CopyRNeRF: Protecting the CopyRight of Neural Radiance Fields

Ziyuan Luo[1,2] Qing Guo[3] Ka Chun Cheung[2,4] Simon See[2] Renjie Wan[1]*
[1]Department of Computer Science, Hong Kong Baptist University
[2]NVIDIA AI Technology Center, NVIDIA
[3]IHPC and CFAR, Agency for Science, Technology and Research, Singapore
[4]Department of Mathematics, Hong Kong Baptist University
ziyuanluo@life.hkbu.edu.hk, guo_qing@cfar.a-star.edu.sg, {chcheung, ssee}@nvidia.com,
renjiewan@hkbu.edu.hk

Abstract

Neural Radiance Fields (NeRF) have the potential to be a major representation of media. Since training a NeRF has never been an easy task, the protection of its model copyright should be a priority. In this paper, by analyzing the pros and cons of possible copyright protection solutions, we propose to protect the copyright of NeRF models by replacing the original color representation in NeRF with a watermarked color representation. Then, a distortion-resistant rendering scheme is designed to guarantee robust message extraction in 2D renderings of NeRF. Our proposed method can directly protect the copyright of NeRF models while maintaining high rendering quality and bit accuracy when compared among optional solutions. Project page: https://luo-ziyuan.github.io/copyrnerf.

1. Introduction

Though Neural Radiance Fields (NeRF) [23] have the potential to be the mainstream for the representation of digital media, training a NeRF model has never been an easy task. If a NeRF model is stolen by malicious users, *how can we identify its intellectual property?*

As with any digital asset (*e.g.*, 3D model, video, or image), copyright can be secured by embedding copyright messages into assest, aka digital watermarking, and NeRF models are no exception. An intuitive solution is to directly watermark rendered samples using an off-the-shelf watermarking approach (*e.g.*, HiDDeN [50] and MBRS [14]). However, this only protects the copyright of rendered samples, leaving the core model unprotected. If the core model has been stolen, malicious users may render new samples using different rendering strategies, leaving no room for external watermarking expected by model creators. Besides, without considering factors necessary for rendering during watermarking, directly watermarking rendered samples may leave easily detectable trace on areas with low geometry values.

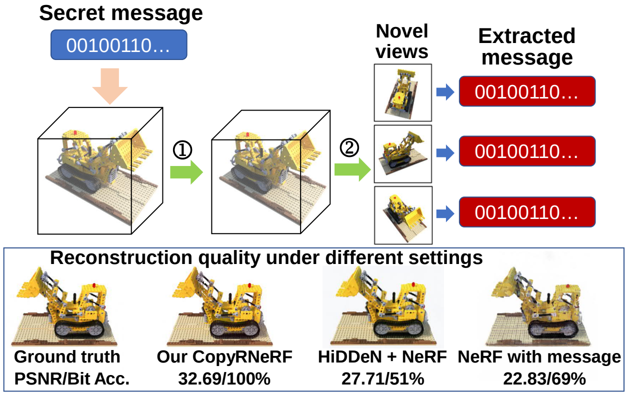

Figure 1: When NeRF models are stolen (①) by maclicious users, CopyRNeRF can help to claim model ownership by transmitting copyright messages embedded in models to rendering samples (②). We show some comparisons with HiDDeN [50] + NeRF [23], and NeRF [23] with messages. PSNR/Bit Accuracy is shown below each example.

The copyright messages are usually embedded into 3D structure (*e.g.*, meshes) for explicit 3D models [43]. Since such structures are all implicitly encoded into the weights of multilayer perceptrons (MLP) for NeRF, its copyright protection should be conducted by watermarking model weights. As the infomration encoded by NeRF can only be accessed via 2D renderings of protected models, two common standards should be considered during the watermark extraction on rendered samples [1, 15, 41, 45]: 1) **invisibility**, which requires that no serious visual distoration are caused by embedded messages, and 2) **robustness**, which ensures robust message extraction even when various distortions are encountered.

*Corresponding author.

One option is to create a NeRF model using watermarked images, while the popular invisible watermarks on 2D images cannot be effectively transmitted into NeRF models. As outlined in Figure 1 (HiDDeN [50] + NeRF [23]), though the rendered results are of high quality, the secret messages cannot be robustly extracted. We can also directly concatenate secret messages with input coordinates, which produces higher bit accuracy (NeRF with message in Figure 1). However, the lower PSNR values of rendered samples indicate that there is an obvious visual distortion, which violates the standard for invisibility.

