MultiNeRF: Multiple Watermark Embedding for Neural Radiance Fields

Yash Kulthe[1] Andrew Gilbert[1] John Collomosse[1,2]
[1]CVSSP, University of Surrey, UK [2]Adobe Research
{y.kulthe, a.gilbert, j.collomosse}@surrey.ac.uk collomos@adobe.com

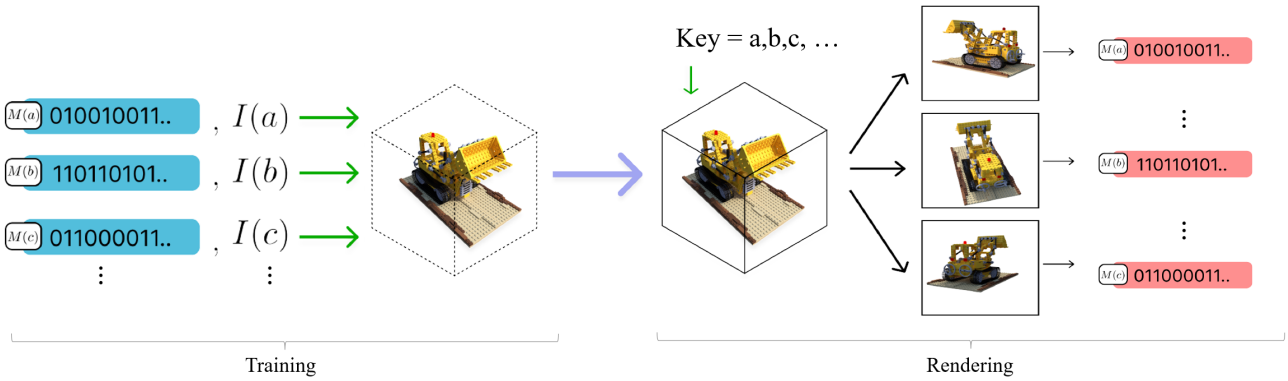

Figure 1. MultiNeRF embeds multiple watermarks within the representation learned by a NeRF model (TensoRF) at training time. Watermarks are keyed by a unique ID specified at rendering time to trigger the embedding of the watermark into the image independent of the viewing position.

Abstract

*We present **MultiNeRF**[1], a 3D watermarking method that embeds multiple uniquely keyed watermarks within images rendered by a single Neural Radiance Field (NeRF) model, whilst maintaining high visual quality. Our approach extends the TensoRF NeRF model by incorporating a dedicated watermark grid alongside the existing geometry and appearance grids. This extension ensures higher watermark capacity without entangling watermark signals with scene content. We propose a FiLM-based conditional modulation mechanism that dynamically activates watermarks based on input identifiers, allowing multiple independent watermarks to be embedded and extracted without requiring model retraining. MultiNeRF is validated on the NeRF-Synthetic and LLFF datasets, with statistically significant improvements in robust capacity without compromising rendering quality. By generalizing single-watermark NeRF methods into a flexible multi-watermarking framework, MultiNeRF provides a scalable solution for 3D content attribution.*

1. Introduction

Neural Radiance Fields (NeRFs) [25] are high-fidelity 3D scene representation method, enabling photorealistic novel view synthesis. Their uses include online gaming, immersive experiences, and large-scale metaverse environments [11, 19, 39]. However, this introduces challenges in intellectual property (IP) protection, as NeRF models, whether representing products, scenes, or avatars, can be expensive to produce and be easily shared or leaked, Establishing the provenance of a NeRF model is crucial for asserting ownership and rights, mitigating these risks, and even opening up novel compensation frameworks for their reuse [9].

Digital watermarking has been a cornerstone of IP protection for visual media, including images [5, 12, 44] and video [14]. However, these methods fall short in the NeRF context because they protect only the 2D outputs (the rendered images) rather than the underlying 3D representation itself. Recent work has focused on watermarking models directly, for example, text-to-image diffusion models [13] and NeRFs [17, 23, 31, 41]. Embedding watermark signals into the NeRF representation ensures persistent identification of models even under novel rendering views.

This paper introduces MultiNeRF, a framework for conditionally embedding *multiple* watermarks within a NeRF model. Existing NeRF watermarking techniques are limited to encoding a single watermark within the rendered image, typically with low capacity (*e.g.* 16–48 bits). Even within the broader image watermarking literature [5, 12, 33, 44], it is difficult to surpass capacities of this magnitude order (*i.e.* < 100 bits) whilst maintaining acceptable visual quality for creative use cases. However, this is insufficient even

[1]Project page: https://yash-research.github.io/multinerf/

to accommodate a URL *e.g.* to an end-user license agreement. NeRF raises the intriguing possibility of embedding multiple distinct watermarks within a single model, each with small capacity, but together can encode higher payloads. Further, multiple independently keyed watermarks admit scenarios requiring multiple licenses or stakeholders (Fig. 1). For example, in collaborative environments such as co-developed metaverse worlds, different contributors may need distinct watermarks to establish ownership or track usage. Our technical contributions are:

1. **Watermark Grid**. We introduce a dedicated watermark grid alongside the existing geometry and appearance grids of the learned NeRF representation. This grid prevents the entanglement of the encoded watermark with scene content, improving the watermark capacity and preserving the rendering quality with minimal model size overhead.
2. **Conditional Modulation**. We introduce FiLM-based modulation [28], applying a lightweight input embedding network to encode watermark-specific identifiers and dynamically control the activation of watermark features. This enables the conditional rendering of multiple watermarks within the same NeRF model.

Fig. 4 illustrates our architecture to achieve multiple watermark embedding. We train the modified NeRF model end-to-end, using a pre-trained HiDDeN decoder [44] as the base model for watermark retrieval. To ensure robustness, we augment training with differentiable noise sources. We later show (Sec. 5) our approach to deliver statistically significant improvements in capacity (*i.e.* the ability to store many watermarks with high bit accuracy), without substantial quality change. We demonstrate this for two standard NeRF datasets of multiple scenes (NeRF-Synthetic [25] and LLFF [24]), thus extending NeRF watermarking for collaborative and commercial use.

2. Related Work

Visual Watermarking and Media Provenance. Traditional watermarking techniques embed information in the spatial [15, 32] or frequency domains [22, 27, 29]. Similarly, recent works have explored steganography in images [4], NeRFs [20] and 3D Gaussian Splatting [21]. Deep watermarking methods such as HiDDeN [44], StegaStamp [33], RoSteALS [6], SSL [12], InvisMark [37] and TrustMark [5] use learned embedding networks to improve imperceptibility and robustness against common transformations. Watermarking has been used to trace digital content's provenance (including ownership and rights) in combination with cryptographic metadata standards *e.g.* C2PA. These standards attach signed metadata to digital assets, but metadata is frequently removed during redistribution. Watermarking offers an alternative by embedding identifiers directly into content, so that provenance persists when metadata is stripped [9]; MultiNeRF explores NeRF watermarking to achieve the same goal.

