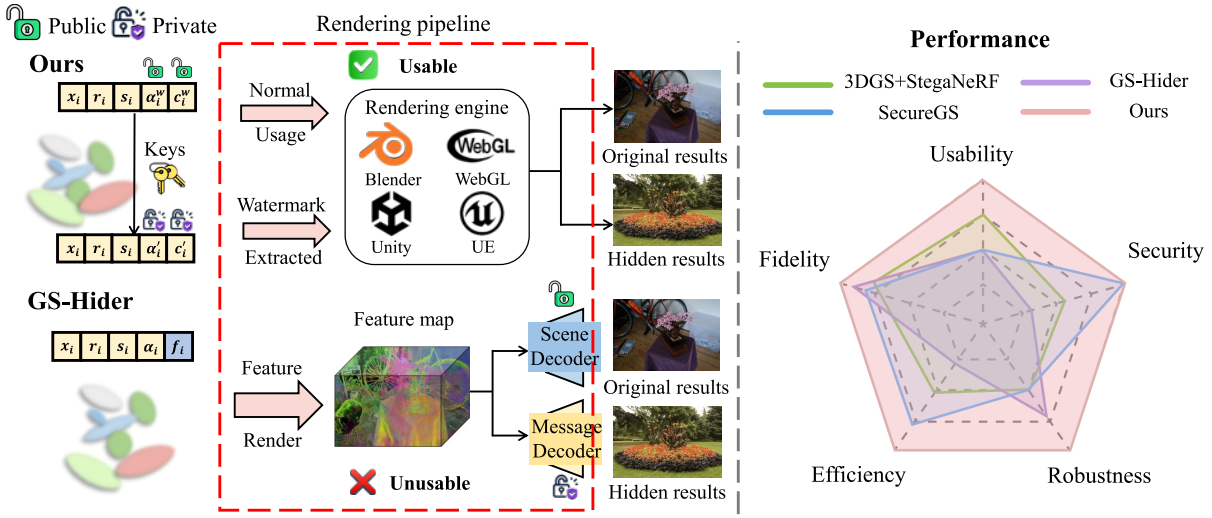


Figure 1: **Left:** GS-Hider and our method's rendering pipeline. GS-Hider (Zhang et al. 2024) employs a coupled feature field and neural decoders to render the original and hidden scenes simultaneously, affecting user's standard utilization. We retain the vanilla 3DGS pipeline to preserve user experience. **Right:** Comparison of different 3DGS steganography methods. Existing works all have shortcomings in terms of robustness, fidelity, efficiency, and usability. Our method can maximize the fidelity of the original scene while ensuring the rendering speed and usability of 3DGS, as well as the security of the steganography.

- We propose splats in splats, the first steganography framework that embeds 3D content in 3DGS itself without modifying any attributes of the vanilla 3DGS.
- Building on a deep insight into the Gaussian primitives' attributes, we devise importance-graded SH coefficient encryption and autoencoder-assisted opacity mapping to preserve the structural integrity of the vanilla 3DGS.
- Extensive experiments indicate that splats in splats achieves state-of-the-art fidelity and efficiency while ensuring security, robustness, and user experience.

## Related Works

### 3D Gaussian Splatting

3D Gaussian splatting (Kerbl et al. 2023) significantly accelerates the training and rendering speed of radiance fields. This breakthrough immediately attracted countless attention. Recently, many researchers have focused on improving the rendering quality (Yu et al. 2024b), effectiveness (Fan et al. 2024; Chen et al. 2024) and the storage (Navaneet et al. 2024) of 3DGS, extending it to dynamic 3D scenes (Wu et al. 2023; Li et al. 2024), large-scale outdoor scenes (Liu et al. 2024), and high-speed scenes (Xiong et al. 2024; Yu et al. 2024a; Guo et al. 2024b), relaxing its restrictions on camera poses (Müller et al. 2022) and sharp images (Chen and Liu 2024). 3DGS is also widely used in the area of inverse rendering (Guo et al. 2024a; Liang et al. 2023; Gao et al. 2023), 3D generation (Yi et al. 2023) and 3D editing (Chen et al. 2023). Patently, 3DGS is becoming a mainstream 3D asset. Thus, the management of the copyright of 3DGS has been emerging as an urgent problem.

### 3D Steganography

3D steganography techniques place greater emphasis on the unique characteristics of 3D representations rather than 2D images (Baluja 2019; Xu et al. 2022), videos (Mou et al. 2023; Luo et al. 2023) or large-scale generative models (Fernandez et al. 2023; Wen et al. 2024). Existing 3D steganography techniques primarily focus on meshes or point clouds, employing domain transformations (Li, Yang, and Jin 2023; Kallel, Grati, and Taktak 2023) or homomorphic encryption (Van Rensburg et al. 2023) for information hiding.

Recently, steganography techniques on implicit 3D representations like NeRF have gained prominence. StegaNeRF (Li et al. 2023) embedding secret images into 3D scenes by fine-tuning NeRF's weights while preserving the original visual quality. WaterRF (Jang et al. 2024) embeded binary messages using discrete wavelet transform, achieving great performance. When considering 3DGS, GS-Hider (Zhang et al. 2024) utilized a scene and a message decoder to disentangle the hidden message from the original scene. However, GS-Hider not only caused a degradation in both rendering speed and visual quality but also damaged the vanilla 3DGS architecture, resulting in limited effectiveness in practical deployment. SecureGS, which is built on Scaffold-GS (Lu et al. 2024), faces the same problems. So there is an urgent need for 3DGS steganography research that ensures copyright protection while maintaining the efficiency and usability of the vanilla 3DGS framework.

