Protecting NeRFs' Copyright via Plug-And-Play Watermarking Base Model

Qi Song[1,2], Ziyuan Luo[1,2], Ka Chun Cheung[2], Simon See[2] and Renjie Wan[1]⋆

[1] Department of Computer Science, Hong Kong Baptist University
[2] NVIDIA AI Technology Center, NVIDIA
{qisong,ziyuanluo}@life.hkbu.edu.hk {chcheung, ssee}@nvidia.com
renjiewan@hkbu.edu.hk

Abstract. Neural Radiance Fields (NeRFs) have become a key method for 3D scene representation. With the rising prominence and influence of NeRF, safeguarding its intellectual property has become increasingly important. In this paper, we propose **NeRFProtector**, which adopts a plug-and-play strategy to protect NeRF's copyright during its creation. NeRFProtector utilizes a pre-trained watermarking base model, enabling NeRF creators to embed binary messages directly while creating their NeRF. Our plug-and-play property ensures NeRF creators can flexibly choose NeRF variants without excessive modifications. Leveraging our newly designed progressive distillation, we demonstrate performance on par with several leading-edge neural rendering methods. Our project is available at: `https://qsong2001.github.io/NeRFProtector`.

Keywords: NeRF · Copyright protection · Plug-and-play

1 Introduction

Neural Radiance Fields (NeRFs) [4,12,29,31] mark a new way for 3D scene representation. As individuals might use radiance fields to represent their chosen 3D scenes and share these representations with the public, it becomes increasingly critical to establish a convenient copyright framework tailored to these novel representations during their creations. Then, once created, the copyright of these scene representations can be easily claimed in the event of an ownership breach.

While CopyRNeRF [25] is developed to uphold copyright protection by embedding binary watermarks into NeRF models and then extracting watermarks from rendered images, its effectiveness is lessened by its weaknesses. **First**, a delay exists between creating the NeRF model and inserting ownership messages because binary messages are embedded into the NeRF model post-creation through model fine-tuning. Should malicious users acquire NeRF models in this delay, they could exploit them for nefarious intentions. **Second**, NeRF creators are required to jointly train a message extractor during message embedding,

⋆ Corresponding author.

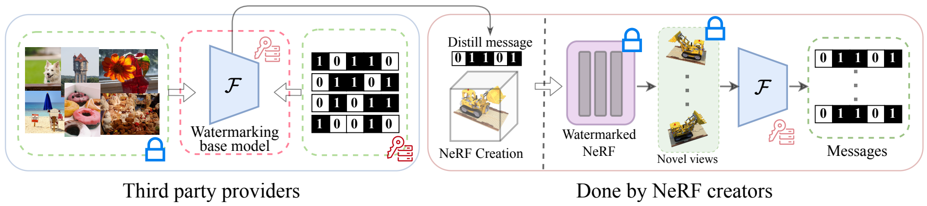

Fig. 1: Proposed scenario of our method. NeRF creators can obtain a pre-trained watermarking extractor from a third party (*e.g.*, open-source library) or train a message extractor separately via standard pipelines, as the **watermarking base model** $\mathcal{F}$. Once obtained, this base model is considered "**plug-and-play**" in our scenario. Throughout the NeRF creation process, creators can readily use this base model to embed watermarks in their NeRF. After the optimization of NeRF is complete, they obtain a watermarked NeRF. When this watermarked NeRF is distributed publicly, the creators can then use the same base model to retrieve binary messages from newly rendered views, thereby asserting their ownership.

which makes the entire watermarking process exceedingly time-intensive and complex. NeRF creators or owners may abandon the watermarking due to its excessive complexity and difficulty.

If NeRF creators are reluctant to use it, any fabulous designs for watermarking become meaningless. Therefore, in addition to the primary evaluation criteria for watermarking systems - robustness and invisibility of embedded binary messages [7] - we introduce NeRFProtector, a NeRF watermarking framework with a **plug-and-play** property. Such a plug-and-play property ensures that binary messages can be conveniently incorporated into existing NeRF variants with minimal modifications.

As shown in Fig. 1, we use a watermarking base model to achieve the above goals. The watermarking base model can be obtained from a third party (*e.g.*, some open-resource message extractors[3]) or trained separately via standard pipelines. NeRF creators simply need to select one watermarking base model and integrate it with their chosen NeRF variants (*e.g.*, Instant-NGP [31], TensoRF [4], and so on), and then message embedding and NeRF optimization can proceed concurrently. Once optimization is complete, NeRF owners can use the base model to retrieve binary messages from the rendered contents of NeRF. This approach spares NeRF creators from extra efforts for modifying NeRF structures. Crucially, message embedding occurs during the creation of NeRF, leaving no window of opportunity for malicious attackers. Based on our investigation, the message extractor from conventional image watermarking frameworks [14,54] can directly be our watermarking base model. These message extractors, through their training, have acquired knowledge of copyright messages and inherently possess the capability to retrieve such messages. Once NeRF creators acquire this watermarking base model, they can watermark their NeRF by distilling

[3] `https://github.com/ando-khachatryan/HiDDeN`

message knowledge from the extractor into their NeRF. Then, we only need to consider how to distill such established knowledge of binary messages into NeRF during creation.

Some methods like CLIP-NeRF [43] leverage knowledge from external CLIP model [37] to execute shape and color manipulation [5]. However, these methods necessitate modifications to NeRF's core structure for representation, which is inconsistent with our aim of ensuring watermarking compatibility with existing NeRF frameworks. To maintain simplicity, our proposed scenario opts for a simple Progressive Global Rendering (PGR) in place of NeRF's typical volume rendering. NeRF's volume rendering typically involves rendering a random subset of pixels, referred to as local rendering here, during optimization to manage computational demands. This limited pixel rendering is insufficient for embedding messages effectively within NeRFs. Rather than only rendering a small subset of pixels each time, our progressive global rendering renders all pixels across various image resolutions. This approach allows for more effective message embedding within NeRF's representation.