Watermarking NeRF models. While deep watermarking has been studied for 2D images and videos, its application to 3D generative models, such as NeRFs, is relatively new. Conventional 2D watermarking fails to ensure persistence across novel viewpoints, leading to recent research into NeRF watermarking methods [17, 23, 31, 41]. CopyRNeRF [23] embeds watermark signals in the color feature field to ensure extraction from arbitrary viewpoints. In contrast, NeRFProtector [31] focuses on embedding a watermark from the start of training a NeRF scene. WateRF [17] introduces a frequency-based embedding, enhancing resilience to noise and compression. Existing NeRF watermarking methods are limited in capacity and cannot handle multiple watermarks, which we address with MultiNeRF.

Watermarking in Generative Models. As generative models in images [1], video [16], and 3D asset creation [25], protecting model IP is a concern. Model provenance methods include the watermarking of training data [2, 30], model fingerprinting [42], and proactive tagging techniques [34]. Recent work in diffusion model watermarking, such as Stable Signature [13], introduces an in-model watermarking approach that fine-tunes the decoder of a latent diffusion model to embed identifiers directly into generated images. Other methods, such as Tree-Ring [36], inject patterns into the noise initialization step of diffusion models. MultiNeRF extends these approaches to 3D by embedding provenance signals into NeRF representations, so the watermark persists across views while enabling multiple uniquely keyed watermarks within a single model.

3. Preliminaries (TensoRF)

This paper builds on TensoRF [7], an explicit tensor-based representation for neural radiance fields (NeRFs). Unlike the original NeRF [25], which uses multi-layered perceptions (MLPs), TensoRF [7] represents a scene as a set of factorized tensors: the geometry grid, denoted by $G_\sigma \in \mathbb{R}^{I\times J\times K}$, which encodes the volume density σ at each voxel in the 3D grid; and the appearance grid $G_c \in \mathbb{R}^{I\times J\times K\times P}$, which encodes the view-dependent color c; where I, J, K represent the resolutions of the feature grid along the X, Y, Z axes. P denotes the number of appearance feature channels. Given a 3D location $\mathbf{x} = (x, y, z) \in \mathbb{R}^3$ and a view direction $\mathbf{d}$, trilinear interpolation is used to sample the two grids and the corresponding density σ and color c is estimated by:

$$\sigma, \mathbf{c} = (G_\sigma(\mathbf{x}), S(G_\mathbf{c}(\mathbf{x}), \mathbf{d})) \tag{1}$$

Here, S is a decoding function, implemented either as a small MLP or Spherical Harmonic (SH) function that covers the appearance features and view direction to an RGB color. The trilinearly interpolated grids G_σ and G_c are represented as:

$$G_\sigma(\mathbf{x}) = \sum_r \sum_m \mathcal{A}^m_{\sigma,r}(\mathbf{x}),$$
$$G_\mathbf{c}(\mathbf{x}) = \mathbf{B}\Big(\bigoplus \big[\mathcal{A}^m_{\mathbf{c},r}(\mathbf{x})\big]_{m,r}\Big) \tag{2}$$

Where, $\mathcal{A}^m_{\sigma,r}(\mathbf{x})$ and $\mathcal{A}^m_{\mathbf{c},r}(\mathbf{x})$ are factorized components of the density and appearance tensors, indexed by mode *m* and rank *r*. $\mathbf{B}$ matrix acts as a global dictionary capturing correlations across the scene. While $\bigoplus$ denotes the concatenation of tensor components.

TensoRF uses differentiable volume rendering. It integrates the radiance values along the rays cast through the scene, which samples *Q* points along the ray. Given a ray passing through a 3D point, the final pixel color *C* is computed as:

$$C = \sum_{q=1}^{Q} \tau_q \left(1 - \exp(-\sigma_q \Delta_q)\right) c_q \tag{3}$$

where, σ_q and c_q are the density and color of the sampled point x_q, Δ_q is the step size along the ray, and τ_q is the transmittance.

$$\tau_q = \exp\left(-\sum_{p=1}^{q-1} \sigma_p \Delta_p\right) \tag{4}$$

4. Methodology

Since the TensoRF decomposes a scene into two separate grids for geometry and appearance, MultiNeRF extends this framework to encode multiple watermarks with a further dedicated grid. In contrast to prior approaches that embed only a single watermark (*e.g.* [17, 31]), our method accommodates multiple distinct messages, each identified by a unique key (ID), which forms an additional model input. Fig. 2 illustrates the pipeline.

4.1. Constructing the Watermarking module

As NeRF models only take position and view direction as inputs to the model, we need an input embedding layer to condition the NeRF and embed the watermark accordingly, to enable message switching. We let $I(n)$ be the integer IDs, each corresponding to a distinct watermark message $M(n)$, and introduce a learnable embedding layer $Emb()$ which maps $I(n)$ to e_n a 16 dimensional message vector.

$$e_n = \text{Emb}(I(n)) \tag{5}$$

Emb is a learnable small MLP, implemented as a three-layer network with ReLU activations and e_n is the 16-dimensional embedding vector, which will serve as a condition to modulate and select from multiple learned watermarks.

Watermark Grid. TensoRF [7] decomposes a scene into two explicit 3D grids: G_σ for geometry (density) and $G_\mathbf{c}$ for appearance (color). Directly embedding the watermark into the existing appearance grid $G_\mathbf{c}$ would risk entangling the watermark features with the scene color, potentially degrading both. We propose an additional grid $G_\mathbf{w}$ of the same spatial resolution $(I \times J \times K)$ but containing watermark-specific feature channels. Given, at any 3D location $\mathbf{x} = (x, y, z)$,

$$w(\mathbf{x}) = \bigoplus \left[\mathcal{A}^m_{\mathbf{w},r}(\mathbf{x})\right]_{m,r} \tag{6}$$

Watermark modulation. A design challenge in embedding multiple watermarks is ensuring that the model 'activates' only the selected watermark's features. To this end, we draw inspiration from FiLM [28]; its FiLM layer can influence the feature space computation using feature-wise affine transform based on external information. In our context, to embed multiple watermarks, our external information is the embedded watermark ID, i.e., e_n. Our watermark modulation has a modulator network, which in our case is a learnable linear layer $Mod()$, implemented as Linear(16→48), which takes e_n as input and outputs a pair of 24-dimensional modulation parameters to scale and shift the feature space of the watermark features: scale γ_n and shift β_n:

$$[\gamma_n, \beta_n] = \text{Mod}(\mathbf{e}_n), \tag{7}$$

Thus, each $w(x)$ channel is conditioned based on the embedding e_n. We then compute:

$$\mathbf{w}'(\mathbf{x}) = \gamma_n \odot \mathbf{w}(\mathbf{x}) + \beta_n, \tag{8}$$

Where $\odot$ is the Hadamard product. The resulting $\mathbf{w}'(\mathbf{x})$ represents the *modulated watermark features* specific to $I(n)$. Intuitively, γ_n and β_n activate certain dimensions of the watermark grid differently for each watermark ID.