Though invisibility is important for a watermarking system, the higher demand for robustness makes watermarking unqiue [50]. Thus, in addition to invisibility, we focus on a more robust protection of NeRF models. As opposed to embedding messages into the entire models as in the above settings, we create a *watermarked color representation* for rendering based on a subset of models, as displayed in Figure 2. By keeping the base representation unchanged, this approach can produce rendering samples with invisible watermarks. By incorporating spatial information into the watermarked color representation, the embedded messages can remain consistent across different viewpoints rendered from NeRF models. We further strengthen the robustness of watermark extraction by using *distortion-resistant rendering* during model optimization. A distortion layer is designed to ensure robust watermark extraction even when the rendered samples are severly distorted (*e.g.*, blurring, noise, and rotation). A random sampling strategy is further considered to make the protected model robust to different sampling strategy during rendering.

Distortion-resistant rendering is only needed during the optimization of core models. If the core model is stolen, even with different rendering schemes and sampling strategies, the copyright message can still be robustly extracted. Our contribution can be summarized as follows:

- a method to produce copyright-embedded NeRF models.
- a watermarked color representation to ensure invisibility and high rendering quality.
- distortion-resistant rendering to ensure robustness across different rendering strategies or 2D distortions.

2. Related work

Neural radiance fields. Various neural implicit scene representation schemes have been introduced recently [25, 42, 48]. The Scene Representation Networks (SNR) [32] represent scenes as a multi-layer perceptron (MLP) that maps world coordinates to local features, which can be trained from 2D images and their camera poses. DeepSDF [27] and DIST [20] use trained networks to represent a continuous signed distance function of a class of shapes. PIFu [30] learned two pixel-aligned implicit functions to infer surface and texture of clothed humans respectively from a single input image. Occupancy Networks [21, 28] are proposed as an implicit representation of 3D geometry of 3D objects or scenes with 3D supervision. NeRF [23, 49] in particular directly maps the 3D position and 2D viewing direction to color and geometry by a MLP and synthesize novel views via volume rendering. The improvements and applications of this implicit representation have been rapidly growing in recent years, including NeRF accelerating [9, 24], sparse reconstruction [44, 6], and generative models [31, 5]. NeRF models are not easy to train and may use private data, so protecting their copyright becomes crucial.

Digital watermarking for 2D. Early 2D watermarking approaches encode information in the least significant bits of image pixels [35]. Some other methods instead encode information in the transform domains [17]. Deep learning based methods for image watermarking have made substantial progress. HiDDeN [50] was one of the first deep image watermarking methods that achieved superior performance compared to traditional watermarking approaches. RedMark [1] introduced residual connections with a strength factor for embedding binary images in the transform domain. Deep watermarking has since been generalized to video [37, 46] as well. Modeling more complex and realistic image distortions also broadened the scope in terms of application [38, 34]. However, those methods all cannot protect the copyright of 3D models.

Digital watermarking for 3D. Traditional 3D watermarking approaches [26, 29, 39] leveraged Fourier or wavelet analysis on triangular or polygonal meshes. Recently, Hou *et al.* [11] introduced a 3D watermarking method using the layering artifacts in 3D printed objects. Son *et al.* [33] used mesh saliency as a perceptual metric to minimize vertex distortions. Hamidi *et al.* [10] further extended mesh saliency with wavelet transform to make 3D watermarking robust. Jing *et al.* [19] studied watermarking for point clouds through analyzing vertex curvatures. Recently, a deep-learning based approach [43] successfully embeds messages in 3D meshes and extracts them from 2D renderings. However, existing methods are for explicit 3D models, which cannot be used for NeRF models with implicit property.

3. Preliminaries

NeRF [23] uses MLPs Θ_σ and Θ_c to map the 3D location $\mathbf{x} \in \mathbb{R}^3$ and viewing direction $\mathbf{d} \in \mathbb{R}^2$ to a color value

