Gaussian Image Steganography via Parameter-Domain Keyed Embeddings

Tong Wu*
East China University of Science and Technology
Shanghai, China
24012920@mail.ecust.edu.cn

Runze Cheng†
University of Cambridge
Cambridge, United Kingdom
rc2062@cam.ac.uk

Xiaoyue Fan
University College London
London, United Kingdom
merryxyfan@gmail.com

Kaan Akşit
University College London
London, United Kingdom
kaanaksit@kaanaksit.com

Abstract

2D Gaussian-based image representation is becoming increasingly popular, and our work proposes a new approach to steganography by embedding information within Gaussian parameters rather than image pixels. We first fit the parameters of this representation to the target image and employ a secret key to select a subset of these parameters for fine-tuning, allowing us to embed an 8-bit message while maintaining high visual fidelity in the reconstructed image. Thirty fitting experiments on three synthetic images show that the correct key can recover the message without error, while decoding with incorrect keys yields a Bit Error Rate (BER) of 0.543, close to random guessing. Compared with random selection using the key, selecting the least-disturbing edits recovers the message more reliably (one-sided $p = 0.031$), and the average PSNR cost is only 0.091 dB in visual quality. The embedding method transfers to 112 natural images at 256×256 using 4,096 Gaussians. The correct-key BER is 0.000, and decoding under a wrong key stays close to random guessing at 0.520. Our method embeds the payload through three Gaussian parameter types: log-anisotropy, opacity, and color luminance. In a separate nine-fit reduced setting, color luminance is removed, so the payload uses two instead of three parameter types, a 33.3% reduction; all 256 Gaussians remain in the fitted representation. The correct key still recovers the message without error. However, wrong-key BER rises from 0.514 to 0.571, moving farther from random guessing (0.5).

CCS Concepts

• **Computing methodologies → Image representations**.

Keywords

Gaussian Splatting, Steganography

1 Introduction

A 2D Gaussian-based image representation models an image as a set of 2D Gaussian primitives [Zhang et al. 2024]. Meanwhile, people often hide additional information, also known as a payload, in images for purposes such as watermarking. The conventional practice is to modify the pixel values of a given image in an imperceptible way. However, whether information can be stored in the parameters of image-space 2D Gaussian primitives and recovered from the modified parameters remains underexplored.

Image-space neural watermarking and latent-space steganography methods hide the payload in pixels or latent representations, and can be decoded from the reconstructed image [Bui et al. 2023; Tancik et al. 2020; Zhu et al. 2018]. 3D-GSW embeds watermarks in 3D Gaussian scenes and recovers them from rendered images [Jang et al. 2025]. GaussianMarker extracts messages from both Gaussian parameters and rendered images [Huang et al. 2024a]. Our payload is read from the fitted image-space 2D Gaussian parameter set, whereas the rendered image alone is insufficient for decoding. This parameter requirement adds an extra layer of security because an image-only copy does not provide the modified Gaussian parameters required by the decoder. If one refits a new set of Gaussian parameters from the rendered image, the hidden information is not retained. We therefore focus on sharing the parameter set itself.

Our method starts by identifying Gaussians to hide the secret information with the help of a secret key. Among the Gaussians that can be safely modified, the key generates a repeatable set of candidates and maps their parameter changes to the message bits. We then select the least-disturbing combination that encodes the message. We recover the embedded information from those Gaussians using the same secret key. In our tests, the tested wrong keys yield a mean BER close to 0.5. We evaluate our method against two baselines: (1) **random selection with the key**; and (2) **selection of the least-disturbing edits without the key**. Notably, we find that reducing the available parameter types preserves correct-key recovery but moves wrong-key decoding farther from random guessing. We have released our code in the project repository.