Merging watermark and appearance. To incorporate these modulated watermark features with appearance, we pass only $G_c(\mathbf{x})$ and viewing direction to the S MLP, which is a color decoding function of TensoRF (see [7] for details), and we apply the modulated watermark features to the last linear layer of the S MLP (see Fig. 3 for justification) :

$$\tilde{\mathbf{c}}(\mathbf{x}) = S(\mathbf{G}_c(\mathbf{x}), \mathbf{w}'(\mathbf{x}), \mathbf{d}), \tag{9}$$

where $\tilde{\mathbf{c}}(\mathbf{x})$ is now a watermark color at location $\mathbf{x}$. The rest of TensoRF's volume rendering proceeds unchanged. Because each ID yields different $\{\gamma_n, \beta_n\}$,, the final color $\tilde{\mathbf{c}}(\mathbf{x})$ implicitly contains an ID-specific watermark pattern. As a result, any novel view rendered from the watermarked NeRF model will contain an embedded signature that can be extracted with the watermark decoder.

Differentiable Augmentation layer. To promote robustness against image attacks, we train with differentiable augmentations on the fully rendered image. Each image is transformed by a small set of augmentations (brightness, contrast, color jiggle, Gaussian blur, Gaussian noise, hue,

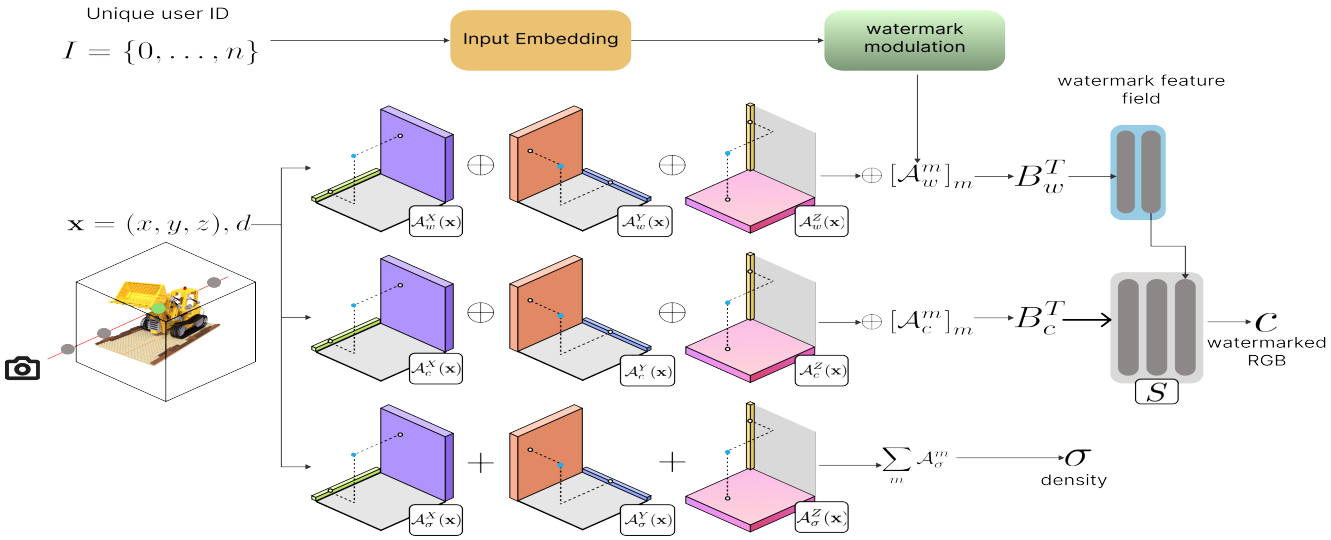

Figure 2. MultiNeRF extends TensoRF by introducing a watermark grid A_w alongside the geometry A_σ and appearance A_c grids. Unique watermark IDs $(1..n)$ are first encoded via a learnable embedding network into compact vectors that are then transformed into per-channel scaling (γ) and shifting (β) parameters. These parameters modulate the watermark grid's features, ensuring that each distinct message is selectively activated and merged with the appearance grid during inference.

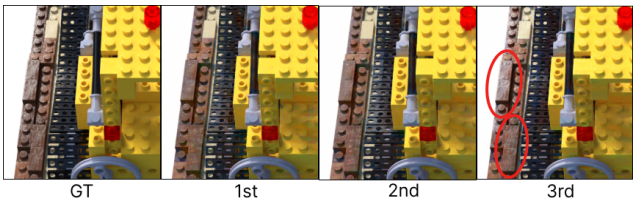

Figure 3. Left to right: Ground Truth, modulation at 1st layer, 2nd layer, and 3rd (last) layer of the color MLP. The last layer preserves best appearance, i.e., it retains better reflection details.

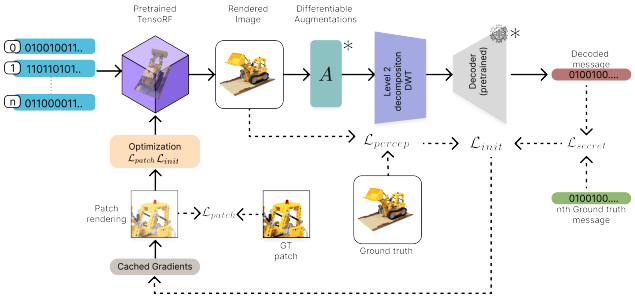

Figure 4. MultiNeRF training process. A learnable encoder transforms multiple distinct watermark IDs into compact embedding vectors. MultiNeRF is trained with an end-to-end HiDDEN decoder module that renders images passing through differentiable noise augmentations. Perceptual (Eq. (12)) and patch-based (Eq. (13)) reconstruction losses balance quality against accuracy (Eq. (11)).

posterize, RGB shift, saturation, median blur, box blur, motion blur, sharpness, and differentiable JPEG compression) before passing them into the watermark decoder function. Formally, if $\tilde{\mathbf{I}}$ is the full rendered image, we apply a random pair of augmentations from $\mathbf{A}$ from a set of $\{A_1, ..., A_m\}$:

$$\hat{\mathbf{I}} = \mathcal{A}(\tilde{\mathbf{I}}), \tag{10}$$

where $\hat{\mathbf{I}}$ is an augmented image, and we pass this image to the watermark decoder.

4.2. Training MultiNeRF to embed watermarks

We begin by training a TensoRF NeRF model, using the geometry and appearance grids to initialize those parts of our MultiNeRF architecture. We initialize the watermark decoder using a pre-trained HiDDeN decoder following [44]. The remaining components are initialized with white noise. Training occurs in two phases.