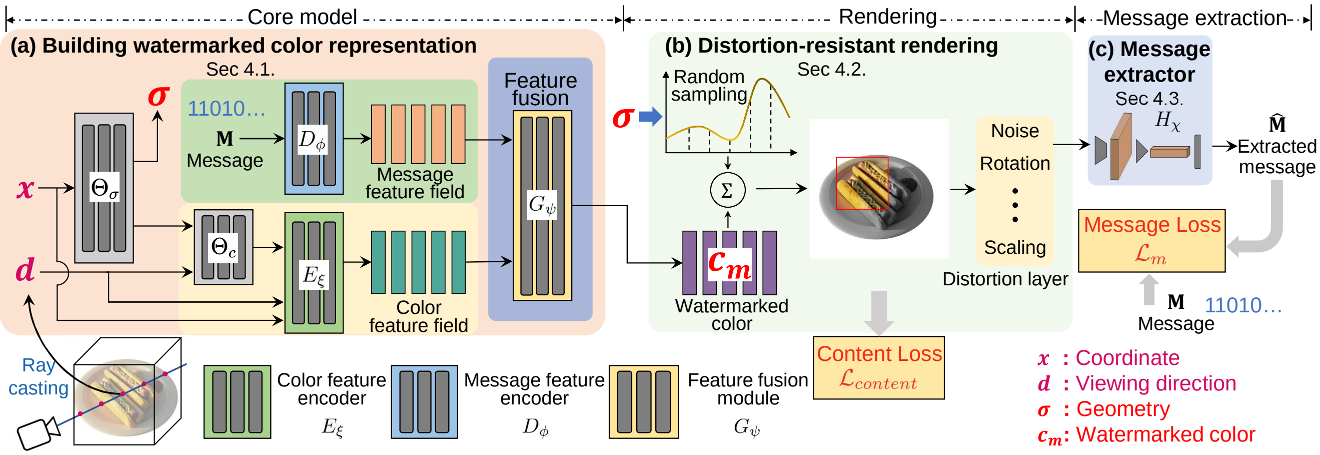

Figure 2: Illustration of our proposed method. (a) A *watermarked color representation* is obtained with the given secret message, which is able to produce watermarked color for rendering. (b) During training, a *distortion-resistant rendering* is deployed to map the geometry (σ) and watermarked color representations to image patches with several distortions. (c) Finally, the secret message can be revealed by a CNN-based *message extractor*.

$\mathbf{c} \in \mathbb{R}^3$ and a geometric value $\sigma \in \mathbb{R}^+$:

$$[\sigma, \mathbf{z}] = \Theta_\sigma\left(\gamma_\mathbf{x}(\mathbf{x})\right), \tag{1}$$

$$\mathbf{c} = \Theta_c\left(\mathbf{z}, \gamma_\mathbf{d}(\mathbf{d})\right), \tag{2}$$

where $\gamma_\mathbf{x}$ and $\gamma_\mathbf{d}$ are fixed encoding functions for location and viewing direction respectively. The intermediate variable $\mathbf{z}$ is a feature output by the first MLP Θ_σ.

For rendering a 2D image from the radiance fields Θ_σ and Θ_c, a numerical quadrature is used to approximate the volumetric projection integral. Formally, N_p points are sampled along a camera ray r with color and geometry values $\{(\mathbf{c}_r^i, \sigma_r^i)\}_{i=1}^N$. The RGB color value $\hat{\mathbf{C}}(r)$ is obtained using alpha composition

$$\hat{\mathbf{C}}(r) = \sum_{i=1}^{N_p} T_r^i (1 - \exp\left(-\sigma_r^i \delta_r^i\right))\mathbf{c}_r^i, \tag{3}$$

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 MLPs Θ_σ and Θ_c are optimized by minimizing a reconstruction loss between observations $\mathbf{C}$ and predictions $\hat{\mathbf{C}}$ as

$$\mathcal{L}_{recon} = \frac{1}{N_r} \sum_{m=1}^{N_r} \|\hat{\mathbf{C}}(r_m) - \mathbf{C}(r_m)\|_2^2, \tag{4}$$

where N_r is the number of sampling pixels. Given Θ_σ and Θ_c, novel views can be synthesized by invoking volume rendering for each ray.

Considering the superior capability of NeRF in rendering novel views and representing various scenes, how can we protect its copyright when it is stolen by malicious users?

4. Proposed method

As outlined in Figure 2, with a collection of 2D images $\{I_n\}_{n=1}^N$ and the binary message $\mathbf{M} \in \{0, 1\}^{N_b}$ with length N_b, we address the issue raised in Section 3 by building a watermarked color representation during optimization. In training, a distortion-resistant rendering is further applied to improve the robustness when 2D distortions or different rendering schemes are encountered. With the above design, the secret messages can be robustly extracted during testing even encountering sever distortions or different rendering strategies.