The framework of NeRFProtector is shown in Fig. 2. In our designs, we do not change the fundamental representation of NeRF. NeRF creators only need to use a pre-trained message extractor as the base model to finish both message embedding and extraction. Despite the plug-and-play property of this base model, our framework can also satisfy the key criteria for robustness and invisibility. Our key contribution can be concluded as follows:

– A plug-and-play mode that NeRF creators easily watermark their NeRFs during the creation process, leaving no delays between NeRF creation and ownership message embedding.
– The utilization of a watermarking base model for efficient message embedding and extraction.
– Progressive global rendering is proposed to effectively distill message knowledge from the watermarking base model into NeRFs by exploring relations between rendering strategies and watermark embedding.

The plug-and-play property of NeRFProtector ensures that minimal modifications are required to the architecture of NeRF representation, and the whole watermarking process can be conducted conveniently. Extensive experiments are conducted to verify the performance of our method, and the potential threats are also analyzed.

2 Related work

Digital watermarking for 2D. Digital watermarking serves as a method for embedding copyright messages, referred to as watermarks, into various digital assets. These assets include images [18, 40, 45, 54], videos [2, 8, 24], and certain generative models [11,50]. Researchers also investigate techniques that aim to ensure the watermark's robustness against various attacks [1,9,34,39] and common image processing operations such as compression, cropping, and resizing [14, 54].

The embedded copyright can be used for various purposes, including copyright protection and intellectual property protection. Our work leverages the prior knowledge of 2D watermarking and achieves copyright information protection for NeRF.

Neural Radiance Fields. Neural Radiance Fields (NeRF) [29,31,42,52,53] is a method for 3D scene representation. It can represent a scene as a continuous 3D function that models the radiance at every point in space. In recent years, NeRF has sparked a lot of research interest and has been widely adopted in the computer vision and graphics communities. NeRF has been extended to incorporate additional functionalities, such as text-to-3D [19,27,33], image-to-3D [22,41,48], scene editing [23, 49], and 3D model copyright protection [16, 25] tasks. These extensions further showcase the versatility and potential applications of NeRF in various domains. As the popularity and impact of NeRF continue to grow, the need for protecting its intellectual property becomes crucial.

Digital watermarking for 3D. Traditional 3D watermarking approaches focus on watermarking on polygonal meshes [32, 34, 38] and point clouds [13, 21] with explicit structures. A deep-learning-based approach [47] embeds binary messages in 3D meshes and extracts them from 2D rendered images. For implicit 3D models such as NeRF, SteagaNeRF [16] introduces a technique for embedding data within NeRF. Specifically, CopyRNeRF [25] suggests safeguarding NeRF's copyright by embedding binary messages into NeRF and subsequently extracting watermarks from the images rendered. CopyRNeRF [25] also investigates several settings in the ownership claim of NeRF representation. However, CopyRNeRF [25] is limited to embedding binary ownership messages through model fine-tuning post-creation of NeRF models, a process that potentially leaves room for exploitation by malicious individuals. Furthermore, its complex usage might discourage NeRF creators from adopting it. In contrast, our NeRFProtector enables the watermarking of NeRF models right at the point of creation, utilizing an easy-to-use plug-and-play watermarking base model.

3 Problem statement

NeRF. NeRF [29] takes in a 3D location $\mathbf{x} \in \mathbb{R}^3$ and viewing direction $\mathbf{d} \in \mathbb{R}^2$, then outputs a color value $\mathbf{c} \in \mathbb{R}^3$ and a density value $\sigma \in \mathbb{R}^+$. To produce a pixel's color in a 2D image, samples of these values can be composited along a ray according to volume rendering. Formally, N_p points are sampled along a camera ray r with color and density values $\{(\mathbf{c}_r^i, \sigma_r^i)\}_{i=1}^N$. The corresponding RGB color value $\hat{\mathbf{C}}(r)$ is obtained using alpha composition as

$$\hat{\mathbf{C}}(r) = \sum_{i=1}^{N_p} T_r^i (1 - \exp\left(-\sigma_r^i \delta_r^i\right)) \mathbf{c}_r^i, \tag{1}$$

where $T_r^i = \prod_{j=1}^{i-1} \left(\exp\left(-\sigma_r^i \delta_r^i\right)\right)$, and δ_r^i is the distance between adjacent sample points. The weights of NeRF can be optimized by minimizing a reconstruction

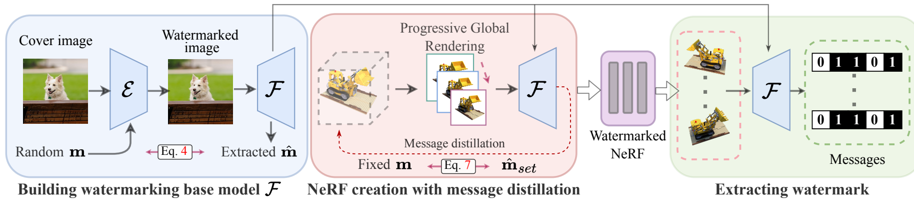

Fig. 2: Our **plug-and-play** method to watermark NeRF during its creation. **(1) Building watermarking base model**: The watermarking base model $\mathcal{F}$ can be sourced from a third party (Sec. 4.1). We implement a HiDDeN [54] framework to get the pre-trained message extractor as our watermarking base model $\mathcal{F}$. During the training, the encoder $\mathcal{E}$ encodes a randomly selected 48-kit message $\mathbf{m}$ and cover image $\mathbf{I}_o$ and outputs a watermarked image $\mathbf{I}_{en}$. Then the message extractor $\mathcal{F}$ extracts embedded message $\hat{\mathbf{m}}$ from the watermarked image. Message encoder $\mathcal{E}$ is discarded after building watermarking base model $\mathcal{F}$. **(2) NeRF creation with message distillation**: NeRF creators first fix a copyright message $\mathbf{m}$, then employ this base model $\mathcal{F}$ to embed selected watermarks to NeRFs during the creation process via Progressive Global Rendering (PGR) and message distillation (Sec. 4.2). When the optimization of NeRF is finalized, creators immediately obtain a watermarked NeRF. **(3) Extracting watermark**: Subsequently, they can utilize the base model $\mathcal{F}$ to retrieve binary watermarks from the rendered images, asserting their ownership.

loss between observations $\mathbf{C}$ and predictions $\hat{\mathbf{C}}$ as

$$\mathcal{L}_{\text{content}} = \frac{1}{N_r} \sum_{m=1}^{N_r} \|\hat{\mathbf{C}}(r_m) - \mathbf{C}(r_m)\|_2^2, \tag{2}$$

where N_r is the number of sampling pixels.