2 Method

2D Gaussian Image Representation. We work with a bounded image-space formulation built from 2D Gaussian primitives [Zhang et al. 2024], following the pipeline in Figure 1. This differs from 3D Gaussian Splatting (3DGS), which represents 3D scenes with 3D Gaussian primitives [Kerbl et al. 2023]. We represent an image with N primitives G_i. Each primitive G_i is parameterized by a center offset $\mu_i \in \mathbb{R}^2$, a scale vector $\mathbf{s}_i = (s_{x,i}, s_{y,i})^T \in \mathbb{R}^2$, an in-plane rotation angle $\theta_i \in \mathbb{R}$, an RGB color vector $\mathbf{c}_i \in [0,1]^3$, and an opacity $\alpha_i \in [0,1]$. Let $P = \{(\mu_i, s_{x,i}, s_{y,i}, \theta_i, \mathbf{c}_i, \alpha_i)\}_{i=1}^N$ denote the parameter set of all primitives. We render the image on the normalized 2D image grid $\mathbf{x} = (x, y)$ by accumulated blending,

$$\hat{\mathbf{I}}(\mathbf{x}) = \sum_{i=1}^{N} \alpha_i \, G_i(\mathbf{x}) \, \mathbf{c}_i, \tag{1}$$

where

$$G_i(\mathbf{x}) = \exp\!\Big(-\tfrac{1}{2}(\mathbf{x}-\mu_i)^T \Sigma_i^{-1} (\mathbf{x}-\mu_i)\Big),$$
$$\Sigma_i = R(\theta_i)\,\mathrm{diag}(s_{x,i}^2,\, s_{y,i}^2)\,R(\theta_i)^T,$$

*Corresponding author.
†Also with Computational Light Laboratory, University College London.

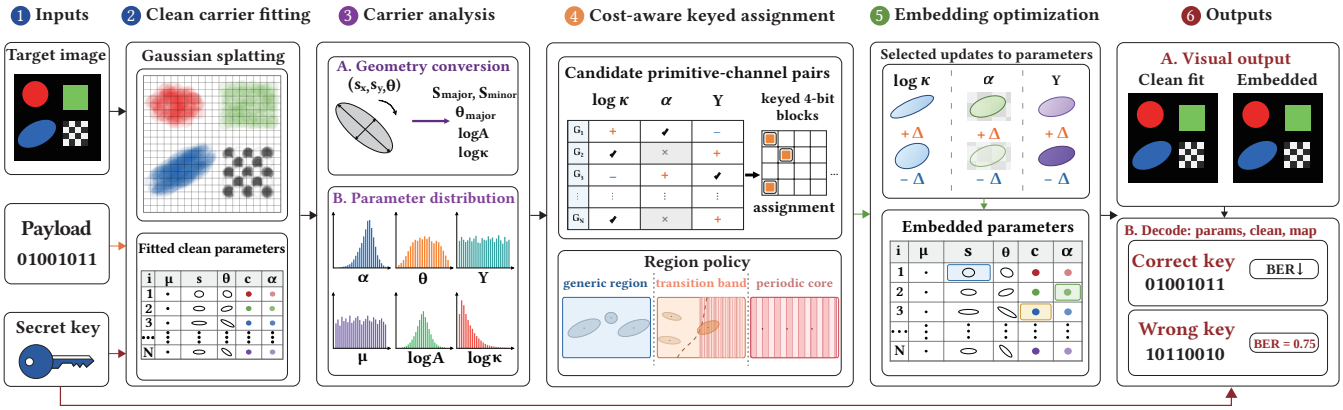

Figure 1: System overview. A clean 2D Gaussian carrier is fitted from an input image. Canonical carrier channels are analyzed and scored. Keyed carrier assignment selects signed parameter updates, and decoding compares correct-key recovery against wrong-key behavior.

with $R(\theta_i)$ the in-plane rotation matrix. There is no transmittance, depth ordering, or multiview consistency in this bounded 2D setting, and our method acts directly on the parameters of G_i to hide the desired payload.