Phase 1. We freeze the watermark decoder D. We generate a random watermark ID $I(n)$, with $\mathbf{m}_n$ the ground truth message for $n-th$ ID. In each training iteration, we render a full-resolution image $\mathbf{I} \in \mathbb{R}^{H \times W \times 3}$. As directly injecting watermarks into the spatial domain can be vulnerable to image compression, we apply a level 2 decomposition of Discrete Wavelet Transform (DWT) on the rendered image (after [17]) and use the LL_2 sub-band as it retains most of the image energy and structure. This sub-band image (denoted $\mathbf{I}_{LL_2}$) is then passed to the watermark decoder D obtaining a decoded message $\mathbf{m}'_n$. A BCE loss is calculated between secrets $\mathbf{m}_n$ and $\mathbf{m}'_n$:

$$\mathcal{L}_{secret} = \text{BCE}(\mathbf{m}_n, \mathbf{m}'_n) \tag{11}$$

We also compute a perceptual loss $\mathcal{L}_{percept}$ via Watson-

VGG [10] to create a total loss for this first phase of training:

$$\mathcal{L}_{init} = \lambda_m \mathcal{L}_{secret} + \lambda_i \mathcal{L}_{percept} \quad (12)$$

Phase 2. Training continues end-to-end via a patch-based process using deferred backpropagation [38]. Since full-resolution optimization is memory-intensive. Therefore, we sample patches and re-render images accordingly. A patch loss $\mathcal{L}_{patch}$ is then calculated, which uses the RGB loss across the rendered pixels, SSIM loss, and total variation regularization:

$$\mathcal{L}_{patch} = \lambda_{rgb} \mathcal{L}_{RGB} + \lambda_{TV} \mathcal{L}_{TV} + \lambda_{SSIM} \mathcal{L}_{SSIM} \quad (13)$$

We unfreeze the decoder D and introduce the differentiable augmentations A at each iteration on the rendered image, on which the 2-level DWT is applied and passed into the watermark decoder D.

5. Experiments and Discussion

5.1. Experimental Setup

Datasets: As the TensorRF [7] framework does not support unbounded scenes, we use two standard datasets in NeRF watermarking research: Nerf-Synthetic [25] (hereafter, SYN) and the LLFF datasets [24]. SYN consists of eight representative scenes: *chair, drums, ficus, hotdog, lego, materials, mic, ship*; and the LLFF dataset consists of eight scenes: *fern, flower, fortress, horns, leaves, orchids, room, trex*. We train and test MultiNeRF on the standard partitions of these datasets.

Baselines: Since no multiple-watermark frameworks exist for NeRFs, we compare our method with two recent NeRF watermarking methods for the single watermarking task: WateRF [17] and NeRFProtector [31]. For fair comparisons, we use the models that those baselines have implemented in their methods; for WateRF, we choose TensoRF [7] as the NeRF model, and for NeRFProtector [31] we use Instant-NGP [26]. As no prior work embeds multiple watermarks into a single NeRF, we tune WateRF [17] by adding an input embedding layer to condition it to embed multiple watermarks (denoted 'WateRF-modified'). Similar to our MultiNeRF setup, this input embedding layer outputs a 16-dimensional vector based on watermark ID, and is attached to the appearance features of TensoRF before being passed into the color decoding MLP, and the model is trained.

Metrics: We use bit accuracy to evaluate the accuracy of the decoder for a given capacity. For visual quality, we measure PSNR, SSIM [35], and LPIPS [40] distances between the ground truth and watermarked image.

Training setup: MultiNeRF is implemented in PyTorch and trained with a batch size of 1 on a single NVIDIA RTX 4080. The ADAM optimizer [18] is used with $\beta_1 = 0.9$ and $\beta_2 = 0.99$, with exponential learning rate decay. Grid parameters are optimized with an initial learning rate of 0.02, and the basis matrix for all grids has a learning rate of $1e{-}3$. The λ_i and λ_m in Eq. (12) is set to 0.05 and 0.95; while for the patch loss, in Eq. (13), we set λ_{RGB}, λ_{TV} and λ_{SSIM} to 0.1, 0.02, and 0.20 respectively.

5.2. Single watermark embedding evaluation

We evaluate two variants of our model: MultiNeRF-Noised (with noise augmentations) and MultiNeRF (without). All methods embed a single 48-bit message. Since bit accuracy varies slightly by different messages, we average results over 50 unique watermarks for all the synthetic and LLFF datasets. We enforce a minimum Hamming distance to ensure message distinctness. Note that NeRFProtector reported on only 3 scenes from SYN and LLFF datasets [31]. Using the official implementation, we cannot reconstruct 3 of the scenes *(ficus, flower, leaves)* and omit these outliers from their average.

Tab. 1 shows that MultiNeRF achieves a mean bit accuracy of 93.18% on SYN outperforming both WateRF (91.51%) and NeRFProtector (90.81%); and like WaterRF saturates performance on LLLF at $\sim$ 99%. MultiNeRF-Noised shows slightly lower accuracy but improved robustness (c.f. Sec. 5.4). MultiNeRF generally achieves comparable quality and slightly higher accuracy at the single watermarking task.

Method (on SYN)	Avg.	Chair	Drums	Ficus	Hotdog	Lego	Materials	Mic	Ship
WateRF [17]	91.51	98.31	92.19	79.83	96.21	93.16	82.33	95.92	94.10
NeRFProtector [31]	90.81	96.41	89.73	-	93.47	90.12	84.05	90.39	91.54
MultiNeRF (ours)	93.18	98.35	95.14	83.06	96.97	94.86	85.16	96.89	95.03
MultiNeRF-Noised (ours)	89.70	92.60	93.61	78.60	94.36	92.49	83.54	89.72	92.65
Method (on LLFF)	Avg.	Fern	Flower	Fortress	Horns	Leaves	Orchids	Room	Trex
WateRF [17]	99.32	99.75	99.56	99.95	99.92	99.53	96.07	99.89	99.91
NeRFProtector [31]	95.73	94.68	-	99.58	98.77	-	82.23	99.73	99.37
MultiNeRF (ours)	99.23	99.39	99.48	99.82	99.87	99.68	95.92	99.77	99.88
MultiNeRF-Noised (ours)	98.55	99.04	99.05	99.90	99.86	99.28	91.81	99.65	99.81

Table 1. Comparing raw bit accuracy ↑ (no error correction) of the proposed method (MultiNeRF) to baseline methods (NeRFProtector, WaterRF) for the single message task on datasets: SYN (upper) and LLFF datasets (lower). Values to 2 d.p.

Fig. 5 shows that quality is equivocal for the baselines versus MultiNeRF, which attains an average PSNR of 30.83 dB for SYN and 26.78 dB for the LLFF dataset. We are slightly higher (SSIM, PSNR) or lower (LPIPS) than our baselines on some metrics. Overall, we conclude that MultiNeRF is comparable in quality and slightly outperforms bit accuracy for the single watermarking task.

5.3. Evaluating multiple watermark embedding

We evaluate embedding multiple distinct watermarks into a single NeRF model. Since all baselines target a single watermark per model, we have adapted WateRF to accept an input variable to select a watermark and trained it as a baseline (WateRF-modified). Each watermark is 16 bits, and we embed several into a single model.