Our scenario. From the scenario shown in Fig. 1, NeRF creators can acquire the off-the-shelf watermarking base models from a third party or train a base model separately via standard pipelines. Then, they can directly combine the base model with their NeRF optimization. Once the optimization has concluded, the message embedding also concludes. Then, the base model can be further used to extract binary messages for ownership claims.

Our approach diverges from conventional watermarking techniques [25] for NeRF models. Rather than embedding the ownership messages after the creation of NeRF, we ensure NeRF models are protected right from their outset, substantially narrowing the window of opportunity for malicious actors. Additionally, this method permits NeRF creators to train their models with minimal deviation from standard procedures, avoiding the extra workload of training supplementary modules.

4 Proposed NeRFProtector

Our NeRFProtector aims at embedding k-bit binary messages $\mathbf{m} \in \{0, 1\}^k$ during the creation of NeRF. As our scenario relies on an important base model for watermarking, we first introduce it in Sec. 4.1. Then, we introduce how to distill knowledge from this base model into NeRF via a newly designed progressive distillation in Sec. 4.2.

4.1 Building watermarking base model

The essence of our approach is to switch out the current processes of message embedding and extraction with a watermarking base model. We envision this base model to possess a **plug-and-play** property, enabling NeRF creators to seamlessly incorporate it into their NeRFs with only minor modifications to established architecture. Additionally, this base model is designed to withstand common image-level distortions, ensuring robust message extraction even from distorted rendered images.

For simplicity, we choose the extractor from HiDDeN [54], a well-established deep-learning-based watermarking framework, to serve as our base model. As a message extractor for image watermarking, the base model naturally owns the capability to extract binary messages from watermarked images. Besides, we can easily make the extraction more robust by employing established studies in image watermarking [14, 54]. At last, the base model has learned the message pattern during its training, and we can then distill such learned knowledge into the representation of NeRF during its optimization.

Due to the requirement for plug-and-play, we refrain from making additional modifications to HiDDeN [54]. In general, HiDDeN [54] jointly optimizes the watermark encoder $\mathcal{E}$ and the watermarking base $\mathcal{F}$ to robustly embed k-bit binary messages into images. Specifically, during optimization, the encoder $\mathcal{E}$ takes a cover image $\mathbf{I}_o$ and a message $\mathbf{m} \in \{0,1\}^k$ as inputs and output a watermarked image $\mathbf{I}_{en}$ as follows,

$$\mathbf{I}_{en} = \mathcal{E}(\mathbf{I}_o, \mathbf{m}). \tag{3}$$

Binary message $\hat{\mathbf{m}}$ is then extracted from distorted images using a watermark extraction network $\mathcal{F}$ as $\hat{\mathbf{m}} = \mathcal{F}(T(\mathbf{I}_{en}))$. Then, the whole network is optimized by minimizing the Binary Cross Entropy (BCE) loss between the original binary message $\mathbf{m}$ and extracted binary message $\hat{\mathbf{m}}$ as:

$$\mathcal{L}_{\mathrm{m}} = \mathrm{BCE}(\mathbf{m}, \hat{\mathbf{m}}), \tag{4}$$

where $\mathrm{BCE}(\mathbf{m}, \hat{\mathbf{m}}) = -\left(\mathbf{m} \cdot \log(\tau(\hat{\mathbf{m}})) + (1-\mathbf{m}) \cdot \log(1-\tau(\hat{\mathbf{m}})\right)$ and τ is a sigmoid operation. After the training, we discard the encoder $\mathcal{E}$ in Eq. (3) and only use the extractor $\mathcal{F}$ as the watermarking base model. To improve the robustness, we follow HiDDeN [54] and employ a randomly selected transformation layer T during training to ensure that the hidden messages are robust to common image distortions.

4.2 Distilling watermarks into NeRF progressively

The base model has encapsulated all necessary knowledge about watermarking. We further distill the knowledge patterns into NeRF as they are being created. Then, the binary messages can be robustly extracted from rendered images regardless of the viewpoints. Although various approaches exist for distilling external knowledge into the representations of NeRF [10, 44], they necessitate changes to the essential structures of NeRF, conflicting with our need for **a plug-and-play property**. We achieve the watermark distillation by only making alternations to the rendering scheme of NeRF, the key obstacles for our distillation.

As displayed in Eq. (2), NeRF hinges on a random and small subset of pixels (N_r pixels in Eq. (2)) for rendering. Despite its efficiency in preventing the computational demand from being too high, the message patterns can only be embedded into random positions during distillation if only such local rendering is used. Then, as shown in Fig. 5, when the optimization settles down, the message pattern cannot effectively form meaningful patterns that can be identified and extracted by the base model. As shown in Fig. 2, we substitute such conventional local rendering in established NeRF with a progressive global rendering that is capable of rendering all pixels across various image resolutions. This modification does not alter the representation of NeRF, ensuring that our mechanism remains adaptable and can be easily applied to a range of NeRF variants [4, 12, 31].

Progressive Global Rendering (PGR). For our PGR, as displayed in Fig. 2, rather than only sampling a random subset of pixels, during rendering, we sample rays corresponding to all pixels across various scales to gain a comprehensive understanding of global information during optimization. By embedding messages on a global scale, they become deeply integrated into the scene's representation. This ensures the consistent properties of binary watermarks from various viewpoints, allowing message extraction irrespective of rendering viewpoints. Additionally, since 3D information often exhibits distinct characteristics in 2D projections at varying scales [35], progressive rendering accentuates these resolution-dependent properties, aiding in message distillation. Furthermore, the globally rendered images are all with reduced resolution. Then, our PGR can achieve comparable results without necessitating full-resolution images for optimization and make the computational cost acceptable.