Clean Image Fitting. First, we fit the target image by optimizing P over $\{G_i\}_{i=1}^N$. On the normalized grid, we obtain $\widehat{P}$ by minimizing

$$\widehat{P} \leftarrow \arg\min_P \mathcal{L}\big(\hat{\mathbf{I}}(\cdot;P), \mathbf{I}^\star\big),$$
$$\mathcal{L}\big(\hat{\mathbf{I}}(\cdot;P), \mathbf{I}^\star\big) = \frac{1}{M}\sum_{m=1}^{M} \big\|\hat{\mathbf{I}}(\mathbf{x}_m;P) - \mathbf{I}^\star(\mathbf{x}_m)\big\|_2^2. \tag{2}$$

Here, $\hat{\mathbf{I}}(\mathbf{x}_m;P)$ denotes the image rendered from P at sampled pixel $\mathbf{x}_m$, $m \in \{1,\dots,M\}$, and $\mathbf{I}^\star$ is the target image. Figure 2 shows the source images, clean fits, and embedded reconstructions.

Canonical Parameterization and Carrier Channels. We utilize a subset of Gaussian parameters as the carrier. Following the scale–rotation covariance parameterization used in Gaussian splatting [Huang et al. 2024b; Kerbl et al. 2023], carrier design uses a canonical parameter space instead of the raw $(s_{x,i}, s_{y,i}, \theta_i)$ representation. Let $s_{\text{major},i} = \max(s_{x,i}, s_{y,i})$ and $s_{\text{minor},i} = \min(s_{x,i}, s_{y,i})$. Swapping the axis ordering rotates the angle by $\pi/2$ and wraps it into $[0,\pi)$, giving a canonical major-axis angle $\theta_{\text{major},i}$. From these we derive two scale summaries:

$$\log A_i = \log\big(s_{\text{major},i} s_{\text{minor},i}\big), \qquad \log \kappa_i = \log\left(\frac{s_{\text{major},i}}{s_{\text{minor},i}}\right), \tag{3}$$

giving log-area and log-anisotropy. This removes the axis-ordering ambiguity and makes scales directly comparable. Guided by the fitted distributions in Supplementary Figure S2, we use three carrier channels for the payload: log-anisotropy, opacity, and color luminance Y_i (the luminance of $\mathbf{c}_i$). We exclude rotation because angular edits are less stable; log-area and position are diagnostics.

Parameter-Domain Embedding Formulation. Given a payload $\mathbf{b} \in \{0,1\}^B$ and a secret key k, we modify the fitted clean parameter set $\widehat{P}$ into $\widetilde{P}$. When decoding with k, the payload can be decoded correctly, while decoding under other keys should remain near chance on average. In our experiments, $\mathbf{b}$ is an 8-bit ASCII character, though the formulation handles any bit vector. Unlike existing 3DGS algorithms, our pipeline neither adds nor removes any G_i in the reconstruction process. We keep α_i and $\mathbf{c}_i$ independent to avoid removing a payload channel.

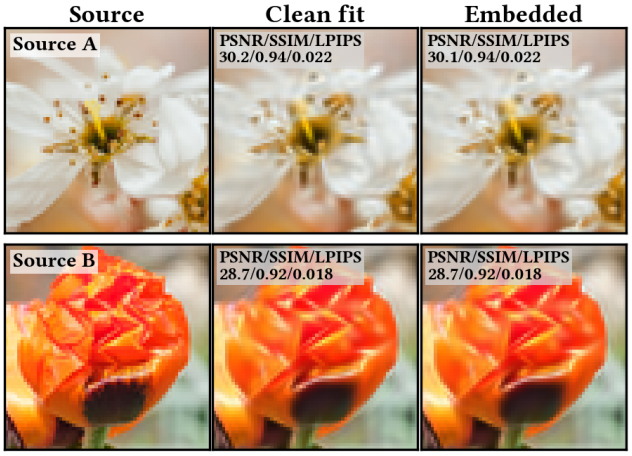

Figure 2: Clean Fit vs. Embedded Reconstruction. We refer to the clean fit as the reconstruction of a source image from our pipeline without payload embedding, whereas the embedded case additionally applies the keyed embedding pipeline. Images are 64 × 64 pixels. Source photographs are by Christian Widell and Christian J. Leuner, respectively.