4.1. Building watermarked color representation

The rendering in Equation (3) relies on color and geometry produced by their corresponding representation in NeRF. To ensure the transmission of copyright messages to the rendered results, we propose embedding messages into their representation. We create a watermarked color representation on the basis of Θ_c defined in Equation (2) to guarantee the message invisibility and consistency across viewpoints. The representation of geometry is also the potential for watermarking, but external information on geometry may undermine rendering quality [36, 12, 7]. Therefore, the geometry does not become our first option, while experiments are also conducted to verify this setting.

We keep the geometry representation in Equation (1) unchanged, and construct the watermarked color representation Θ_m to produce the message embedded color $\mathbf{c}_m$ as follows:

$$\mathbf{c}_m = \Theta_m\left(\mathbf{c}, \gamma_\mathbf{x}(\mathbf{x}), \gamma_\mathbf{d}(\mathbf{d}), \mathbf{M}\right), \tag{5}$$

where $\mathbf{M}$ denotes the message to be embedded and Θ_m contains several MLPs to ensure reliable message embedding.

The input $\mathbf{c}$ is obtained by querying Θ_c using Equation (2). Several previous methods have pointed out the importance of building a 3D feature field when distributed features are needed to characterize composite information [40, 4]. Thus, instead of directly fusing those information, we first construct their corresponding feature field and then combine them progressively.

Color feature field. In this stage, we aim at fusing the spatial information and color representation to ensure message consistency and robustness across viewpoints. We adopt a color feature field by considering color, spatial positions, and viewing directions simultaneously as follows:

$$f_c = E_\xi(\mathbf{c}, \gamma_{\mathbf{x}}(\mathbf{x}), \gamma_{\mathbf{d}}(\mathbf{d})). \tag{6}$$

Given a 3D coordinate $\mathbf{x}$ and a viewing direction $\mathbf{d}$, we first query the color representation $\Theta_c\left(\mathbf{z}, \gamma_{\mathbf{d}}(\mathbf{d})\right)$ to get $\mathbf{c}$, and then concatenate them with $\mathbf{x}$ and $\mathbf{d}$ to obtain spatial descriptor $\mathbf{v}$ as the input. Then the color feature encoder E_ξ transforms $\mathbf{v}$ to the high-dimensional color feature field f_c with dimension N_c. The Fourier feature encoding is applied to $\mathbf{x}$ and $\mathbf{d}$ before the feature extraction.

Message feature field. We further construct the message feature field. Specifically, we follow the classical setting in digital watermarking by transforming secret messages into higher dimensions [2, 3]. It ensures more succinctly encoding of desired messages [2]. As shown in Figure 2, a message feature encoder is applied to map the messages to its corresponding higher dimensions as follows:

$$f_{\mathbf{M}} = D_\phi(\mathbf{M}). \tag{7}$$

In Equation (7), given message $\mathbf{M}$ of length N_b, the message feature encoder D_ϕ applies a MLP to the input message, resulting in a message feature field $f_{\mathbf{M}}$ of dimension N_m.

Then, the watermarked color can be generated via a feature fusion module G_ψ that integrates both color feature field and message feature field as follows:

$$\mathbf{c}_m = G_\psi(f_c, f_{\mathbf{M}}, \mathbf{c}). \tag{8}$$

Specifically, $\mathbf{c}$ is also employed here to make the final results more stable. $\mathbf{c}_m$ is with the same dimension to $\mathbf{c}$, which ensures this representation can easily adapt to current rendering schemes.

4.2. Distortion-resistant rendering

Directly employing the watermarked representation for volume rendering has already been able to guarantee invisibility and robustness across viewpoints. However, as discussed in Section 1, the message should be robustly extracted even when encountering diverse distortion to the rendered 2D images. Besides, for an implicit model relying on rendering to display its contents, the robustness should also be secured even when different rendering strategies are employed. Such requirement for robustness cannot be achieved by simply using watermarked representation under the classical NeRF training framework. For example, the pixel-wise rendering strategy cannot effectively model the distortion (*e.g.*, blurring and cropping) only meaningful in a wider scale. We, therefore, propose a distortion-resistant rendering by strengthing the robustness using a random sampling strategy and distortion layer.