## Preliminary

In 3DGS, every Gaussian primitive $G$ is defined by a full 3D covariance matrix $\Sigma$ in world space centered at $x \in \mathbb{R}^3$:

$$G(x) = e^{-1/2(x)^T \Sigma^{-1}(x)}. \quad (1)$$

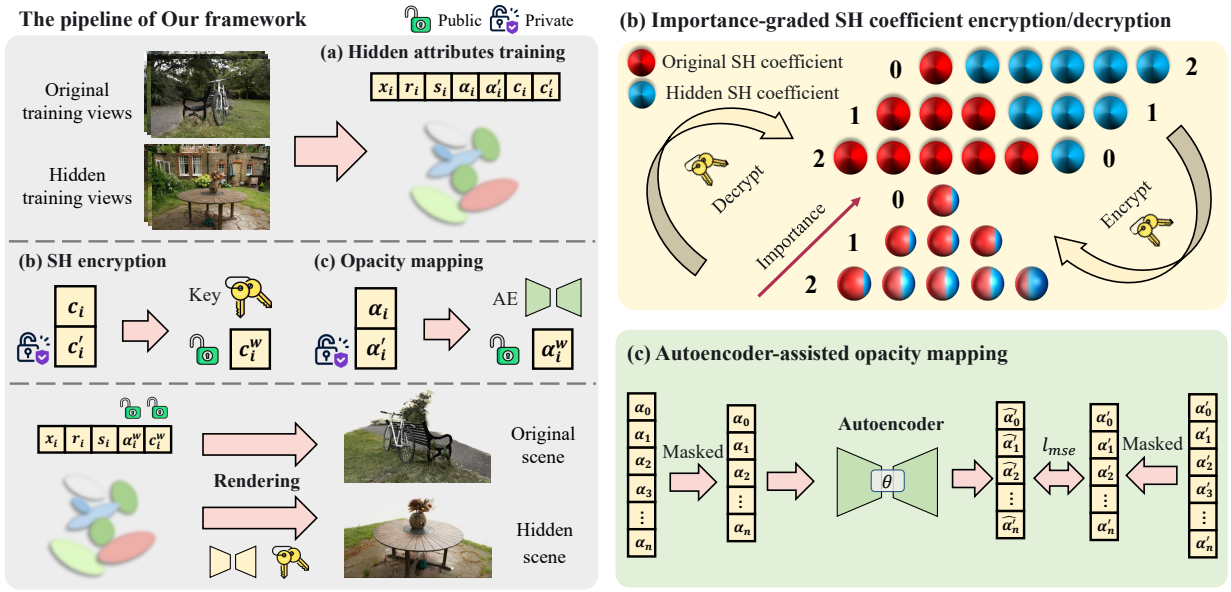


Figure 2: **Overview of the Framework**. To ensure a seamless user experience, we preserves the same attributes as the vanilla 3DGS, while enabling the extraction of embedded 3D content for owners. The information embedding and extraction process involves three key steps: **a) Hidden attributes training**: We utilize the original and hidden views to train two sets of SH coefficients and opacity, while ensuring that both sets share the same Gaussian primitive locations. **b) Importance-graded SH coefficient encryption/decryption**: We prioritize SH coefficients by significance, embedding more important hidden SH coefficients into higher-order components of the original SH via bit-shifting, aiming to achieve superior rendering fidelity and high robustness against noise attack. **c) Autoencoder-assisted opacity mapping**: We apply a threshold to discard trivial hidden opacities, subsequently employing an convolutional autoencoder to model the transformation from original to hidden opacity.

The covariance matrix $\Sigma$ of a 3D Gaussian primitive can be described as a rotation matrix $R$ and a scale matrix $S$ and independently optimize of both them.

$$\Sigma = RSS^T R^T. \tag{2}$$

Further, we utilize the method of splatting to project our 3D Gaussian primitives to 2D camera planes for rendering:

$$\Sigma' = JW\Sigma W^T J^T. \tag{3}$$

Where $J$ is the Jacobian of the affine approximation of the projective transformation and $W$ is the viewing transformation. Following this, the pixel color is obtained by alpha-blending $N$ sequentially layered 2D Gaussian splats:

$$C = \sum_{i \in N} c_i \alpha_i \prod_{j=1}^{i-1} (1 - \alpha_j). \tag{4}$$

Where $c_i$ is the color of each point and $\alpha_i$ is given by evaluating a 2D Gaussian with covariance $\Sigma$ multiplied with a learned per-point opacity.

In summary, each 3D Gaussian primitive is defined by five attributes: $\{x_i, r_i, s_i, \alpha_i, c_i\}$. Specifically, $x_i$ and $s_i$ are represented as $1 \times 3$ vectors, while $r_i$ is formatted as a $1 \times 4$ vector, and $\alpha_i$ is a scalar value. Notably, $c_i$ is constituted as a $n \times 3$ matrix of Spherical Harmonic (SH) coefficients, which effectively compensates for view-dependent effects. cspace

# Methodology

## Overview

The overall workflow of our method is depicted in Fig. 2. Our objective is to seamlessly embed a 3D content into 3DGS assets while ensuring that the typical usage pipeline for regular users remains unaffected. Accordingly, the asset owner can leverage a private key to recover the hidden attributes and extract the 3D content, enabling robust verification. To achieve this, we first pre-train hidden SH coefficients and opacity for each 3D Gaussian primitive to align with the hidden scene. Subsequently, we devise importance-graded SH coefficient encryption/decryption and autoencoder-assisted opacity mapping strategies to effectively accomplish 3D information embedding and extraction. With these strategies, we can not only protect the copyright of 3DGS assets but also maintain the essential attributes of vanilla 3DGS, thereby ensuring its usability.