Cost-Aware Carrier Scoring. For every signed action $(i, \ell, \pm)$, where ℓ indexes the candidate channel, we define four cost terms: rendering sensitivity $\sigma_{i,\ell}$, distribution shift $d^{\pm}_{i,\ell}$, visual importance $v_{i,\ell}$, and outlier-value penalty $t_{i,\ell}$. Their weighted sum is $\rho^{\pm}_{i,\ell} = w_s\sigma_{i,\ell} + w_d d^{\pm}_{i,\ell} + w_v v_{i,\ell} + w_t t_{i,\ell}$. A sensitivity analysis of these weights is provided in Supplementary Section S7.

Structure-Aware Region Policy for Periodic Images. Since periodic structures are visually sensitive to parameter perturbations, we partition the fitted Gaussians into three regions with different carrier permissions and perturbation strengths: the "periodic core", "transition band", and "generic region". We estimate a dominant spatial frequency from the source and determine the stripe-normal direction and its orthogonal tangent. For each Gaussian we measure three components contributing to the periodicity score: repetition intensity r_i, direction alignment a_i, and normalized edge support e_i. Their product defines the periodicity score $\phi_i = r_i a_i e_i$. See Supplementary Section S2 for details.

- "Periodic core" — values above the 75th percentile. It retains only Y_i, with a scale factor of 0.5.
- "Transition band" — values above the 50th percentile, after excluding core members; lower-percentile primitives can also enter when edge and repetition strength are high. It uses scale factors of 0.5 for α_i and 0.75 for Y_i.
- "Generic region" — the rest. These use the payload-channel set $(\log \kappa_i, \alpha_i, Y_i)$ with a perturbation scale factor of 1.0.

We select these percentile cutoffs empirically in preliminary experiments. They remain fixed in all reported runs and may not be optimal for every image. Not all eligible G_i are safe carriers: we exclude rotation entirely, and further filter out candidate actions located in high-edge, high-importance, or stripe-aligned periodic positions before assignment. We organize the remaining carrier actions into keyed 4-bit blocks: the key determines the initial candidate signed actions and the block parity mapping from signed actions to payload bits. Block assignment then follows a syndrome-inspired method [Filler et al. 2011]: signed actions are selected according to the region-dependent channel rules and coverage balancing constraints, which limit the concentration of selected actions. Concretely, the assignment minimizes the total action cost $\sum_{i,\ell} \rho^{\pm}_{i,\ell}$ subject to the key-induced parity constraints. A signal-to-noise score then refines the less reliable block assignments. In periodic sources, the cost avoids luminance edits where they are most visible, preferring scale $(\log \kappa_i)$ and opacity (α_i) edits.

Embedding Optimization and Regularization. Once a keyed assignment is in place, we fine-tune selected carriers to embed the payload while keeping image and parameter distribution close to the clean fit. We minimize

$$\begin{aligned}\mathcal{L}_{\text{embed}} = \lambda_{\text{msg}}\,\mathcal{L}_{\text{msg}} + \lambda_{\text{fid}}\,\mathcal{L}_{\text{fid}} \\ + \lambda_{\text{imp}}\,\mathcal{L}_{\text{imp}} + \lambda_{\text{key}}\,\mathcal{L}_{\text{key}} + \lambda_{\text{aux}}\,\mathcal{L}_{\text{aux}}.\end{aligned} \tag{4}$$

The five terms enforce message recovery, rendered-image fidelity, parameter-distribution similarity, key specificity, and auxiliary robustness, respectively. See Supplementary Section S3 for details.