Fig. 7 shows average bit accuracy on SYN and LLFF across varying numbers of watermarks. For SYN, WateRF-modified returns a random response (50% bit accuracy) for

LPIPS (Lower is better)

Method (on SYN)	Avg.	Chair	Drums	Ficus	Hotdog	Lego	Materials	Mic	Ship
WateRF	0.04	0.02	0.05	0.02	0.03	0.02	0.04	0.02	0.08
NeRFProtector	0.08	0.04	0.07	-	0.08	0.03	0.08	0.05	0.19
MultiNeRF (ours)	0.04	0.02	0.05	0.02	0.04	0.02	0.04	0.02	0.08
MultiNeRF-Noised (ours)	0.04	0.02	0.06	0.03	0.04	0.02	0.04	0.03	0.09
Method (on LLFF)	Avg.	Fern	Flower	Fortress	Horns	Leaves	Orchids	Room	Trex
WateRF	0.10	0.13	0.09	0.07	0.08	0.12	0.17	0.06	0.06
NeRFProtector	0.07	0.10	-	0.07	0.15	-	0.08	0.05	0.06
MultiNeRF	0.09	0.14	0.09	0.06	0.08	0.12	0.18	0.05	0.07
MultiNeRF-Noised (ours)	0.10	0.14	0.09	0.07	0.08	0.12	0.17	0.08	0.06

PSNR (Higher is better)

Method (on SYN)	Avg.	Chair	Drums	Ficus	Hotdog	Lego	Materials	Mic	Ship
WateRF	30.58	32.33	25.30	31.04	33.61	32.74	28.45	31.94	29.30
NeRFProtector	28.44	31.40	24.41	-	32.51	31.00	26.20	30.61	23.00
MultiNeRF (ours)	30.83	32.81	25.58	31.22	34.12	32.83	28.15	32.05	29.87
MultiNeRF-Noised (ours)	30.61	32.59	25.38	30.64	34.07	33.14	28.16	31.35	29.53
Method (on LLFF)	Avg.	Fern	Flower	Fortress	Horns	Leaves	Orchids	Room	Trex
WateRF	26.31	25.08	27.60	30.26	27.88	20.96	19.98	31.30	27.43
NeRFProtector	26.08	27.58	-	28.13	23.05	-	20.19	30.73	26.83
MultiNeRF (ours)	26.78	24.99	27.53	30.20	28.15	20.97	19.99	34.68	27.70
MultiNeRF-Noised (ours)	26.36	24.89	27.45	30.45	28.05	20.97	19.96	31.48	27.65

SSIM (Higher is better)

Method (on SYN)	Avg.	Chair	Drums	Ficus	Hotdog	Lego	Materials	Mic	Ship
WateRF	0.94	0.96	0.93	0.97	0.96	0.96	0.93	0.97	0.87
NeRFProtector	0.91	0.95	0.90	-	0.94	0.95	0.90	0.96	0.78
MultiNeRF (ours)	0.95	0.97	0.93	0.97	0.96	0.96	0.93	0.97	0.88
MultiNeRF-Noised (ours)	0.94	0.97	0.93	0.96	0.96	0.97	0.93	0.97	0.88
Method (on LLFF)	Avg.	Fern	Flower	Fortress	Horns	Leaves	Orchids	Room	Trex
WateRF	0.82	0.80	0.83	0.87	0.87	0.73	0.64	0.93	0.90
NeRFProtector	0.85	0.87	-	0.87	0.77	-	0.74	0.91	0.89
MultiNeRF (ours)	0.82	0.80	0.84	0.88	0.88	0.73	0.64	0.94	0.91
MultiNeRF-Noised (ours)	0.83	0.80	0.84	0.88	0.89	0.73	0.64	0.94	0.91

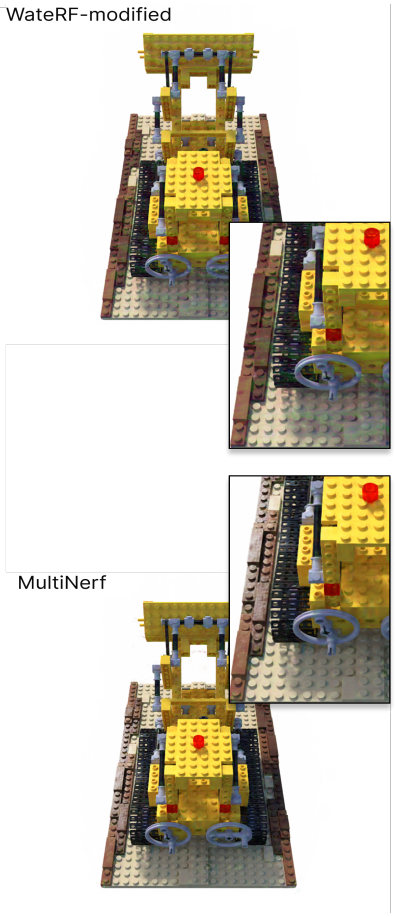

Figure 5. Comparing visual quality. Left: Table of metrics (LPIPS ↓, PSNR ↑, SSIM ↑) of MultiNeRF to baselines (NeRFProtector, WateRF) for single message task on SYN and LLFF datasets. Values to 2 d.p. b) Right: Examples of visual artifacts (colored ripples) present in WateRF-modified versus the proposed MultiNeRF method (zoom inset).

32 watermarks and onwards whereas MultiNeRF achieves 70% at 32 watermarks, degrading to random at 64. On LLFF, both models perform similarly at 2 watermarks, with WateRF-modified dropping to random response at 32 watermarks, whilst MultiNeRF achieves 82% bit accuracy at 32 watermarks and 68% at 64 watermarks. Error bars confirm MultiNeRF significantly outperforms WateRF-modified for the multiple watermarking task (*i.e.* $p < 0.0001$ for both SYN and LLFF).

Fig. 6 shows similar visual quality across methods, with no statistically significant differences. Tab. 3b shows the performance–size trade-off for 16 watermarks. Compared to TensoRF [7], introducing a watermark grid increases parameter count and storage overhead by $\sim 12\%$.

5.4. Evaluating watermark robustness

We evaluate robustness on the SYN/Lego scene, averaging results over 50 watermarks. We apply standard image transformations (cropping, blur, JPEG compression) and regeneration attacks proposed from [43]. Fig. 9 shows bit accuracy under each attack type. The MultiNeRF-Noised model has greater bit accuracy on these attacks vs. baselines and a 3% bit lower accuracy than MultiNeRF.

	No Attack	Diffusion [43]	VAE-Cheng [8]	VAE-Bmshj [3]
WateRF	89.98	64.68	56.78	56.96
NeRFProtector	78.59	51.56	63.35	63.00
MN	91.51	67.53	56.26	56.48
MN-noised	**92.52**	**80.48**	**73.94**	**74.72**

Table 2. Comparison of bit accuracies (persistence) under various regeneration attacks.