## An Insight in Spherical Harmonics

Any function $F(s)$ is defined on the sphere $S$ can be represented as a set of SH basis functions:

$$F(s) \approx \sum_{l=0}^{q-1} \sum_{m=-l}^{l} f_l^m Y_l^m(s), \tag{5}$$

and the basis functions are defined as:

$$Y_l^m(\theta, \phi) = K_l^m e^{im\phi} P_l^{|m|}, l \in N, m \in (-l, l), \tag{6}$$


Figure 3: Importance of Spherical Harmonics order. We retain the specified order of SH and render the image while setting other orders to zero. Higher-order SH contains only a small amount of High-frequency information.

where $P_l^{|m|}$ are the associated Legendre polynomials and $K_l^m$ are the normalization constants:

$$K_l^m = \sqrt{\frac{(2l+1)(l-|m|)!}{4\pi(l+|m|)!}}. \tag{7}$$

Low values of $l$ (called the band index) represent low-frequency basis functions over the sphere while high values of $l$ represent high-frequency basis functions. In most cases, higher-frequency reflections occupy only a small proportion of the scene. Consequently, the contribution of higher-order spherical harmonic coefficient is minimal, leading to information redundancy in these higher-order terms, as illustrated in Fig. 3. Embedding information within these terms would pose considerable challenges for detection and would ultimately preserve fidelity. In this paper, we capitalize on this property of spherical harmonics to embed information while simultaneously improving resilience against noise attacks that target spherical harmonics.

## 3D Information Embedding

As illustrated in Fig. 2, we embed the trained hidden SH coefficients and opacities into the original Gaussian primitives. For the SH coefficient encryption, we prioritize SH coefficients by order, embedding the more significant hidden coefficients into the higher-order components of the original SH through bit shifting. For the opacity mapping, we set a threshold to filter out insignificant hidden opacities and utilize an convolutional autoencoder to learn the mapping from the original opacity to the hidden opacity.

**Importance-graded SH Coefficient Encryption** For the original SH coefficients $c_i \in \mathbb{R}^{n\times 3}$ and the hidden SH coefficients $c_i' \in \mathbb{R}^{n\times 3}$, let $c_{i,j}$ denote the $j$-th $(0 \le j \le n-1)$ component of $c_i$. Then, the order of $c_{i,j}$ is $\lfloor\sqrt{j}\rfloor$ according to the definition, where $\lfloor\cdot\rfloor$ denotes the floor function. The lower the order of $c_{i,j}$, the more significant the coefficient is in the Gaussian primitive. Drawing from the insight from Sec. , we embed more bits of the low-order $c'_{i,j}$ into the higher-order $c_{i,j}$ to achieve higher quality. For computational convenience, we scale these coefficients and represent them in integer form as binary values of $\{0,1\}^\gamma$.

Firstly, we nullify the lower bits of $c_{i,j}$ **based on its graded importance:**

$$\tilde{c}_{i,j} = c_{i,j} \ \ \& \sim ((1 << (k + \lfloor\sqrt{j}\rfloor)) - 1). \tag{8}$$

Where $<<$ denotes left bit shifting operation, $\sim$ operator represents the bitwise NOT operation, and $k$ represents the bit shifting length corresponding to the 0-order coefficient. Subsequently, we shift $c'_{i,j}$ to the corresponding position and **perform an exclusive OR operation with $\tilde{c}_{i,j}$:**

$$c^w_{i,j} = \tilde{c}_{i,j} \oplus (c'_{i,n-1-j} >> (\gamma - (k + \lfloor\sqrt{j}\rfloor))). \tag{9}$$

Where $>>$ denotes right bit shifting operation, $\gamma$ represents the maximum bit length, and $c^w_{i,j}$ denotes the fused SH coefficient. The $n-1-j$ term indicates that $c'_{i,j}$ has been reversed to match our importance-graded selection. Building on this strategy, we maintain the inherent attributes of original 3DGS, while achieving superior rendering fidelity.

**Autoencoder-assisted Opacity Mapping** To balance the quality of both the original and hidden scenes, we also incorporate hidden opacity attributes and employ a convolutional autoencoder to learn the mapping from original opacity to hidden opacity. Let the original opacity be denoted as $\alpha \in \mathbb{R}^{N\times 1}$ and the hidden opacity attribute as $\alpha' \in \mathbb{R}^{N\times 1}$. Here $N$ is the num of Gaussian primitives after training. We first set a threshold $\tau$ to obtain the indices of the more significant hidden opacity values:

$$\mathcal{I} = \{i \mid \alpha'_i > \tau,\ i \in \{1, 2, \ldots, N\}\}. \tag{10}$$

Notably, we store the coordinates $x_\mathcal{I}$ of the Gaussian primitives at the indices $\mathcal{I}$, which are then used in the hidden opacity estimation process. Then we denote $\alpha_\mathcal{I}$ and $\alpha'_\mathcal{I}$ as the values of $\alpha$ and $\alpha'$ at $\mathcal{I}$, respectively. Based on the observation that $\alpha_i$ and $\alpha'_i$ exhibit complementary relationships at many positions, we apply an autoencoder to $1-\alpha_\mathcal{I}$, performing encoding and decoding to obtain $\hat{\alpha}'_\mathcal{I}$. We then utilize the Mean Squared Error (MSE) to train the autoencoder, learning the mapping from $\alpha_\mathcal{I}$ to $\alpha'_\mathcal{I}$:

$$W_p^* = \arg\min_{\mathcal{E},\mathcal{D}} \ell_{mse}(\mathcal{D}(\mathcal{E}(1-\alpha_\mathcal{I})), \alpha'_\mathcal{I}). \tag{11}$$

Here, $\mathcal{E}$ and $\mathcal{D}$ represent the encoder and decoder, respectively. To ensure real-time rendering, our autoencoder is composed of several simple convolutional and deconvolutional layers. The trained model parameters $W_p^*$ then serve as the private attribute for the owner. Through this strategy, we reduce the storage requirements for the asset owner, while preserving the usability of the modified 3DGS.