As shown in Fig. 2, we consider rendering N_k progressive views $\hat{\mathbf{I}}_{set}$ at different scales denoted as follows:

$$\hat{\mathbf{I}}_{set} = \{\hat{\mathbf{I}}_n\}_{n=1}^{N_k} = \{\hat{\mathbf{I}}_1, \hat{\mathbf{I}}_2, ..., \hat{\mathbf{I}}_{N_k}\}, \tag{5}$$

where N_k and $\hat{\mathbf{I}}_n$ denote the number of layers and the cascade rendered view at different scales. Each $\hat{\mathbf{I}}_n \in \mathbb{R}^{\frac{W}{2^n} \times \frac{H}{2^n} \times 3}$ from progressive views set $\hat{\mathbf{I}}_{set}$ is obtained by rendering global rays of at corresponding resolution. In our setting, we empirically set $N_k = 3$, a value that can well balance efficiency and performance.

Distilling messages. We then utilize the base model $\mathcal{F}$ and progressive global rendering to distill message patterns into NeRF. For N_k rendered cascade views $\hat{\mathbf{I}}_{set}$ at different scales, we use the watermarking base model $\mathcal{F}$ to extract the embedded message $\hat{\mathbf{m}}_{set}$ as follows:

$$\hat{\mathbf{m}}_{set} = \mathcal{F}(\hat{\mathbf{I}}_{set}) = \{\hat{\mathbf{m}}_1, \hat{\mathbf{m}}_2, ..., \hat{\mathbf{m}}_{N_k}\}. \tag{6}$$

Then a distillation loss $\mathcal{L}_{\text{dis}}$ is computed by calculating the BCE loss between the predicted message set $\hat{\mathbf{m}}_{set}$ and the ground truth message $\mathbf{m}$:

$$\mathcal{L}_{\text{dis}} = \sum_{i=1}^{N_k} \alpha_i \cdot \text{BCE}(\mathbf{m}, \hat{\mathbf{m}}_i), \tag{7}$$

where N_k is the number of cascade layers, and α_i represents the weight of i layer's loss function. During the creation, minimizing this equation can distill messages into NeRF.

To make the distilled message pattern invisible, we enforce the high similarity between the rendering image and its corresponding ground truth as follows:

$$\mathcal{L}_{\text{inv}} = \|\hat{\mathbf{I}}_1 - \mathbf{I}_1\|_2^2, \tag{8}$$

where $\hat{\mathbf{I}}_1$ denotes the cascade rendered images with the highest resolution, and $\mathbf{I}_1$ is its corresponding ground truth. The ground truth is with a reduced resolution to be compatible with $\hat{\mathbf{I}}_1$.

Besides progressive global rendering, we still use local rendering for building radiance fields. During optimization, the local rendering randomly selects N_r pixels for rendering, which ensures the local pattern can be effectively considered during the message distillation. Thus, our whole loss functions for creating NeRF with a plug-and-play watermarking base model can be concluded as follows:

$$\mathcal{L} = \lambda_1 \mathcal{L}_{\text{local}} + \lambda_2 \mathcal{L}_{\text{inv}} + \lambda_3 \mathcal{L}_{\text{dis}}, \tag{9}$$

where λ_1, λ_2, and λ_3 are hyperparameters that balance the performance between embedding watermarking and reconstruction, and the $\mathcal{L}_{\text{local}}$ is same to Eq. (2). The network setup described above can be effectively adapted for embedding copyright messages into NeRF. PGR is only used during the optimization. Once NeRF is created, users can still render images via established ways [29].

4.3 Implementation details

For NeRF, we implement our method with Instant-NGP [31]. Instant-NGP takes a 3D grid as input and encodes it using a grid encoder. It then predicts density values σ using two-layer MLP with 64 channels. For the color branch, a three-layer MLP with 64 channels is used to predict color c and σ from view direction (θ, ϕ). As for the rendering, the number of cascade layers N_k is set to 3, and the rendered pixel count N_r for the local rendering is set to 4096, respectively. *For base model*, we implement our method with HiDDeN [54]. The watermark encoder $\mathcal{E}$ and extractor $\mathcal{F}$ in HiDDeN [54] are jointly trained on COCO [20] dataset. During its training process, encoder $\mathcal{E}$ encodes a randomly selected 48-kit message $\mathbf{m}$ and cover image $\mathbf{I}_o$ and predicts a watermarked image $\mathbf{I}_{en}$. Then extractor $\mathcal{F}$ extracts embedded messages from watermarked image $\mathbf{I}_{en}$. The whole network is optimized with Adam [15] optimizer in $100K$ iterations.

During NeRF creation, the NeRF creator chooses one preferable message $\mathbf{m}$ as copyright messages. We fix the weight of watermarking base model $\mathcal{F}$ and minimize the distance between the chosen message $\mathbf{m}$ and extracted message $\hat{\mathbf{m}}_{set}$. Hyperparameters in Eq. (9) are set as $\lambda_1 = 0.01$ and $\lambda_3 = 0.001$. We use the Adam [15] optimizer with setting the values of $\beta_1 = 0.9$, $\beta_2 = 0.999$, and a learning rate of 0.01. The learning rate decays following the exponential scheduler during optimization. We train our model for each scene on a single NVIDIA A100 GPU.

5 Experiments

5.1 Experiment settings

Dataset. We conduct experiments on the Blender [29] and LLFF [28] dataset. The Blender dataset comprises 8 synthetic object images captured from different viewpoints, while the LLFF dataset consists of 8 real-world scenes captured in a forward-facing manner. We choose these datasets based on previous research [29, 31] and use them to assess the bit accuracy and reconstruction performance of NeRF. For each scene, we render testing views and calculate their average values for our analysis.