Since most 2D distortions can only be obviously observed in a certain area, we consider the rendering process in a patch level [16, 8]. A window with the random position is cropped from the input image with a certain height and width, then we uniformly sample the pixels from such window to form a smaller patch. The center of the patch is denoted by $\mathbf{u} = (u, v) \in \mathbb{R}^2$, and the size of patch is determined by $K \in \mathbb{R}^+$. We randomly draw the patch center $\mathbf{u}$ from a uniform distribution $\mathbf{u} \sim \mathcal{U}(\Omega)$ over the image domain Ω. The patch $\mathcal{P}(\mathbf{u}, K)$ can be denoted by by a set of 2D image coordinates as

$$\mathcal{P}(\mathbf{u}, K) = \{(x+u, y+v) \mid x, y \in \{-\frac{K}{2}, \dots, \frac{K}{2} - 1\}\}. \tag{9}$$

Such a patch-based scheme constitutes the backbone of our distortion-resistant rendering, due to its advantages in capturing information on a wider scale. Specifically, we employ a variable patch size to accommodate diverse distortions during rendering, which can ensure higher robustness in message extraction. This is because small patches increase the robustness against cropping attacks and large patches allow higher redundancy in the bit encoding, which leads to increased resilience against random noise [8].

As the corresponding 3D rays are uniquely determined by $\mathcal{P}(\mathbf{u}, K)$, the camera pose and intrinsics, the image patch $\widetilde{\mathbf{P}}$ can be obtained after points sampling and rendering. Based on the sampling points in Section 3, we use a random sampling scheme to further improve the model's robustness, which is described as follows.

Random sampling. During volume rendering, NeRF [23] is required to sample 3D points along a ray to calculate the RGB value of a pixel color. However, the sampling strategy may vary as the renderer changes [24, 18]. To make our message extraction more robust even under different sampling strategies, we employ a random sampling strategy by adding a shifting value to the sampling points. Specifically, the original N_p sampling points along ray r is denoted by a sequence, which can be concluded as $\mathcal{X} = (x_r^1, x_r^2, \cdots, x_r^{N_p})$, where $x_r^i, i = 1, 2, \cdots, N_p$ denotes the sampling points during rendering. The randomized sample sequence $\mathcal{X}_{random}$ can be denoted by adding a shifting value as

$$\mathcal{X}_{random} = (x_r^1 + z^1, x_r^2 + z^2, \cdots, x_r^{N_p} + z^{N_p}),\\ z^i \sim \mathcal{N}(0, \beta^2),\ i = 1, 2, \cdots, N_p, \tag{10}$$

where $\mathcal{N}(0, \beta^2)$ is the Gaussian distribution with zero mean and standard deviation β.

By querying the watermarked color representation and geometry values at N_p points in $\mathcal{X}_{random}$, the rendering operator can be then applied to generate the watermarked color $\widetilde{\mathbf{C}}_m$ in rendered images:

$$\widetilde{\mathbf{C}}_m(r) = \sum_{i=1}^{N_p} T_r^i (1 - \exp\left(-\sigma_r^i \delta_r^i\right)) \mathbf{c}_m^i, \tag{11}$$

where T_r^i and δ_r^i are with the same definitions to their counterparts in Equation (3).

All the colors obtained by coordinates $\mathcal{P}$ can form a $K \times K$ image patch $\widetilde{\mathbf{P}}$. The content loss $\mathcal{L}_{content}$ of the 3D representation is calculated between watermarked patch $\widetilde{\mathbf{P}}$ and the $\hat{\mathbf{P}}$, where $\hat{\mathbf{P}}$ is rendered from the non-watermarked representation by the same coordinates $\mathcal{P}$. In detail, the content loss $\mathcal{L}_{content}$ has two components namely pixel-wise MSE loss and perceptual loss:

$$\mathcal{L}_{content} = \|\widetilde{\mathbf{P}} - \hat{\mathbf{P}}\|_2^2 + \lambda \|\Psi(\widetilde{\mathbf{P}}) - \Psi(\hat{\mathbf{P}})\|_2^2, \tag{12}$$

where $\Psi(\cdot)$ denotes the feature representation obtained from a VGG-16 network, and λ is a hyperparameter to balance the loss functions.

Distortion layer. To make our watermarking system robust to 2D distortions, a distortion layer is employed in our watermarking training pipeline after the patch $\widetilde{\mathbf{P}}$ is rendered. Several commonly used distortions are considered: 1) additive Gaussian noise with mean μ and standard deviation ν; 2) random axis-angle rotation with parameters α; and 3) random scaling with a parameter s; 4) Gaussian blur with kernel k. Since all these distortions are differentiable, we could train our network end-to-end.