## 3D Information Extraction

In cases of suspected copyright infringement, we can use SH coefficient decryption and opacity estimation to recover the hidden $c'_{i,j}$ and $\alpha'_i$:

$$c'_{i,j} = c^w_{i,n-1-j} \ \ \& \ \ (1 << (k + \lfloor\sqrt{n-1-j}\rfloor)), \tag{12}$$

$$\alpha'_\mathcal{I} = \mathcal{D}_p(\mathcal{E}_p(1-\alpha_\mathcal{I})). \tag{13}$$

Where $c^w_{i,j}$ is the public SH coefficient, and $\mathcal{E}_p$ and $\mathcal{D}_p$ represent the encoder and decoder obtained from $W_p^*$, respectively. In practical scenarios, we can choose to retain only the recovered low-order SH coefficients to defend against potential noise attacks. For indices outside of the set $\mathcal{I}$ (i.e., positions from 1 to $N$ not included in $\mathcal{I}$), we set

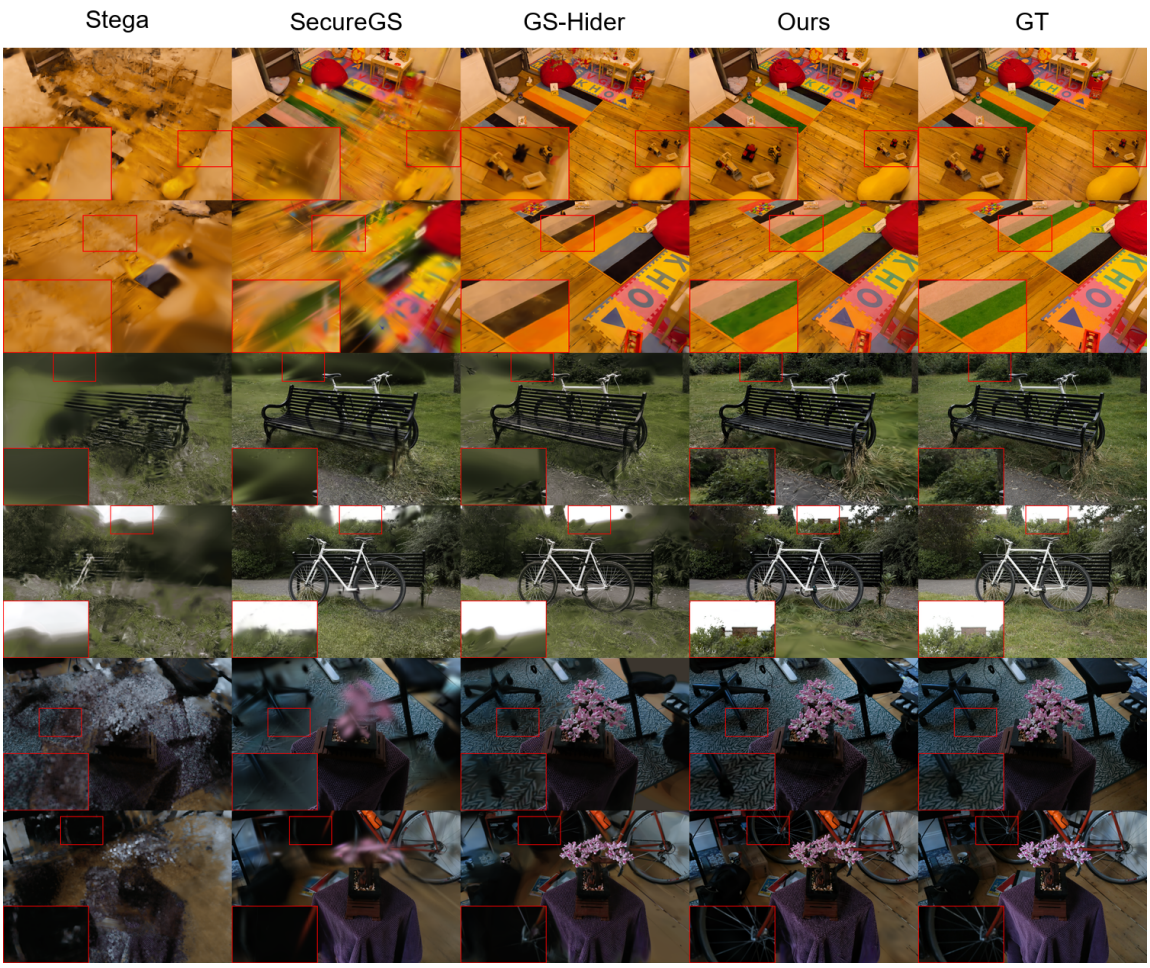


Figure 4: Qualitative comparisons on Mipnerf360 datasets. Our method achieves sharper results with more consistent structures.

the hidden opacity to zero, as these locations are considered less significant. Finally, we use the recovered attributes $\{x_i, r_i, s_i, \alpha_i', c_i'\}$ as the hidden 3DGS, which contains the embedded 3D information.

## Experiment

### Implementation Details

Our code is based on 3DGS and we train models for 30000 iterations on NVIDIA A800 GPU with the same optimizer and hyper-parameters as 3DGS. Unless otherwise specified, $\tau$ and $k$ in Eq. 10 and Eq. 8 are set to 0.25 and 17.