Baselines. We compare four strategies to guarantee a fair comparison: 1) HiDDeN [54] + NeRF [31]: processing images with classical 2D watermarking method HiDDeN [54] before training the NeRF model; 2) MBRS [14] + NeRF [31]: processing images with a recently proposed 2D watermarking method MBRS [14] before training the NeRF model; 3) CopyRNeRF [25]: a state-of-the-art method for protecting the copyright of NeRF models via digital watermarking. To evaluate the impact of watermarks on reconstruction, we also compare the results with the non-watermarked version of NeRF; 4) NeRF [31] w/o watermark: we also compare with NeRF [31] w/o watermark to evaluate the reconstruction quality. **To further illustrate our plug-and-play properties, we implement our method with other NeRF variants [4, 12] and base model [9] as demonstrated in Sec. 5.2**.

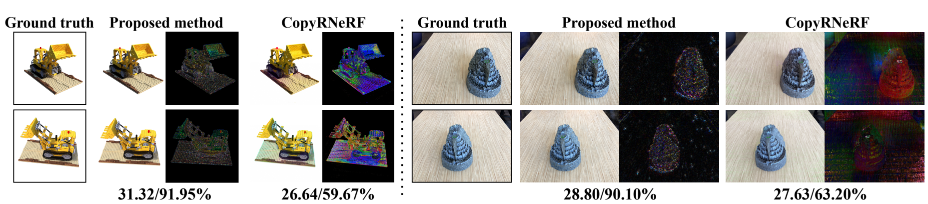

Fig. 3: Visual quality comparisons with CopyRNeRF [25]. The texts below the images show the results of PSNR and bit accuracy. We show the differences ×10 between the rendered views and the ground truth for both methods. Our NeRFProtector exhibits better consistency across multiple viewpoints, achieving a well-balanced trade-off between reconstruction quality and bit accuracy.

Table 1: Quantitative results of visual representation and watermarking extraction performance. We report the results of watermarking robustness when rendered images meet common distortions (*e.g.*, Crop, Resize, and JPEG compression). "None" indicates that no distortion has been applied. "N/A" denotes Not Applicable as NeRF w/o watermark does not involve message embedding.

Dataset	Method	PSNR ↑	SSIM ↑	LPIPS ↓	Bit accuracy %			
					None	Crop	Resize	JPEG
Blender-test	NeRF w/o watermark	30.62	0.9579	0.0343	N/A			
	HiDDeN [54]+NeRF	28.30	0.9219	0.0495	50.29	50.03	52.32	50.84
	MBRS [14]+NeRF	24.17	0.8461	0.2967	50.53	49.64	51.72	49.81
	CopyRNeRF [25]	25.50	0.9073	0.0885	62.15	56.63	57.32	58.41
	NeRFProtector	**29.26**	**0.9393**	**0.0483**	**92.69**	**92.95**	**91.87**	**78.62**
LLFF-test	NeRF w/o watermark	26.37	0.8352	0.1013	N/A			
	HiDDeN [54]+NeRF	25.70	0.8300	0.1096	51.69	50.48	50.53	51.11
	MBRS [14]+NeRF	25.44	0.8198	0.1238	50.84	51.55	50.85	49.08
	CopyRNeRF [25]	25.80	0.8302	0.1035	63.72	60.45	55.34	54.11
	NeRFProtector	**26.82**	**0.8569**	**0.0834**	**96.99**	**93.57**	**80.53**	**76.26**

Evaluation methodology. We assess our method against other baselines using the established digital watermarking standards of invisibility and robustness [7]. For **invisibility**, we measure the performance by employing metrics including PSNR, SSIM [46], and LPIPS [51] to compare the visual fidelity of the output images after binary message embedding. For **robustness**, we conduct experiments to determine if the embedded messages can be accurately extracted by assessing the bit accuracy under different types of image distortions. We consider various distortions [11,36] for message extraction, including JPEG compression, image crop, image scaling, contrast changes, and text overlay. We directly set the bit length used in our experiments as 48 bits, the largest length used in previous methods for watermarking 3D models [25,47]. We further discuss potential intended watermarking removal technologies in Sec. 5.3.

Invisibility against Bit accuracy. As shown in Tab. 1 and Fig. 3, we first investigate the invisibility of our watermarks. This is to show whether the embedded messages undermine the visual quality. Similar to previous studies [25], for HiDDeN [54]/MBRS [14] + NeRF, binary messages cannot be effectively extracted from rendered images, though they all achieve acceptable visual quality. CopyRNeRF [25] achieves the second-best results while it is still largely below our values for bit accuracy. Only our approach can effectively balance visual quality (PSNR/SSIM/LPIPS) and message embedding (Bit accuracy). Besides, from Fig. 3, the viewpoints have a higher impact on the bit accuracy of message extraction, while our method can achieve more consistent performances across different viewpoints.

Is it robust to common distortion? Our method is robust to common image-level distortions since we have considered their impact during building watermarking base model $\mathcal{F}$. As shown in Tab. 1 and 4, when image operations are applied to the rendered image, our base model can still effectively extract the embedded binary messages on both evaluation datasets.

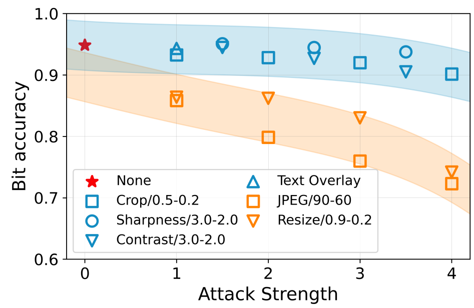

Fig. 4: Image-level distortion experiment on common image distortions on rendered views to determine if the extractor can still extract the message. Operations like sharpness, cropping, contrast, and text overlay have little impact on message extraction. Even after severe operations (JPEG and Resize), our method still maintains satisfactory accuracy ($\geq 70\%$).

Effectiveness. We compare the training time of our method and CopyRNeRF [25] with same message length setting. In contrast to CopyRNeRF, our approach costs only around 50m. As CopyRNeRF [25] jointly trains NeRF and the watermarking extractor at each step with a random message, it significantly extends the time required to embed the watermark (30h).

5.2 Ablation study

Is progressive global rendering good? We propose a progressive distillation to effectively distill message patterns from the base model to NeRF. We compare three settings in Tab. 2: 1) **Local rendering only**, the rendering strategy used in vanilla NeRF; 2) **Single-scale global rendering**: we replace the progressive global rendering with a single-scale rendering; 3) **Our complete progressive global rendering**.