The distortion-resistant rendering is only applied during training. It is not a part of the core model. If the core model is stolen, even malicous users use different rendering strategy, the expected robustness can still be secured.

4.3. Message extractor

To retrieve message $\hat{\mathbf{M}}$ from the $K \times K$ rendered patch $\mathbf{P}$, a message extractor H_χ is proposed to be trained end-to-end:

$$H_\chi : \mathbb{R}^{K \times K} \to \mathbb{R}^{N_b},\ \mathbf{P} \mapsto \hat{\mathbf{M}}, \tag{13}$$

where χ is a trainable parameter. Specifically, we employ a sequence of 2D convolutional layers with the batch normalization and ReLU functions [13]. An average pooling is then performed, following by a final linear layer with a fixed output dimension N_b, which is the length of the message, to produce the continuous predicted message $\hat{\mathbf{M}}$. Because of the use of average pooling, the message extractor is compatible with any patch sizes, which means the network structure can remain unchanged when applying size-changing distortions such as random scaling.

The message loss $\mathcal{L}_m$ is then obtained by calculating the mean square error between predicted message $\hat{\mathbf{M}}$ and the ground truth message $\mathbf{M}$:

$$\mathcal{L}_m = \|\hat{\mathbf{M}} - \mathbf{M}\|_2^2. \tag{14}$$

To evaluate the bit accuracy during testing, the binary predicted message $\hat{\mathbf{M}}_b$ can be obtained by rounding:

$$\hat{\mathbf{M}}_b = \mathrm{clamp}(\mathrm{sign}(\hat{\mathbf{M}}), 0, 1), \tag{15}$$

where clamp and sign are of the same definitions in [43]. It should be noted that we use the continuous result $\hat{\mathbf{M}}$ in the training process, while the binary one $\hat{\mathbf{M}}_b$ is only adopted in testing process.

Therefore, the overall loss to train the copyright-protected neural radiance fields can be obtained as

$$\mathcal{L} = \gamma_1 \mathcal{L}_{content} + \gamma_2 \mathcal{L}_m, \tag{16}$$

where γ_1 and γ_2 are hyperparameters to balance the loss functions.

4.4. Implementation details

We implement our method using PyTorch. An eight-layer MLP with 256 channels and the following two MLP branches are used to predict the original colors $\mathbf{c}$ and opacities σ, respectively. We train a "coarse" network along with a "fine" network network for importance sampling. we sample 32 points along each ray in the coarse model and 64 points in the fine model. Next, the patch size is set to 150×150. The hyperparameters in Equation (12) and Equation (16) are set as $\lambda_1 = 0.01$, $\gamma_1 = 1$, and $\gamma_2 = 10.00$. We use the Adam optimizer with defaults values $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$, and a learning rate 5×10^{-4} that decays following the exponential scheduler during optimization. In our experiments, we set N_m in Equation (7) as 256. We first optimize MLPs Θ_σ and Θ_c using loss function Equation (4) for 200K and 100K iterations for Blender dataset [23] and LLFF dataset [22] separately, and then train the models E_ξ, D_ϕ, and H_χ for about 500K iterations on a single NVIDIA V100 GPU. During training, we have considered messages with different bit lengths and forms. If a message has 4 bits, we take into account all 2^4 situations during training. The model creator can choose one message considered in our training as the desired message.

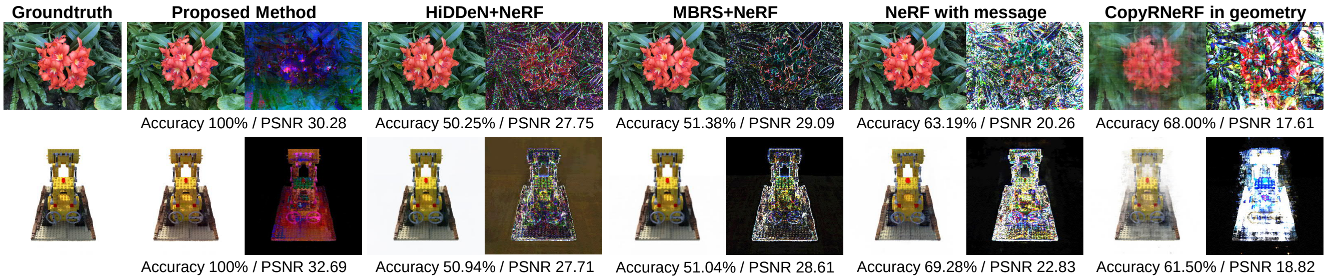
Groundtruth
Proposed Method
HiDDeN+NeRF
MBRS+NeRF
NeRF with message
CopyRNeRF in geometry
Accuracy 100% / PSNR 30.28
Accuracy 50.25% / PSNR 27.75
Accuracy 51.38% / PSNR 29.09
Accuracy 63.19% / PSNR 20.26
Accuracy 68.00% / PSNR 17.61
Accuracy 100% / PSNR 32.69
Accuracy 50.94% / PSNR 27.71
Accuracy 51.04% / PSNR 28.61
Accuracy 69.28% / PSNR 22.83
Accuracy 61.50% / PSNR 18.82