Tab. 2 compares the bit accuracy of MultiNeRF and baselines under three regeneration attacks. We evaluate one diffusion-based attack (**Diffusion** [43] and two VAE-based attacks (**VAE-Cheng** [8], and **VAE-Bmshj** [3]). MN-noised maintains higher robustness across regeneration attacks. These results indicate that while other methods experience bit accuracy degradation under the image transformation and regeneration attacks, introducing the Differentiable

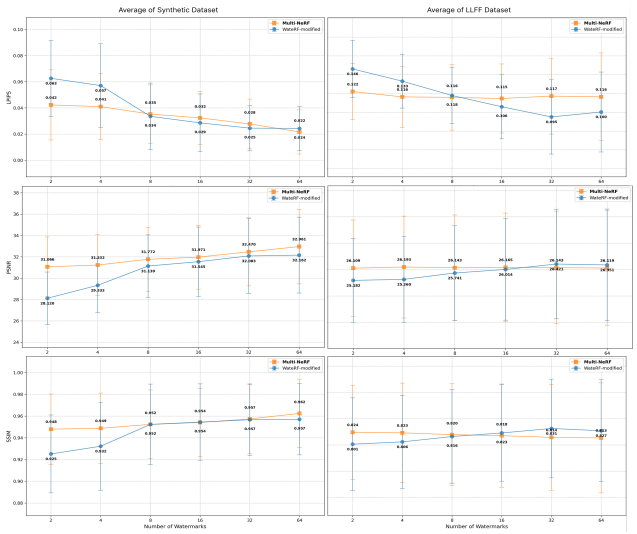

Figure 6. Evaluating the visual quality of MultiNeRF vs. baseline WateRF-modified for the multi-watermarking task: LPIPS (top); PSNR (mid.); SSIM (bot.).

Augmentation Layer (in MN-noised) increases the overall robustness.

5.5. Ablation Study

Watermark Grid. Here, we present ablation experiments to validate the design choices in MultiNeRF (Tab. 3a) conducted on the Lego scene with 16 watermarks per model. First, we remove the proposed watermark grid (-GRID), which falls back to an MLP to encode the watermark information. Without a separate spatial grid, bit accuracy drops to 51.60%, suggesting that an explicit grid provides higher capacity for multiple watermarks. We also examine the effect of removing modulation (-FiLM), causing a drop from 82.79 to 70 %. We explored a training simplification where the watermark grid is learned without adjusting the appearance or geometry grid (G_σ frozen), showing that bit accuracy suffers when we do not allow the appearance parameters to adapt. Another variation injects the FiLM-modulated features into earlier layers (+EARLY) of MLP rather than the last layer. While the bit accuracy (78.04%) remains high, visual quality suffers *e.g.* as with WateRF-modified, scene reflections are absent. This added capacity comes at a minor runtime cost, with MultiNeRF reducing FPS by $\sim 19\%$ compared to TensoRF.

Number of Tensor Components. The number of tensor components for watermark grid (#WMCom) refers to the rank of the factorized tensor as described in [7]. We fix 16 and 48 components for the density and appearance grids while varying #WMCom. From Tab. 4, we observe an upward trend in accuracy as we increase #WMCom, reaching a peak at 48 components. However, this comes at the cost of decreasing quality (PSNR and LPIPS). Both 8 and 16 have

(a) Ablation Study

Method	Bit Acc. ↑	PSNR ↑	LPIPS ↓	SSIM ↑
MN(-GRID)(+MLP)	51.60	35.50	0.01	0.98
MN(-FiLM)(-GRID)	66.28	33.57	0.02	0.97
MN(-FiLM)	70.00	34.69	0.01	0.97
MN G_σ frozen	67.72	34.56	0.01	0.97
MN(+EARLY)	78.04	34.29	0.02	0.97
MN(-GRID)	66.35	33.79	0.02	0.97
MN (ours)	**82.79**	34.03	0.01	0.97

(b) Performance vs. Size

Method	Bit Acc. ↑	Size (MB) ↓	Params (M)
TensoRF	-	68.6	17.75
MN(-GRID)(-FiLM)	66.28	68.8	17.76
MN (ours)	**82.79**	77.1	19.98

Table 3. Top: Ablation Study of MultiNeRF (abbrev. MN), evaluating the impact of key components. Bottom: Comparison of performance gain (accuracy) versus size increase. Figures to 2.d.p.

#WMCom	Bit Acc. ↑	PSNR ↑	LPIPS ↓	SSIM ↑	Param-WMGrid(M)
2	78.91	34.00	0.01	0.97	0.5
4	79.77	34.08	0.01	0.97	1.1
8	82.79	34.03	0.01	0.97	2.2
16	82.83	34.00	0.01	0.97	4.4
24	85.66	33.92	0.02	0.97	6.6
32	85.40	33.89	0.02	0.97	8.8
48	86.03	33.85	0.02	0.97	13.2

Table 4. Ablation Study on the Number of Watermark Components.

comparable bit accuracy $\sim 82.8\%$, and as 16 components offers only marginal accuracy gain over 8 whilst doubling model parameter count, we selected 8 components as the optimal trade-off between robustness and quality.

Scalability across different bit-length decoder. We compare different bit-length decoders trained on 16 watermarks per scene (Tab. 5). Both quality and accuracy somewhat invariant to different bit-length decoders for this task, showing the flexibility of our method, which adapts seamlessly to varying bit payload size.

User Study on Watermark Artifacts. To evaluate the perceptual quality of our method, we conducted a user study where participants rated watermark artifacts on a scale from 1 (low artifacts) to 5 (high artifacts). Tab. 6 shows the mean artifact scores, standard deviations and statistical significance (p-values) from t-tests comparing each method to MultiNeRF. Our method achieves one of the lowest artifact scores (3.34), comparable to WateRF [17] (3.46, $p > 0.05$), indicating no statistically significant difference. However, compared to WateRF-modified (3.65, $p < 0.05$) and NeRF-Protector [31] (3.61, $p < 0.05$), our method shows a statistically significant reduction in artifacts, confirming that it introduces fewer perceptible watermark distortions than the other baselines. This demonstrates that MultiNeRF effectively balances robustness with perceptual degradation.

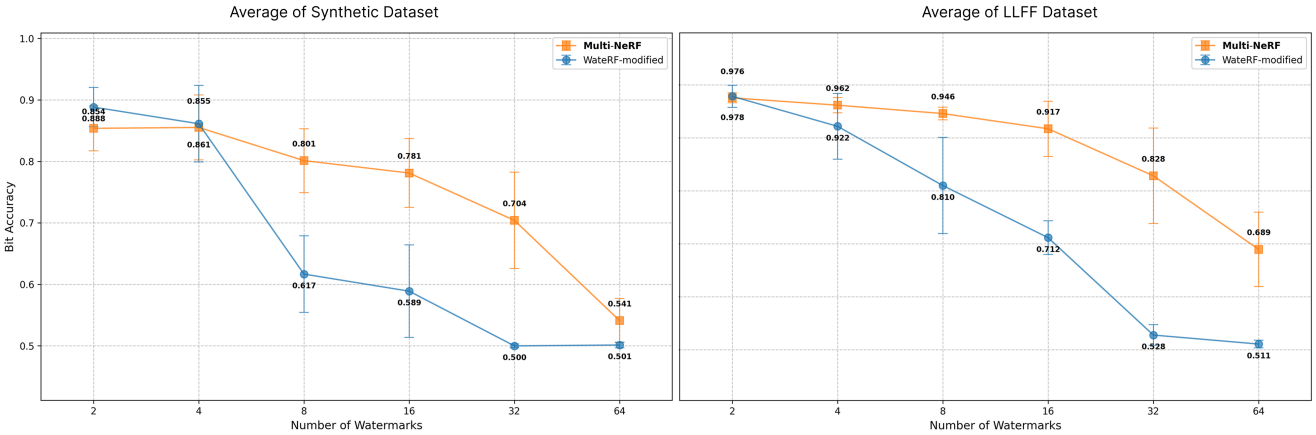

Figure 7. Evaluating bit accuracy ↑ of MultiNeRF versus baseline WateRF-modified for the multi-watermarking task. Accuracies averaged for SYN (left) and LLFF (right) datasets. Performance is significantly higher for MultiNeRF beyond the single watermarking case.