Table 2: Quantitative results for evaluating progressive global rendering. Local denotes the rendering strategy used in vanilla NeRF. Single-scale represents single-scale global rendering from progressive rendering. Progressive denotes our proposed progressive global rendering.

Settings	PSNR ↑	SSIM ↑	LPIPS ↓	Bit acc. %
① Local	30.38	0.9521	0.0360	45.99
② Single-scale	29.57	0.9402	0.0449	87.27
③ Progressive	29.26	0.9394	0.0483	92.69

As shown in Tab. 2, the local rendering in current mechanisms of NeRF cannot guarantee successful message embedding, where the bit accuracy is kept at a very low level. Besides, we also show the heat maps obtained from our progressive global rendering and classical local rendering in Fig. 5. From the results for "Single-scale global rendering", even with a single-scale global rendering, the accuracy of embedding bits is further enhanced while ensuring the reconstruction quality. Our experiments have demonstrated that the progressive rendering approach more effectively incorporates distillation information into the NeRF model.

Can proposed method adapt to other NeRF variants? Our base model for watermarking can also be easily adapted to other NeRF variants. As displayed in Tab. 3, besides Instant-NGP [31] used in our experiments, we further combine the base model with TensorRF [4], and Plenoxels [12]. From the experimental results, our base model for watermarking can also achieve comparable performance on different NeRF variants.

Can the base model be other extractors? In our default setting, we use the extractor of HiDDeN [54] as our base model. However, we can also consider using other extractors trained in different settings [14] as our base model. In Tab. 4, we show that message extractor from MBRS [14] can also be used as the base model, and they also achieve comparable performance.

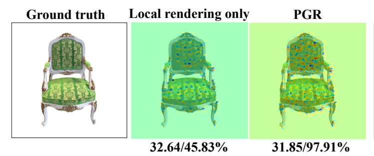

Fig. 5: Visual heat map comparisons of local rendering and progressive global rendering. The text below denotes PSNR/Bit accuracy. The colors represent the area's impact on message extraction. Our PGR can distill messages into NeRF with high accuracy.

Table 3: Quantitative results on combining the same watermarking base model with different NeRF variants. The extractor of HiDDeN [54] is employed as the watermarking base model $\mathcal{F}_1$. We combine $\mathcal{F}_1$ with Instant-NGP [31], TensorRF [4], and Plenoxels [12] to see whether the base model can be adapted to NeRF variants in a plug-and-play manner. N/A denotes not applicable as message embedding is not involved.

Method	PSNR ↑	SSIM ↑	LPIPS ↓	Bit acc.%
Instant NGP [31]	33.42	0.9709	0.0157	N/A
Instant NGP + $\mathcal{F}_1$	32.92	0.9697	0.0161	91.96
TensorRF [4]	34.96	0.9786	0.0105	N/A
TensorRF + $\mathcal{F}_1$	32.73	0.9743	0.0125	89.35
Plenoxels [12]	35.80	0.9872	0.0102	N/A
Plenoxels + $\mathcal{F}_1$	34.19	0.9730	0.0127	97.92

Table 4: Quantitative results on combining different watermarking base model. $\mathcal{F}_1$ and $\mathcal{F}_2$ denote HiDDeN [54] and MBRS [14] extractors. We combine a different base model from MBRS [14] with Instant-NGP [31] to see whether the backbone of NeRF can be adapted to other base models. N/A denotes not applicable as message embedding is not involved.

Method	PSNR ↑	SSIM ↑	LPIPS ↓	Bit acc.%
Instant-NGP [31]	33.42	0.9709	0.0157	N/A
Instant-NGP + $\mathcal{F}_1$	32.92	0.9697	0.0161	91.96
Instant-NGP + $\mathcal{F}_2$	31.71	0.9635	0.0152	89.13

5.3 Analyzing potential threats

Our target is to propose a framework that is able to utilize the robustness within the base model to defend some common image-level threats. The experiments also demonstrate the advantages of our approach. We further examine the embedded watermark's robustness to more potential intentional tampering and threats.

Threat via neural compression. Malicious users may alter the image to remove the watermark with deep learning techniques, like methods used for image compression with neural auto-encoders [3, 6, 30]. We evaluate the robustness of the watermark against neural auto-encoders at different compression rates (Fig. 6). Specifically, when the bit accuracy decreases to 50%, a threshold indicating watermarking failure, the image quality significantly deteriorates. This suggests that our watermarking method is resilient to the effects of auto-encoder models. Even in extreme cases where bit accuracy reaches 50%, the image quality is so compromised that the images are no longer suitable for sharing or distribution purposes.

Threat via watermarking base model (white-box attack). In this scenario, attackers obtain the watermarking base model used by the NeRF creator. Then, malicious users can implement adversarial attacks [26] with a leaked watermarking base model to remove copyright messages in rendered images. We launch a Projected gradient descent (PGD) attack [26], a classic adversarial attack on rendered images. As depicted in Fig. 6, experimental results show attackers can remove messages with minimal distortion to the rendered content. This highlights the importance of keeping the watermarking base models confidential [11, 17].

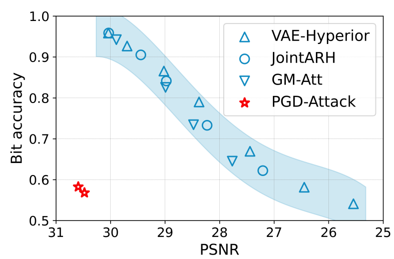

Fig. 6: Attacks on rendered images. We evaluate neural compression methods including VAE-Hyperior [3], JointARH [30] and GM-Att [6]. PGD-attack [26] is also applied to attack the base model. Attacks using image compression methods can only succeed when there is a significant decrease in image quality. However, it is possible to remove watermarks with PGD attack [26] on the base model without obvious distortion.