Figure 3: Visual quality comparisons of each baseline. We show the differences ($\times 10$) between the synthesized results and the ground truth next to each method. Our proposed CopyRNeRF can achieve a well balance between the reconstruction quality and bit accuracy.

Table 1: Bit accuracies with different lengths compared with baselines. The results are averaged on all all examples.

	4 bits	8 bits	16 bits	32 bits	48 bits
Proposed CopyRNeRF	**100%**	**100%**	**91.16%**	**78.08%**	**60.06%**
HiDDeN [45]+NeRF[23]	50.31%	50.25%	50.19%	50.11%	50.04%
MBRS [14]+NeRF [23]	53.25%	51.38%	50.53%	49.80%	50.14%
NeRF[23] with message	72.50%	63.19%	52.22%	50.00%	51.04%
CopyRNeRF in geometry	76.75%	68.00%	60.16%	54.86%	53.36%

5. Experiments

5.1. Experimental settings

Dataset. To evaluate our methods, we train and test our model on Blender dataset [23] and LLFF dataset [22], which are common datasets used for NeRF. Blender dataset contains 8 detailed synthetic objects with 100 images taken from virtual cameras arranged on a hemisphere pointed inward. As in NeRF [23], for each scene we input 100 views for training. LLFF dataset consists of 8 real-world scenes that contain mainly forward-facing images. Each scene contains 20 to 62 images. The data split for this dataset also follows NeRF [23]. For each scene, we select 20 images from their testing dataset to evaluate the visual quality. For the evaluation of bit accuracy, we render 200 views for each scenes to test whether the message can be effectively extracted under different viewpoints. We report average values across all testing viewpoints in our experiments.

Baselines. To the best of our knowledge, there is no method specifically for protecting the copyright of NeRF models. We, therefore, compare with four strategies to guarantee a fair comparison: 1) **HiDDeN [50]+NeRF[23]**: processing images with classical 2D watermarking method HiDDeN [50] before training the NeRF model; 2) **MBRS [14]+NeRF [23]**: processing images with state-of-the-art 2D watermarking method MBRS [14] before training the NeRF model; 3) **NeRF with message**: concatenating the message $\mathbf{M}$ with location $\mathbf{x}$ and viewing direction $\mathbf{d}$ as the input of NeRF; 4) **CopyRNeRF in geometry**: changing our CopyRNeRF by fusing messages with the geometry to evaluate whether geometry is a good option for message embedding.

Evaluation methodology. We evaluate the performance of our proposed method against other methods by following the standard of digital watermarking about the invisibility, robustness, and capacity. For *invisibility*, we evaluate the performance by using PSNR, SSIM, and LPIPS [47] to compare the visual quality of the rendered results after message embedding. For *robustness*, we will investigate whether the encoded messages can be extracted effectively by measuring the bit accuracy on different distortions. Besides normal situations, we consider the following distortions for message extraction: 1) Gaussian noise, 2) Rotation, 3) Scaling, and 4) Gaussian blur. For *capacity*, following the setting in previous work for the watermarking of explicit 3D models [43], we investigate the invisibility and robustness under different message length as $N_b \in \{4, 8, 16, 32, 48\}$, which has been proven effective in protecting 3D models [43]. Since we have included different viewpoints in our experiments for each scene, our evaluation can faithfully reflect whether the evaluated method can guarantee its robustness and consistency across viewpoints.

5.2. Experimental results

Qualitative results. We first compare the reconstruction quality visually against all baselines and the results are shown in Figure 3. Actually, all methods except NeRF with message and CopyRNeRF in geometry can achieve high reconstruction quality. For HiDDeN [50] + NeRF [23]

Table 2: Bit accuracies and reconstruction qualities compared with our baselines. ↑ (↓) means higher (lower) is better. We show the results on $N_b = 16$ bits. The results are averaged on all all examples. The best performances are highlighted in **bold**.