Decoding and Wrong-Key Evaluation. Decoding uses the embedded parameters $\widetilde{P}$, the corresponding clean reference parameters $\widehat{P}$, and an assignment map produced by the encoder under key k. The map specifies which parameter changes to read and how to convert them into message bits. Decoding reads the realized action scores from the deviation $\widetilde{P} - \widehat{P}$, thresholds them into per-action binary decisions, and applies the block parity mapping to recover $\hat{\mathbf{b}}_k$. See Supplementary Section S2 for the extraction rule. The primary metric is bit error rate,

$$\text{BER}(\hat{\mathbf{b}}_k, \mathbf{b}) = \frac{1}{B}\sum_{j=1}^{B} \mathbf{1}\left[\hat{b}_{k,j} \neq b_j\right]. \tag{5}$$

Correct-key BER checks whether the payload survives embedding. Wrong-key BER evaluates decoding under assignments generated from unrelated keys. Wrong-key BER near 0.5 indicates key specificity, with larger deviations indicating weaker specificity.

3 Implementation

Training. We first optimize a clean Gaussian fit and then fine-tune the selected carrier parameters for payload embedding. Optimization settings accompany the code.

Data. We evaluate three $3 \times 64 \times 64$ RGB targets—a flat gradient, a periodic stripe pattern, and a checkerboard pattern—each fit with 10 random seeds and 256 Gaussian primitives (see Supplementary Figure S1). For the 64×64, 256-primitive setting, the 8-bit payload corresponds to 0.031 bits per primitive, or 0.00195 bits per rendered pixel. The payload is carried by the selected carrier actions.

4 Evaluation

Setup. The synthetic targets are a flat gradient, a periodic stripe, and a checkerboard (see Supplementary Figure S1). Synthetic targets isolate flat, periodic, and mixed structure. For each synthetic target, we independently fit 256 Gaussians using 10 random seeds that were not used during method development, giving 30 fitted Gaussian sets in total. We evaluate every method on the same 30 sets. Under a fixed k, each set receives an 8-bit payload. We report three metrics: correct-key $\text{BER}_{\text{ck}} \downarrow$, mean wrong-key BER_{wk} over 16 wrong keys (target 0.5), and ΔPSNR from the clean fit. Both optimization and checkpoint selection use the same 16 wrong keys.

Comparison on Synthetic Targets. Table 1 compares our method with two baselines. Both use the same allowed Gaussian parameter edits, 8-bit payload, and spatial restrictions as our method. *Matched-random* uses the key but chooses edits at random. *Greedy no-key* chooses the lowest-cost allowed edits without a key. We do not compare against radiance-field or 3DGS watermarking methods, which evaluate a different 3D setting. Our method and greedy no-key recover all 240 payload bits. Matched-random makes payload errors on 5 of the 30 sets and matches our method on the other 25. The paired permutation p-values are 0.031 (one-sided) and 0.062 (two-sided). Mean wrong-key BER is 0.543 for our method and 0.565 for matched-random. Our wrong-key BER stays between 0.4 and 0.6 on 27 of the 30 sets. Matched-random has a smaller ΔPSNR, but it does not recover every payload. Most ΔPSNR values for our method are near 0, while a few outliers raise the mean. Filtering leaves 182–422 eligible parameter edits per set, but only 2–5 Gaussians are modified.

Ablation Study. Table 2 removes one component at a time on the nine-fit development set. Without the Sinkhorn distribution term [Cuturi 2013], some correct-key errors appear, but wrong-key BER remains near 0.5. The two-channel variant uses only log-anisotropy and opacity. It recovers every payload bit, but its wrong-key BER increases to 0.571. The most periodic region permits only

Table 1: Results on 30 fitted Gaussian sets with an 8-bit payload. Values are reported as mean ± standard deviation.