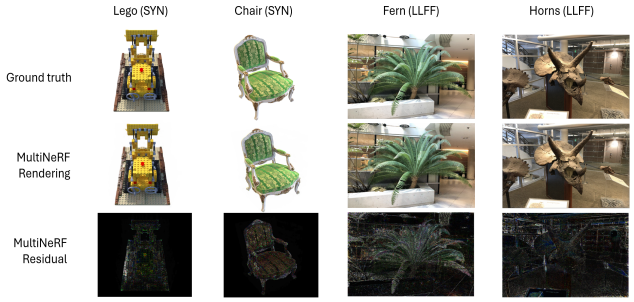

Figure 8. Visualizing the imperceptible watermark residual (bottom, amplified) introduced by MultiNeRF between the ground truth (top) and watermarked image (middle).

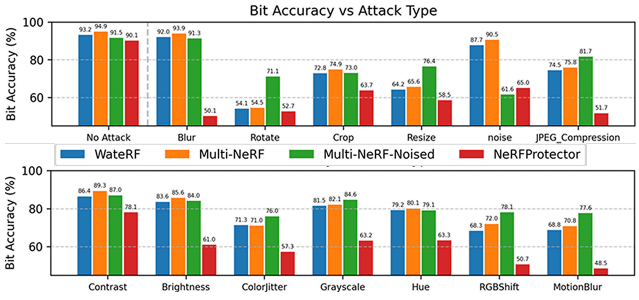

Figure 9. Evaluating the robustness to various degrading transformations for the proposed MultiNeRF method trained with (MN-noised) and without noise augmentations to baselines WateRF and NeRFProtector for watermarking a single message (averaged for 50 runs over Lego).

	Decoder Bit Length							
	Synthetic				LLFF			
	8	16	32	48	8	16	32	48
Bit Acc.	81.7	82.00	83.50	84.50	94.14	91.10	91.71	91.87
PSNR	34.51	34.73	33.79	33.96	27.11	27.63	28.54	27.57
LPIPS	0.02	0.02	0.03	0.03	0.14	0.11	0.11	0.11
SSIM	0.97	0.97	0.96	0.95	0.83	0.85	0.84	0.84

Table 5. Comparison of different decoder bit length metrics for SYN and LLFF datasets.

Method	Artifacts Score (Mean) ↓	Standard Deviation	p-value
WateRF-modified	3.65	1.16	$p < 0.05$
NerfProc	3.61	1.29	$p < 0.05$
WateRF	3.46	1.28	$p > 0.05$
MultiNeRF	**3.34**	**1.35**	-

Table 6. User study results.

6. Conclusion

We presented **MultiNeRF**, the first NeRF watermarking technique to embed multiple conditional watermarks simultaneously within a NeRF (TensoRF) model. MultiNeRF performs competitively on single-message embedding and uniquely supports multiple watermarks. This increases model capacity because the watermark grid enables multiple messages to be simultaneously encoded without degrading the visual quality. FiLM-based modulation enables selection between the distinct watermarks at rendering time. Future work could explore learnable perceptual metrics to help better quantify artifacts introduced by watermark embedding. As NeRFs continue to evolve and find new applications, frameworks like MultiNeRF will play an essential role in securing the intellectual property rights of 3D content creators. Their integration with emerging media provenance standards presents another future direction.

References

[1] Ramesh Aditya, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical Text-Conditional Image Generation with CLIP Latents. *arXiv preprint arXiv:2204.06125*, 2022. 2

[2] Vishal Asnani, John Collomosse, Tu Bui, Xiaoming Liu, and Shruti Agarwal. ProMark: Proactive Diffusion Watermarking for Causal Attribution. In *CVPR*, 2024. 2

[3] Johannes Ballé, David Minnen, Saurabh Singh, Sung Jin Hwang, and Nick Johnston. Variational image compression with a scale hyperprior. *arXiv preprint arXiv:1802.01436*, 2018. 6

[4] Shumeet Baluja. Hiding Images in Plain Sight: Deep Steganography. *NeurIPS*, 30, 2017. 2

[5] Tu Bui, Shruti Agarwal, and John Collomosse. TrustMark: Universal Watermarking for Arbitrary Resolution Images. *arXiv preprint arXiv:2311.18297*, 2023. 1, 2

[6] Tu Bui, Shruti Agarwal, Ning Yu, and John Collomosse. RoSteALS: Robust Steganography using Autoencoder Latent Space. In *Proc. CVPR WMF*, pages 933–942, 2023. 2

[7] Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. TensoRF: Tensorial Radiance Fields. In *European Conference on Computer Vision*, pages 333–350. Springer, 2022. 2, 3, 5, 6, 7

[8] Zhengxue Cheng, Heming Sun, Masaru Takeuchi, and Jiro Katto. Learned Image Compression with Discretized Gaussian Mixture Likelihoods and Attention Modules. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7939–7948, 2020. 6

[9] John Collomosse and Andy Parsons. To Authenticity, and Beyond! Building safe and fair generative AI upon the three pillars of provenance. *IEEE Computer Graphics and Applications*, 44(3):82–90, 2024. 1, 2

[10] Steffen Czolbe, Oswin Krause, Ingemar Cox, and Christian Igel. A loss function for generative neural networks based on watson's perceptual model. *Advances in Neural Information Processing Systems*, 33:2051–2061, 2020. 5

[11] Lidia Fabra, J Ernesto Solanes, Adolfo Muñoz, Ana Martí-Testón, Alba Alabau, and Luis Gracia. Application of Neural Radiance Fields (NeRFs) for 3D Model Representation in the Industrial Metaverse. *Applied Sciences*, 14(5):1825, 2024. 1

[12] Pierre Fernandez, Alexandre Sablayrolles, Teddy Furon, Hervé Jégou, and Matthijs Douze. Watermarking Images in Self-Supervised Latent Spaces. In *Proc. ICASSP*, pages 3054–3058. IEEE, 2022. 1, 2

[13] Pierre Fernandez, Guillaume Couairon, Hervé Jégou, Matthijs Douze, and Teddy Furon. The Stable Signature: Rooting Watermarks in Latent Diffusion Models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 22466–22477, 2023. 1, 2

[14] Pierre Fernandez, Hady Elsahar, I Zeki Yalniz, and Alexandre Mourachko. Video Seal: Open and Efficient Video Watermarking. *arXiv preprint arXiv:2412.09492*, 2024. 1

[15] Kazem Ghazanfari, Shahrokh Ghaemmaghami, and Saeed R Khosravi. LSB++: An Improvement to LSB+ Steganography. In *TENCON 2011-2011 IEEE Region 10 Conference*, pages 364–368. IEEE, 2011. 2