### Experimental Settings

**Datasets:** Same as GS-Hider (Zhang et al. 2024), we conduct experiments on 9 original scenes from the Mip-NeRF360 (Barron et al. 2022), and the correspondence between the hidden and original scenes also follows GS-Hider.
**Baselines:** we compare our method with existing 3DGS steganography method GS-Hider (Zhang et al. 2024) and SecureGS (Zhang et al. 2025). Meanwhile, we feed the rendering results of the original 3DGS to a U-shaped decoder and constrain it to output hidden scenes according to (Li et al. 2023), called 3DGS+StegaNeRF.
**Metrics:** We evaluate the synthesized novel view in terms of Peak Singal-to-Noise Ratio (PSNR) and Structural Similarity Index Measure (SSIM). Meanwhile, we use rendering FPS to evaluate our efficiency.

### Evaluation

Firstly, we compare splats in splats with baselines in terms of fidelity, efficiency, robustness, security, and usability.
**a) Fidelity:** We use Novel View Synthesis (NVS) results to evaluate scene fidelity. As shown in Tab. 1, our method achieves the best NVS quality on both original and hidden scenes. Fig. 4 shows more visual details which indicate that our splats in splats achieves the most appealing visualization results without the artifacts seen in SecureGS or the erroneous textures and colors observed in GS-Hider.
**b) Efficiency:** Tab. 2 demonstrates that we achieve SOTA rendering and training efficiency. Our method achieves 100 rendering FPS, which is 3x faster than the existing 3DGS steganography methods using neural networks, such as GS-Hider. Our training time is only half of the existing methods.

| Method | Type | Bicycle | Flowers | Garden | Stump | Treehill | Room | Counter | Kitchen | Bonsai | Average |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 3DGS+ StegaNeRF | Scene | 22.789 | 19.811 | 23.140 | 24.260 | 21.465 | 28.659 | 26.400 | 25.039 | 25.521 | 24.120 |
| | Message | 15.990 | 14.753 | 16.615 | 16.513 | 17.543 | 17.213 | 16.706 | 17.534 | 17.259 | 16.681 |
| GS-Hider | Scene | 24.018 | 20.109 | 26.753 | 24.573 | 21.503 | 28.865 | 27.445 | 29.447 | 29.643 | 25.817 |
| | Message | 28.219 | 26.389 | **32.348** | 25.161 | 20.276 | 22.885 | 20.792 | 26.690 | 23.846 | 25.179 |
| SecureGS | Scene | 24.237 | 20.566 | 26.624 | **25.523** | **22.218** | **31.223** | 28.499 | 29.779 | 30.499 | 26.574 |
| | Message | 21.798 | 23.838 | 26.883 | 24.615 | **23.778** | 21.576 | 20.851 | 23.240 | 21.367 | 23.679 |
| Ours | Scene | **24.431** | **20.685** | **26.831** | 25.414 | 22.156 | 30.930 | **28.642** | **30.699** | **30.953** | **26.749** |
| | Message | **28.988** | **29.006** | 29.013 | **28.811** | 22.794 | **23.355** | **22.514** | **28.345** | **25.826** | **26.517** |

Table 1: Comparison of the quantitative performance of the original and hidden message scenes. We also report the average rendering speed and the adaptability of SIBR viewer (official rendering engine of 3DGS). **Best** results are marked in **bold** and the second best results are marked with underline.

| Method | Usability | | | Efficiency | |
|---|---|---|---|---|---|
| | Pipe. | Attr. | Viewer | Train time | FPS |
| GS+StegaNeRF | ✗ | ✓ | ✗ | 97 Mins | 22 |
| SecureGS | ✗ | ✗ | ✗ | 71 Mins | 36 |
| GS-Hider | ✗ | ✗ | ✗ | 119 Mins | 44 |
| Ours | ✓ | ✓ | ✓ | 47 Mins | 118 |

Table 2: Comparison of the usability and efficiency. Pipe. and Atrr. represent maintaining vanilla 3DGS's rendering pipeline and attributes.

**c) Usability:** Tab. 2 also demonstrates the adaptability of splats in splats and baselines to the rendering engine (SIBR Viewer) provided by 3DGS. Our method can be directly integrated into the rendering pipeline provided by 3DGS, offering a seamless user experience that existing approaches cannot achieve. We also demonstrate whether all methods maintain the vanilla 3DGS's rendering pipeline (Pipe. in short) and attributes (Attr. in short). Tab. 2 also indicates that our method is the first framework that embeds 3D content in 3DGS itself without modifying any attributes and pipelines.

| Method | SecureGS | | GS-Hider | | Ours | |
|---|---|---|---|---|---|---|
| Ratio | PSNR↑ | SSIM↑ | PSNR↑ | SSIM↑ | PSNR↑ | SSIM↑ |
| 5% | 22.131 | 0.711 | 25.179 | 0.780 | **26.519** | **0.797** |
| 10% | 20.418 | 0.678 | 25.179 | 0.780 | **26.518** | **0.797** |
| 15% | 19.158 | 0.645 | 25.179 | 0.780 | **26.518** | **0.797** |
| 25% | 16.960 | 0.577 | 25.167 | 0.780 | **26.517** | **0.797** |

Table 3: Quantitative comparisons of sequential pruning. Our method achieves better robustness with lower metrics decline under sequential pruning.