Method	$\mathrm{BER}_{\mathrm{ck}}\downarrow$	$\mathrm{BER}_{\mathrm{wk}}\to 0.5$	$\Delta\mathrm{PSNR}\downarrow$
Matched random	$0.042_{\pm 0.111}$	$0.565_{\pm 0.066}$	$0.066_{\pm 0.129}$
Greedy no-key	$\mathbf{0.000}_{\pm 0.000}$	N/A	$0.075_{\pm 0.206}$
Ours	$\mathbf{0.000}_{\pm 0.000}$	$\mathbf{0.543}_{\pm 0.059}$	$0.091_{\pm 0.154}$

Table 2: Ablation results on a separate nine-fit development set. All rows use the same nine fits and 8-bit payload. Values are reported as mean ± standard deviation.

Ablation	$\mathrm{BER}_{\mathrm{ck}}\downarrow$	$\mathrm{BER}_{\mathrm{wk}}\to 0.5$	$\Delta\mathrm{PSNR}\downarrow$
Ours	$\mathbf{0.000}_{\pm 0.000}$	$\mathbf{0.514}_{\pm 0.026}$	$0.584_{\pm 1.348}$
− Sinkhorn	$0.028_{\pm 0.055}$	$0.512_{\pm 0.048}$	$0.509_{\pm 0.846}$
2-channel only	$\mathbf{0.000}_{\pm 0.000}$	$0.571_{\pm 0.057}$	$1.021_{\pm 1.869}$
− region policy	$0.069_{\pm 0.110}$	$0.567_{\pm 0.054}$	$0.011_{\pm 0.024}$
− coverage balancing	$0.042_{\pm 0.125}$	$0.523_{\pm 0.039}$	$0.899_{\pm 1.496}$

luminance edits, so this region cannot carry the payload in the two-channel variant. Removing the region policy or coverage balancing, which spreads edits across image regions, also introduces correct-key errors and moves wrong-key BER away from 0.5. The small ΔPSNR without the region policy is not a comparable improvement because that variant does not recover every bit.

Robustness. Using the seed-0 fit of each synthetic target, we test all 256 possible 8-bit byte values. The average correct-key BER is 0.030, while wrong-key decoding stays close to chance (0.500). In total, 623/768 target-byte cases decode correctly. Errors concentrate on the periodic stripe target, the hardest case. Supplementary Section S6 and Figure S3 report recovery and ΔPSNR at different payload lengths on the nine-fit development set. We then add noise to the payload-carrying parameters of nine fitted sets. The uniform and Gaussian noise have standard deviations of approximately 0.0058 and 0.005, respectively. Supplementary Table S1 reports four noise scales over 27 evaluations each. Mean correct-key BER rises to 0.042 under uniform noise and 0.060 under Gaussian noise, while wrong-key BER stays near 0.5. After 200 additional optimization steps using only the reconstruction loss, correct-key BER rises to 0.264. A new Gaussian fit of the rendered image raises correct-key BER to about 0.75, so recovery fails.

Natural Images. We use 24 Kodak images [Franzen 1999] and 100 DIV2K validation crops [Agustsson and Timofte 2017] prepared at $3 \times 256 \times 256$ to embed the fixed 8-bit payload. All methods succeed with 8,192 Gaussians, so we reduce the count. We select 4,096 Gaussians using a separate set of 12 images and evaluate all three methods on the remaining 112 images. Our method and greedy no-key recover all 112 payloads, while matched-random recovers 110. Our wrong-key BER remains close to chance at 0.520. Mean clean-fit PSNR is 33.68 dB, with 97/112 images above 30 dB. On the same covers, pretrained RoSteALS recovers 109 of 112 bytes at 26.77 dB, while our method recovers all 112 at 57.24 dB (Supplementary Table S2). RoSteALS needs no key for decoding.

Table 3: Results on 112 held-out RGB images at $3 \times 256 \times 256$ using 4,096 Gaussians and a fixed 8-bit payload. Recovery requires all eight bits to be correct. Values are mean ± standard deviation.