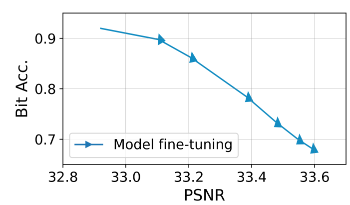

Fig. 7: In an extreme situation, attackers might gain access to the images utilized in creating NeRF. With these unwatermarked images, they could then fine-tune the model to eliminate the watermarks. In this example, as the fine-tuning progresses, the PSNR keeps increasing while the detected bit accuracy decreases.

Threat on the representation of NeRF. If malicious users have obtained NeRF models, they may choose to eliminate the watermarks by finetuning the

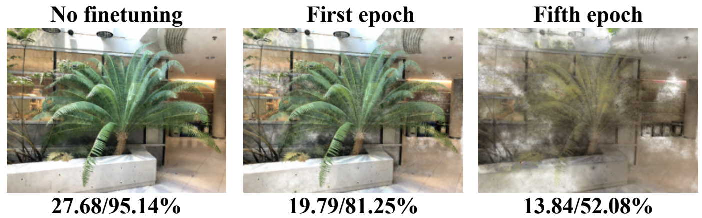

Fig. 8: Attackers might fine-tune watermarked NeRF with unrelated scenes to remove the watermarks. However, the model quality significantly decreases after fine-tuning NeRF via unrelated images. The text below the image shows the results of PSNR and bit accuracy.

models with no-watermark images. This process involves finetuning NeRF only via the content loss in Eq. (2) without the message embedding process.

We assume that malicious users can obtain the original images corresponding to the scenes represented by NeRF. Then, from the results shown in Fig. 7, the bit accuracy indeed drops, highlighting the need to securely manage both the original data and the NeRF models. However, if malicious users cannot obtain the images originally used for creating the NeRF, launching the attack becomes difficult. For example, if they directly use some images unrelated to the scenes represented by NeRF, it may significantly change the information stored in NeRF, making the model sharing unfeasible (as shown in Fig. 8).

6 Conclusion

This paper introduces a plug-and-play method for watermarking NeRF during their creation. Utilizing a base watermarking model, NeRF creators can embed binary messages directly into their models as they are being developed. We propose a progressive global rendering to integrate message patterns into NeRF. The base model inherently offers basic robustness to its users. Our experimental results showcase the effectiveness of our approach. Our base model is compatible with NeRF variants, offering a more versatile watermarking solution.

Limitations. Though we have provided an effective technical solution for protecting the copyright of NeRF, copyright protection should be a multi-pronged effort involving various stakeholders. The legitimate enforcement of copyright requires a comprehensive strategy that goes beyond technological solutions. Legislative measures from relevant parties are also important to establish a regulatory framework that safeguards intellectual property rights.

Acknowledgements. This work was done at Renjie's Research Group at the Department of Computer Science of Hong Kong Baptist University. Renjie's Research Group is supported by the National Natural Science Foundation of China under Grant No. 62302415, Guangdong Basic and Applied Basic Research Foundation under Grant No. 2022A1515110692, 2024A1515012822, and the Blue Sky Research Fund of HKBU under Grant No. BSRF/21-22/16.

References

1. Alzahrani, A., et al.: Enhanced invisibility and robustness of digital image watermarking based on DWT-SVD. Appl. Bionics. Biomech (2022)
2. Asikuzzaman, M., Pickering, M.R.: An overview of digital video watermarking. TCSVT (2017)
3. Ballé, J., Minnen, D., Singh, S., Hwang, S.J., Johnston, N.: Variational image compression with a scale hyperprior. In: ICLR (2018)
4. Chen, A., Xu, Z., Geiger, A., Yu, J., Su, H.: TensoRF: Tensorial radiance fields. In: ECCV (2022)
5. Cheng, Y., Wan, R., Weng, S., Zhu, C., Chang, Y., Shi, B.: Colorizing monochromatic radiance fields. In: AAAI (2024)
6. Cheng, Z., Sun, H., Takeuchi, M., Katto, J.: Learned image compression with discretized gaussian mixture likelihoods and attention modules. In: CVPR (2020)
7. Cox, I., Miller, M., Bloom, J., Honsinger, C.: Digital watermarking. J. Electron Imaging (2002)
8. Doerr, G., Dugelay, J.L.: A guide tour of video watermarking. Signal Process. Image Commun. (2003)
9. Evsutin, O., Dzhanashia, K.: Watermarking schemes for digital images: Robustness overview. Signal Process. Image Commun (2022)
10. Fan, Z., Wang, P., Jiang, Y., Gong, X., Xu, D., Wang, Z.: NeRF-SOS: Any-view self-supervised object segmentation on complex scenes. In: ICLR (2023)
11. Fernandez, P., Couairon, G., Jégou, H., Douze, M., Furon, T.: The Stable Signature: Rooting watermarks in latent diffusion models. In: ICCV (2023)
12. Fridovich-Keil, S., Yu, A., Tancik, M., Chen, Q., Recht, B., Kanazawa, A.: Plenoxels: Radiance fields without neural networks. In: CVPR (2022)
13. Hamidi, M., Chetouani, A., El Haziti, M., El Hassouni, M., Cherifi, H.: Blind robust 3D mesh watermarking based on mesh saliency and wavelet transform for copyright protection. Information (2019)
14. Jia, Z., Fang, H., Zhang, W.: MBRS: Enhancing robustness of dnn-based watermarking by mini-batch of real and simulated jpeg compression. In: ACM MM (2021)
15. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
16. Li, C., Feng, B.Y., Fan, Z., Pan, P., Wang, Z.: StegaNeRF: Embedding invisible information within neural radiance fields. In: ICCV (2023)
17. Li, J., Pang, M., Dong, Y., Jia, J., Wang, B.: Graph neural network explanations are fragile. In: ICML (2024)
18. Li, Y., Li, Y., Wu, B., Li, L., He, R., Lyu, S.: Invisible backdoor attack with sample-specific triggers. In: ICCV (2021)
19. Lin, C.H., Gao, J., Tang, L., Takikawa, T., Zeng, X., Huang, X., Kreis, K., Fidler, S., Liu, M.Y., Lin, T.Y.: Magic3D: High-resolution Text-to-3D content creation. In: CVPR (2023)
20. Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft COCO: Common objects in context. In: ECCV (2014)
21. Liu, J., Yang, Y., Ma, D., He, W., Wang, Y.: A novel watermarking algorithm for three-dimensional point-cloud models based on vertex curvature. Int J Distrib Sens N (2019)
22. Liu, R., Wu, R., Van Hoorick, B., Tokmakov, P., Zakharov, S., Vondrick, C.: Zero-1-to-3: Zero-shot one image to 3D object. In: ICCV (2023)