**d) Robustness:** To evaluate the robustness of our method, we degrade the Gaussian primitives using sequential pruning and random pruning following (Zhang et al. 2024). For sequential pruning, where primitives with lower opacity are deleted, our method exhibits exceptional stability: under 25% sequential pruning, our PSNR decreases by merely 0.002. This significantly outperforms both GS-Hider (0.012 drop) and SecureGS (substantial 5.171 degradation). Similarly, for random pruning (25% removal), our approach maintains strong robustness with only 0.09 PSNR reduction, whereas GS-Hider suffers a 1.260 decline and SecureGS shows even greater vulnerability with 1.435 PSNR loss. The comparative results in Tab. 3 and 4 demonstrate that our framework achieves superior robustness against pruning attacks compared to existing SOTA methods.

| Method | SecureGS | | GS-Hider | | Ours | |
|---|---|---|---|---|---|---|
| Ratio | PSNR↑ | SSIM↑ | PSNR↑ | SSIM↑ | PSNR↑ | SSIM↑ |
| 5% | 22.920 | 0.727 | 24.923 | 0.774 | **26.415** | **0.794** |
| 10% | 22.596 | 0.721 | 24.673 | 0.767 | **26.375** | **0.793** |
| 15% | 22.280 | 0.713 | 24.371 | 0.760 | **26.346** | **0.793** |
| 25% | 21.485 | 0.695 | 23.661 | 0.741 | **26.320** | **0.792** |

Table 4: Quantitative comparisons of random pruning. Our method achieves better robustness compared to SecureGS and GS-Hider under random pruning.

**e) Security:** We analyze the security of splats in splats from the following aspects. From the view of point cloud file, it preserves the same structure as the original 3DGS. From the view of output, our scenes do not exhibit any artifacts or edges of hidden scenes and maintain high fidelity (shown in Fig. 4 and Tab. 1). Finally, we visualize the rendering results which can also be directly obtained by users. Fig. 5 indicates that the rendering results of GS-Hider and 3DGS+Steganerf have obvious hidden information leakage while our method ensure the security.

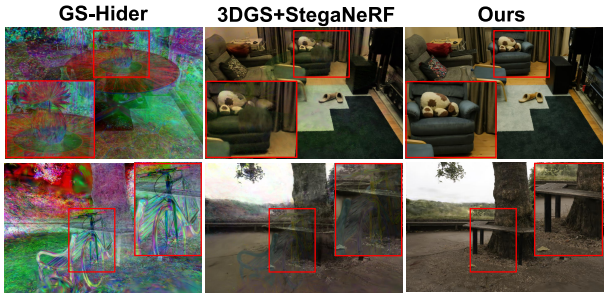


Figure 5: Comparison of rendering results. GS-hider and 3DGS+SteganeNeRF have obvious information leakage. **Better viewed on screen with zoom in**.

## Ablation Study

**a) SH/Opacity encryption:** We first conduct ablations to demonstrate the necessity of opacity mapping and SH encryption. Quantitative results in Tab. 5 demonstrate that opacity mapping offers a significant advantage in novel view synthesis for both original and hidden scenes, while SH encryption plays a crucial role in maintaining hidden scene fidelity. The aforementioned results are consistent with a fundamental principle that 3D assets are composed of two essential components **geometric structure** and **visual appearance.** Our method uses opacity mapping to hide geometric information and SH encryption to hide appearance.

| Method | Scene | | | Message | | |
|---|---|---|---|---|---|---|
| | $PSNR\uparrow$ | $SSIM\uparrow$ | $LPIPS\downarrow$ | $PSNR\uparrow$ | $SSIM\uparrow$ | $LPIPS\downarrow$ |
| w/o op | 24.209 | 0.739 | 0.301 | 23.346 | 0.767 | 0.330 |
| w/o SH | **26.795** | **0.793** | **0.241** | 11.092 | 0.432 | 0.331 |
| SH+op (Ours) | 26.749 | 0.784 | 0.262 | **26.517** | **0.797** | **0.308** |

Table 5: Quantitative results of ablations on SH/Opacity encryption.

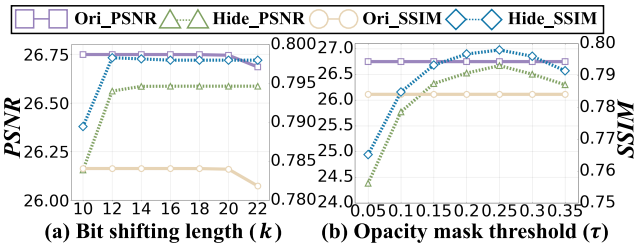


Figure 6: Effect of different shifting lengths $k$ for importance-graded bit encryption/decryption (sub-figure a) and thresholds $\tau$ for autoencoder-assisted opacity mapping (sub-figure b).

**b) Opacity mask threshold:** We conduct ablations to investigate the effect of different opacity mask thresholds $\tau$. We vary $\tau$ in a wide range from 0.0 to 0.3 and plot the average PSNR and SSIM in Fig. 6. Results demonstrate that the opacity mask threshold has a significant influence on rendering quality and splats in splats achieve the best performance when opacity $\tau$ is set to 0.25. The visualization results in Fig. 7 further indicate that our opacity mask significantly reduces artifacts and geometric errors, leading to visually appealing rendering outcomes.

**c) Bit shifting length:** We vary the bit shifting length $k$ from 10 to 22 and plot the average PSNR and SSIM. The results in Fig. 6 demonstrate that $k$ has minimal impact on rendering results of both original and hidden scene, which further demonstrates the robustness of our method.