Method	Payloads recovered ↑	$\mathrm{BER}_{\mathrm{ck}}\downarrow$	$\mathrm{BER}_{\mathrm{wk}}\to 0.5$	$\Delta\mathrm{PSNR}\downarrow$
Matched random	110/112	$0.003_{\pm 0.026}$	$0.525_{\pm 0.019}$	$0.013_{\pm 0.023}$
Greedy no-key	**112/112**	$\mathbf{0.000}_{\pm 0.000}$	N/A	$0.006_{\pm 0.014}$
Ours	**112/112**	$\mathbf{0.000}_{\pm 0.000}$	$\mathbf{0.520}_{\pm 0.029}$	$0.009_{\pm 0.016}$

Conclusion. We introduce a novel steganography method for image representations based on 2D Gaussians, embedding payloads directly in the Gaussian primitives rather than image pixels. Experiments on natural images and controlled synthetic cases—flat, periodic, and mixed structures—show that the correct key reliably recovers the payload, while other keys yield near-random decoding. Our method adds a new level of complexity in steganography methods, benefiting content creators with additional security against malicious activities.

References

Eirikur Agustsson and Radu Timofte. 2017. NTIRE 2017 Challenge on Single Image Super-Resolution: Dataset and Study. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 1122–1131. doi:10.1109/CVPRW.2017.150

Tu Bui, Shruti Agarwal, Ning Yu, and John Collomosse. 2023. RoSteALS: Robust Steganography Using Autoencoder Latent Space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*. 933–942. doi:10.1109/CVPRW59228.2023.00100

Marco Cuturi. 2013. Sinkhorn Distances: Lightspeed Computation of Optimal Transport. In *Advances in Neural Information Processing Systems 26*, C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger (Eds.). Curran Associates, Inc., 2292–2300. arXiv:1306.0895.

Tomáš Filler, Jan Judas, and Jessica Fridrich. 2011. Minimizing Additive Distortion in Steganography Using Syndrome-Trellis Codes. *IEEE Transactions on Information Forensics and Security* 6, 3 (2011), 920–935. doi:10.1109/TIFS.2011.2134094

Rich Franzen. 1999. Kodak Lossless True Color Image Suite. https://r0k.us/graphics/kodak/. Accessed 2026.

Binbin Huang, Zehao Yu, Anpei Chen, Andreas Geiger, and Shenghua Gao. 2024b. 2D Gaussian Splatting for Geometrically Accurate Radiance Fields. In *ACM SIGGRAPH 2024 Conference Papers*. doi:10.1145/3641519.3657428

Xiufeng Huang, Ruiqi Li, Yiu ming Cheung, Ka Chun Cheung, Simon See, and Renjie Wan. 2024a. GaussianMarker: Uncertainty-Aware Copyright Protection of 3D Gaussian Splatting. In *Advances in Neural Information Processing Systems*, Vol. 37. doi:10.52202/079017-1040

Youngdong Jang, Hyunje Park, Feng Yang, Heeju Ko, Euijin Choo, and Sangpil Kim. 2025. 3D-GSW: 3D Gaussian Splatting for Robust Watermarking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 5938–5948. doi:10.1109/CVPR52734.2025.00557

Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 2023. 3D Gaussian Splatting for Real-Time Radiance Field Rendering. *ACM Transactions on Graphics* 42, 4 (2023). doi:10.1145/3592433

Matthew Tancik, Ben Mildenhall, and Ren Ng. 2020. StegaStamp: Invisible Hyperlinks in Physical Photographs. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2114–2123. doi:10.1109/CVPR42600.2020.00219

Xinjie Zhang, Xingtong Ge, Tongda Xu, Dailan He, Yan Wang, Hongwei Qin, Guo Lu, Jing Geng, and Jun Zhang. 2024. GaussianImage: 1000 FPS Image Representation and Compression by 2D Gaussian Splatting. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 327–345. doi:10.1007/978-3-031-72673-6_18

Jiren Zhu, Russell Kaplan, Justin Johnson, and Li Fei-Fei. 2018. HiDDeN: Hiding Data With Deep Networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 682–697. doi:10.1007/978-3-030-01267-0_40