23. Liu, S., Zhang, X., Zhang, Z., Zhang, R., Zhu, J.Y., Russell, B.: Editing conditional radiance fields. In: ICCV (2021)
24. Luo, X., Li, Y., Chang, H., Liu, C., Milanfar, P., Yang, F.: DVMark: A deep multiscale network for video watermarking. TIP (2023)
25. Luo, Z., Guo, Q., Cheung, K.C., See, S., Wan, R.: CopyRNeRF: Protecting the copyright of neural radiance fields. In: ICCV (2023)
26. Madry, A., Makelov, A., Schmidt, L., Tsipras, D., Vladu, A.: Towards deep learning models resistant to adversarial attacks. In: ICLR (2018)
27. Metzer, G., Richardson, E., Patashnik, O., Giryes, R., Cohen-Or, D.: Latent-NeRF for shape-guided generation of 3D shapes and textures. In: CVPR (2023)
28. Mildenhall, B., Srinivasan, P.P., Ortiz-Cayon, R., Kalantari, N.K., Ramamoorthi, R., Ng, R., Kar, A.: Local light field fusion: Practical view synthesis with prescriptive sampling guidelines. TOG (2019)
29. Mildenhall, B., Srinivasan, P.P., Tancik, M., Barron, J.T., Ramamoorthi, R., Ng, R.: NeRF: Representing scenes as neural radiance fields for view synthesis. Commun. ACM (2021)
30. Minnen, D., Ballé, J., Toderici, G.D.: Joint autoregressive and hierarchical priors for learned image compression. In: NeurIPS (2018)
31. Müller, T., Evans, A., Schied, C., Keller, A.: Instant neural graphics primitives with a multiresolution hash encoding. TOG (2022)
32. Ohbuchi, .R., Mukaiyama, .A., Takahashi, .S.: A frequency-domain approach to watermarking 3D shapes. In: Comput Graph Forum (2002)
33. Poole, B., Jain, A., Barron, J.T., Mildenhall, B.: DreamFusion: Text-to-3D using 2D diffusion. In: ICLR (2023)
34. Praun, E., Hoppe, H., Finkelstein, A.: Robust mesh watermarking. In: CGIT (1999)
35. Qi, C.R., Yi, L., Su, H., Guibas, L.J.: PointNet++: Deep hierarchical feature learning on point sets in a metric space. In: NeurIPS (2017)
36. Qi, S., Shi, W., Ge, G., Chang, L.: Degradation conditioned GAN for degradation generalization of face restoration models. In: ICIP (2023)
37. Radford, A., Kim, J.W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., et al.: Learning transferable visual models from natural language supervision. In: ICML (2021)
38. Son, J., Kim, D., Choi, H.Y., Jang, H.U., Choi, S.: Perceptual 3D watermarking using mesh saliency. In: ICISA (2017)
39. Su, Q., Liu, D., Sun, Y.: A robust adaptive blind color image watermarking for resisting geometric attacks. Inform Sciences (2022)
40. Tancik, M., Mildenhall, B., Ng, R.: StegaStamp: Invisible hyperlinks in physical photographs. In: CVPR (2020)
41. Tang, J., Wang, T., Zhang, B., Zhang, T., Yi, R., Ma, L., Chen, D.: Make-It-3D: High-fidelity 3D creation from a single image with diffusion prior. In: ICCV (2023)
42. Tang, Y., Zhu, C., Wan, R., Xu, C., Shi, B.: Neural underwater scene representation. In: CVPR (2024)
43. Wang, C., Chai, M., He, M., Chen, D., Liao, J.: CLIP-NeRF: Text-and-image driven manipulation of neural radiance fields. In: CVPR (2022)
44. Wang, H., Ren, J., Huang, Z., Olszewski, K., Chai, M., Fu, Y., Tulyakov, S.: R2L: Distilling neural radiance field to neural light field for efficient novel view synthesis. In: ECCV (2022)
45. Wang, R., Wan, R., Guo, Z., Guo, Q., Huang, R.: Spy-Watermark: Robust invisible watermarking for backdoor attack. In: ICASSP (2024)
46. Wang, Z., Bovik, A.C., Sheikh, H.R., Simoncelli, E.P.: Image quality assessment: From error visibility to structural similarity. TIP (2004)

47. Yoo, I., Chang, H., Luo, X., Stava, O., Liu, C., Milanfar, P., Yang, F.: Deep 3D-to-2D watermarking: Embedding messages in 3D meshes and extracting them from 2D renderings. In: CVPR (2022)
48. Yu, A., Ye, V., Tancik, M., Kanazawa, A.: pixelNeRF: Neural radiance fields from one or few images. In: CVPR (2021)
49. Yuan, Y.J., Sun, Y.T., Lai, Y.K., Ma, Y., Jia, R., Gao, L.: NeRF-Editing: Geometry editing of neural radiance fields. In: CVPR (2022)
50. Zhang, K.A., Cuesta-Infante, A., Xu, L., Veeramachaneni, K.: SteganoGAN: High capacity image steganography with gans. arXiv preprint arXiv:1901.03892 (2019)
51. Zhang, R., Isola, P., Efros, A.A., Shechtman, E., Wang, O.: The unreasonable effectiveness of deep features as a perceptual metric. In: CVPR (2018)
52. Zhu, C., Wan, R., Shi, B.: Neural transmitted radiance fields. In: NeurIPS (2022)
53. Zhu, C., Wan, R., Tang, Y., Shi, B.: Occlusion-free scene recovery via neural radiance fields. In: CVPR (2023)
54. Zhu, J., Kaplan, R., Johnson, J., Fei-Fei, L.: HiDDeN: Hiding data with deep networks. In: ECCV (2018)