**d) Importance-graded SH coefficient encryption:** We further conduct ablations to illustrate why importance-graded SH coefficient encryption is necessary. We compare the performance of our method with the average encryption (AVG) under different levels of Gaussian noise, with noise levels ranging from 0.0005 to 0.01. Quantitative results in Tab. 6 demonstrate that importance-graded encryption offers a significant advantage in noise resistance, which enhances both

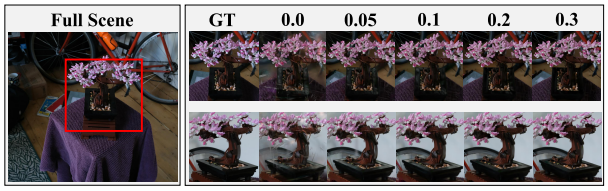


Figure 7: Visualization results of different opacity mask threshold. **Better viewed on screen with zoom in.**

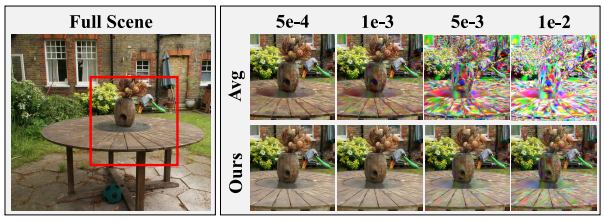


Figure 8: Visualization results under different noise levels. **Better viewed on screen with zoom in.**

security and robustness. Fig. 8 confirms that as the noise level increases, AVG encryption causes a dramatic degradation in the quality of the recovered hidden scene. In contrast, splats in splats exhibits strong robustness against high-level noise, ensuring successful extraction of the hidden 3D information.

| Method | Gaussian noise level | | | | Average |
|---|---|---|---|---|---|
| | 0.0005 | 0.001 | 0.005 | 0.01 | |
| AVG | 24.167 | 21.991 | 11.442 | 7.471 | 16.267 |
| Ours | **24.577** | **24.509** | **22.797** | **20.032** | **22.979** |

Table 6: Quantitative results of PSNR for rendered hidden images under different noise levels.

## Conclusion

We propose splats in splats, an effective and flexible steganography framework for 3D Gaussian splatting. To the best of our knowledge, splats in splats is the first 3DGS steganography method that ensures security, fidelity, robustness, and rendering efficiency while maintaining usability and scalability. By carefully designed importance-graded SH coefficient encryption and autoencoder-assisted opacity mapping, splats in splats injects 3D information into the original 3D scenes presented by 3DGS while fully preserving the attributes of the vanilla 3DGS. Extensive experiments indicate that splats in splats achieves state-of-the-art rendering efficiency and fidelity while being well-suited for deployment across diverse applications. This paper offers a promising outlook on provenance verification in 3DGS and calls for more effort on user experience and scalability. **Limitation:** Splats in splats has some impact on view-dependent details in both original and hidden scenes. We will continue to improve our method to make it applicable to more scenes.

## Acknowledgments

This work was supported by National Science and Technology Major Project (Grant No. 2022ZD0116305), National Nature Science Foundation of China under Grant No. 62088102, 62572314, 62202303, and 62471301, and the Beijing Natural Science Foundation (Grant Nos. F251020 and JQ24023).

## References

Baluja, S. 2019. Hiding images within images. *IEEE transactions on pattern analysis and machine intelligence*, 42(7): 1685–1697.

Barron, J. T.; Mildenhall, B.; Verbin, D.; Srinivasan, P. P.; and Hedman, P. 2022. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 5470–5479.

Chen, W.; and Liu, L. 2024. Deblur-GS: 3D Gaussian Splatting from Camera Motion Blurred Images. *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, 7(1): 1–15.

Chen, Y.; Chen, Z.; Zhang, C.; Wang, F.; Yang, X.; Wang, Y.; Cai, Z.; Yang, L.; Liu, H.; and Lin, G. 2023. Gaussianeditor: Swift and controllable 3d editing with gaussian splatting. *arXiv preprint arXiv:2311.14521*.

Chen, Y.; Xu, H.; Zheng, C.; Zhuang, B.; Pollefeys, M.; Geiger, A.; Cham, T.-J.; and Cai, J. 2024. Mvsplat: Efficient 3d gaussian splatting from sparse multi-view images. *arXiv preprint arXiv:2403.14627*.

Fan, Z.; Cong, W.; Wen, K.; Wang, K.; Zhang, J.; Ding, X.; Xu, D.; Ivanovic, B.; Pavone, M.; Pavlakos, G.; et al. 2024. Instantsplat: Unbounded sparse-view pose-free gaussian splatting in 40 seconds. *arXiv preprint arXiv:2403.20309*.

Fernandez, P.; Couairon, G.; Jégou, H.; Douze, M.; and Furon, T. 2023. The stable signature: Rooting watermarks in latent diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 22466–22477.

Gao, J.; Gu, C.; Lin, Y.; Zhu, H.; Cao, X.; Zhang, L.; and Yao, Y. 2023. Relightable 3d gaussian: Real-time point cloud relighting with brdf decomposition and ray tracing. *arXiv preprint arXiv:2311.16043*.

Guo, Y.; Bai, Y.; Hu, L.; Guo, Z.; Liu, M.; Cai, Y.; Huang, T.; and Ma, L. 2024a. PRTGS: Precomputed Radiance Transfer of Gaussian Splats for Real-Time High-Quality Relighting. *arXiv preprint arXiv:2408.03538*.

Guo, Y.; Hu, L.; Ma, L.; and Huang, T. 2024b. SpikeGS: Reconstruct 3D scene via fast-moving bio-inspired sensors. *arXiv preprint arXiv:2407.03771*.

Huang, X.; Li, R.; Cheung, Y.-m.; Cheung, K. C.; See, S.; and Wan, R. 2024. Gaussianmarker: Uncertainty-aware copyright protection of 3d gaussian splatting. *Advances in Neural Information Processing Systems*, 37: 33037–33060.