[16] Jonathan Ho, Chitwan Saharia, William Chan, David J Fleet, Mohammad Norouzi, and Tim Salimans. Imagen Video: High Definition Video Generation with Diffusion Models. In *NeurIPS*, 2022. 2

[17] Youngdong Jang, Dong In Lee, MinHyuk Jang, Jong Wook Kim, Feng Yang, and Sangpil Kim. WateRF: Robust Watermarks in Radiance Fields for Protection of Copyrights. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12087–12097, 2024. 1, 2, 3, 4, 5, 7

[18] Diederik P Kingma. Adam: A Method for Stochastic Optimization. *arXiv preprint arXiv:1412.6980*, 2014. 5

[19] Chaojian Li, Sixu Li, Yang Zhao, Wenbo Zhu, and Yingyan Lin. RT-NeRF: Real-Time On-Device Neural Radiance Fields Towards Immersive AR/VR Rendering. In *Proceedings of the 41st IEEE/ACM International Conference on Computer-Aided Design*, pages 1–9, 2022. 1

[20] Chenxin Li, Brandon Y Feng, Zhiwen Fan, Panwang Pan, and Zhangyang Wang. StegaNeRF: Embedding Invisible Information within Neural Radiance Fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 441–453, 2023. 2

[21] Chenxin Li, Hengyu Liu, Zhiwen Fan, Wuyang Li, Yifan Liu, Panwang Pan, and Yixuan Yuan. InstantSplamp: Fast and Generalizable Stenography Framework for Generative Gaussian Splatting. In *The Thirteenth International Conference on Learning Representations*, 2025. 2

[22] Xiaoxia Li and Jianjun Wang. A steganographic method based upon JPEG and particle swarm optimization algorithm. *Information Sciences*, 177(15):3099–3109, 2007. 2

[23] Ziyuan Luo, Qing Guo, Ka Chun Cheung, Simon See, and Renjie Wan. CopyRNeRF: Protecting the CopyRight of Neural Radiance Fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 22401–22411, 2023. 1, 2

[24] Ben Mildenhall, Pratul P Srinivasan, Rodrigo Ortiz-Cayon, Nima Khademi Kalantari, Ravi Ramamoorthi, Ren Ng, and Abhishek Kar. Local Light Field Fusion: Practical View Synthesis with Prescriptive Sampling Guidelines. *ACM Transactions on Graphics (ToG)*, 38(4):1–14, 2019. 2, 5

[25] 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. 1, 2, 5

[26] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant Neural Graphics Primitives with a Multiresolution Hash Encoding. *ACM Transactions on Graphics (TOG)*, 41(4):1–15, 2022. 5

[27] K. A. Navas, Mathews Cheriyan Ajay, M. Lekshmi, Tampy Archana, and M. Sasikumar. DWT-DCT-SVD based watermarking. In *COMSWARE'08*, pages 271–274. IEEE, 2008. 2

[28] Ethan Perez, Florian Strub, Harm De Vries, Vincent Dumoulin, and Aaron Courville. FiLM: Visual Reasoning with a General Conditioning Layer. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2018. 2, 3

[29] Tomáš Pevnỳ, Tomáš Filler, and Patrick Bas. Using High-Dimensional Image Models to Perform Highly Undetectable

Steganography. In *Proc. Int. Conf. Information Hiding*, pages 161–177. Springer, 2010. 2

[30] Alexandre Sablayrolles, Matthijs Douze, Cordelia Schmid, and Hervé Jégou. Radioactive Data: Tracing Through Training. *International Conference on Machine Learning (ICML)*, 2020. 2

[31] Qi Song, Ziyuan Luo, Ka Chun Cheung, Simon See, and Renjie Wan. Protecting NeRFs' Copyright via Plug-And-Play Watermarking Base Model. In *European Conference on Computer Vision*, pages 57–73. Springer, 2024. 1, 2, 3, 5, 7

[32] Mustafa Sabah Taha, Mohd Shafry Mohd Rahem, Mohammed Mahdi Hashim, and Hiyam N Khalid. High payload image steganography scheme with minimum distortion based on distinction grade value method. *Multimedia Tools and Applications*, 81(18):25913–25946, 2022. 2

[33] Matthew Tancik, Ben Mildenhall, and Ren Ng. StegaStamp: Invisible Hyperlinks in Physical Photographs. In *Proc. CVPR*, pages 2117–2126, 2020. 1, 2

[34] Shan Wang, Ting Liu, and Yizhong Wang. FakeTagger: Automated Fake News Detection Using Proactive Watermarks. In *NeurIPS Workshop on AI for Social Good*, 2021. 2

[35] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image Quality Assessment: From Error Visibility to Structural Similarity. *IEEE transactions on Image Processing*, 13(4):600–612, 2004. 5

[36] Yuxin Wen, John Kirchenbauer, Jonas Geiping, and Tom Goldstein. Tree-Ring Watermarks: Invisible Fingerprints for Diffusion Images. In *Advances in Neural Information Processing Systems*, 2023. 2

[37] Rui Xu, Deren Lei, Yaxi Li, David Lowe, Alex Gorevski, Mingyu Wang, Emily Ching, Alex Deng, et al. InvisMark: Invisible and Robust Watermarking for AI-generated Image Provenance. *arXiv preprint arXiv:2411.07795*, 2024. 2

[38] Kai Zhang, Nick Kolkin, Sai Bi, Fujun Luan, Zexiang Xu, Eli Shechtman, and Noah Snavely. ARF: Artistic Radiance Fields. In *European Conference on Computer Vision*, pages 717–733. Springer, 2022. 5

[39] Ruier Zhang. Moving museums into the metaverse. *Science and Technology of Engineering, Chemistry and Environmental Protection*, 1(5), 2024. 1

[40] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The Unreasonable Effectiveness of Deep Features as a Perceptual Metric. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 586–595, 2018. 5

[41] Xuanyu Zhang, Jiarui Meng, Runyi Li, Zhipei Xu, Yongbing Zhang, and Jian Zhang. GS-Hider: Hiding Messages into 3D Gaussian Splatting. *arXiv preprint arXiv:2405.15118*, 2024. 1, 2

[42] Yifan Zhang, Fangchang Wu, Wei Yan, Yun Fu, Si Wu, Xiaojun Wang, and Xiaolei Zhang. DeepMarks: A Secure Fingerprinting Framework for Digital Content Protection. *IEEE Transactions on Information Forensics and Security*, 16:2700–2715, 2021. 2

[43] Xuandong Zhao, Kexun Zhang, Zihao Su, Saastha Vasan, Ilya Grishchenko, Christopher Kruegel, Giovanni Vigna, Yu-Xiang Wang, and Lei Li. Invisible Image Watermarks Are Provably Removable Using Generative AI. In *Advances in Neural Information Processing Systems*, 2024. 6

[44] Jiren Zhu, Russell Kaplan, Justin Johnson, and Li Fei-Fei. HiDDeN: Hiding Data with Deep Networks. In *Proc. ECCV*, pages 657–672, 2018. 1, 2, 4