Jang, Y.; Lee, D. I.; Jang, M.; Kim, J. W.; Yang, F.; and Kim, S. 2024. WateRF: Robust Watermarks in Radiance Fields for Protection of Copyrights. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 12087–12097.

Kallel, I. F.; Grati, A.; and Taktak, A. 2023. 3D Data Security: Robust 3D Mesh Watermarking Approach for Copyright Protection. In *Examining Multimedia Forensics and Content Integrity*, 1–37. IGI Global.

Kerbl, B.; Kopanas, G.; Leimkühler, T.; and Drettakis, G. 2023. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42(4): 1–14.

Li, C.; Feng, B. Y.; Fan, Z.; Pan, P.; and Wang, Z. 2023. Steganerf: Embedding invisible information within neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 441–453.

Li, D.; Huang, S.-S.; Lu, Z.; Duan, X.; and Huang, H. 2024. ST-4DGS: Spatial-Temporally Consistent 4D Gaussian Splatting for Efficient Dynamic Scene Rendering. In *ACM SIGGRAPH 2024 Conference Papers*, 1–11.

Li, D.; Yang, Z.; and Jin, X. 2023. Zero watermarking scheme for 3D triangle mesh model based on global and local geometric features. *Multimedia Tools and Applications*, 82(28): 43635–43648.

Liang, Z.; Zhang, Q.; Feng, Y.; Shan, Y.; and Jia, K. 2023. Gs-ir: 3d gaussian splatting for inverse rendering. *arXiv preprint arXiv:2311.16473*.

Liu, Y.; Guan, H.; Luo, C.; Fan, L.; Peng, J.; and Zhang, Z. 2024. Citygaussian: Real-time high-quality large-scale scene rendering with gaussians. *arXiv preprint arXiv:2404.01133*.

Lu, T.; Yu, M.; Xu, L.; Xiangli, Y.; Wang, L.; Lin, D.; and Dai, B. 2024. Scaffold-gs: Structured 3d gaussians for view-adaptive rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 20654–20664.

Luo, X.; Li, Y.; Chang, H.; Liu, C.; Milanfar, P.; and Yang, F. 2023. DVMark: a deep multiscale framework for video watermarking. *IEEE Transactions on Image Processing*.

Mou, C.; Xu, Y.; Song, J.; Zhao, C.; Ghanem, B.; and Zhang, J. 2023. Large-capacity and flexible video steganography via invertible neural network. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 22606–22615.

Müller, T.; Evans, A.; Schied, C.; and Keller, A. 2022. Instant neural graphics primitives with a multiresolution hash encoding. *ACM transactions on graphics (TOG)*, 41(4): 1–15.

Navaneet, K.; Pourahmadi Meibodi, K.; Abbasi Koohpayegani, S.; and Pirsiavash, H. 2024. Compgs: Smaller and faster gaussian splatting with vector quantization. In *European Conference on Computer Vision*, 330–349. Springer.

Van Rensburg, B. J.; Puteaux, P.; Puech, W.; and Pedeboy, J.-P. 2023. 3D object watermarking from data hiding in the homomorphic encrypted domain. *ACM Transactions on Multimedia Computing, Communications and Applications*, 19(5s): 1–20.

Wen, Y.; Kirchenbauer, J.; Geiping, J.; and Goldstein, T. 2024. Tree-rings watermarks: Invisible fingerprints for diffusion images. *Advances in Neural Information Processing Systems*, 36.

Wu, G.; Yi, T.; Fang, J.; Xie, L.; Zhang, X.; Wei, W.; Liu, W.; Tian, Q.; and Wang, X. 2023. 4d gaussian splatting for real-time dynamic scene rendering. *arXiv preprint arXiv:2310.08528*.

Xiong, T.; Wu, J.; He, B.; Fermuller, C.; Aloimonos, Y.; Huang, H.; and Metzler, C. A. 2024. Event3DGS: Event-based 3D Gaussian Splatting for Fast Egomotion. *arXiv preprint arXiv:2406.02972*.

Xu, Y.; Mou, C.; Hu, Y.; Xie, J.; and Zhang, J. 2022. Robust invertible image steganography. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 7875–7884.

Yi, T.; Fang, J.; Wu, G.; Xie, L.; Zhang, X.; Liu, W.; Tian, Q.; and Wang, X. 2023. Gaussiandreamer: Fast generation from text to 3d gaussian splatting with point cloud priors. *arXiv preprint arXiv:2310.08529*.

Yi, T.; Fang, J.; Zhou, Z.; Wang, J.; Wu, G.; Xie, L.; Zhang, X.; Liu, W.; Wang, X.; and Tian, Q. 2024. GaussianDreamerPro: Text to Manipulable 3D Gaussians with Highly Enhanced Quality. *arXiv preprint arXiv:2406.18462*.

Yu, W.; Feng, C.; Tang, J.; Jia, X.; Yuan, L.; and Tian, Y. 2024a. EvaGaussians: Event Stream Assisted Gaussian Splatting from Blurry Images. *arXiv preprint arXiv:2405.20224*.

Yu, Z.; Chen, A.; Huang, B.; Sattler, T.; and Geiger, A. 2024b. Mip-splatting: Alias-free 3d gaussian splatting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 19447–19456.

Zhang, X.; Meng, J.; Li, R.; Xu, Z.; Zhang, Y.; and Zhang, J. 2024. GS-Hider: Hiding Messages into 3D Gaussian Splatting. *arXiv preprint arXiv:2405.15118*.

Zhang, X.; Meng, J.; Xu, Z.; Yang, S.; Wu, Y.; Wang, R.; and Zhang, J. 2025. SecureGS: Boosting the Security and Fidelity of 3D Gaussian Splatting Steganography. *arXiv preprint arXiv:2503.06118*.