	Bit Acc↑	PSNR ↑	SSIM ↑	LPIPS ↓
Proposed CopyRNeRF	**91.16%**	26.29	0.910	0.038
HiDDeN [50]+NeRF[23]	50.19%	26.53	0.917	0.035
MBRS [14]+NeRF [23]	50.53%	**28.79**	**0.925**	**0.022**
NeRF with message	52.22%	22.33	0.773	0.108
CopyRNeRF in geometry	60.16%	20.24	0.771	0.095

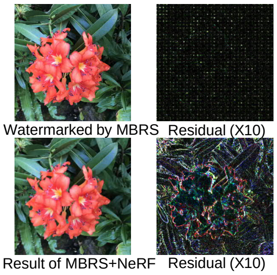

Figure 4: Analysis for failure of MBRS [14]+NeRF.

Table 3: Bit accuracies with different distortion types compared with each baseline and our CopyRNeRF without distortion-resistant rendering (DRR). We show the results on $N_b = 16$ bits. The results are averaged on all all examples.

	No Distortion	Gaussian noise	Rotation	Scaling	Gaussian blur
		(ν=0.1)	($\pm\pi/6$)	($\leq 25\%$)	($deviation = 0.1$)
Proposed CopyRNeRF	91.16%	**90.44%**	**88.13%**	**89.33%**	**90.06%**
HiDDeN [50]+NeRF[23]	50.19%	49.84%	50.12%	50.09%	50.16%
MBRS [14]+NeRF [23]	50.53%	51.00%	51.03%	50.12%	50.41%
NeRF with message	52.22%	50.53%	50.22%	50.19%	51.34%
CopyRNeRF in geometry	60.16%	58.00%	56.94%	60.09%	59.38%
CopyRNeRF W/o DRR	**91.25%**	89.12%	75.81%	87.44%	87.06%

and MBRS [14]+NeRF [23], although they are efficient approaches in 2D watermarking, their bit accuracy values are all low for rendered images, which proves that the message are not effectively embedded after NeRF model training. From the results shown in Figure 4, the view synthesis of NeRF changes the information embedded by 2D watermarking methods, leading to their failures. For NeRF with message, as assumed in our previous discussions, directly employing secret messages as an input change the appearance of the output, which leads to their lower PSNR values. Besides, its lower bit accuracy also proves that this is not an effective embedding scheme. For CopyRNeRF in geometry, it achieves the worst visual quality among all methods. The rendered results look blurred, which confirms our assumption that the geometry is not a good option for message embedding.

Bit Accuracy vs. Message Length. We launch 5 experiments for each message length and show the relationship between bit accuracy and the length of message in Table 1. We could clearly see that the bit accuracy drops when the number of bits increases. However, our CopyRNeRF achieves the best bit accuracy across all settings, which proves that the messages can be effectively embedded and robustly extracted. CopyRNeRF in geometry achieves the second best results among all setting, which shows that embedding message in geometry should also be a potential option for watermarking. However, the higher performance of our proposed CopyRNeRF shows that color representation is a better choice.

Bit Accuracy vs. Reconstruction Quality. We conduct more experiments to evaluate the relationship between bit accuracy and reconstruction quality. The results are shown in Table 2[1]. Our proposed CopyRNeRF achieves a good balance between bit accuracy and error metric values. Though the visual quality values are not the highest, the bit accuracy is the best among all settings. Though HiDDeN [50] + NeRF [23] and MBRS [14]+NeRF [23] can produce better visual quality values, its lower bit accuracy indicates that the secret messages are not effectively embedded and robustly extracted. NeRF with message also achieves degraded performance on bit accuracy, and its visual quality values are also low. It indicates that the embedded messages undermine the quality of reconstruction. Specifically, the lower visual quality values of CopyRNeRF in geometry indicates that hidding messages in color may lead to better reconstruction quality than hiding messages in geometry.

Model robustness on 2D distortions. We evaluate the robustness of our method by appyling several traditional 2D distortions. Specifically, as shown in Table 3, we consider several types of 2D distortions including noise, rotation, scaling, and cropping. We could see that our method is quite robust to different 2D distortions. Specifically, CopyRNeRF w/o DRR achieves similar performance to the complete CopyRNeRF when no distortion is encountered. How-

[1] Results for other lengths of raw bits can be found in the supplementary materials.

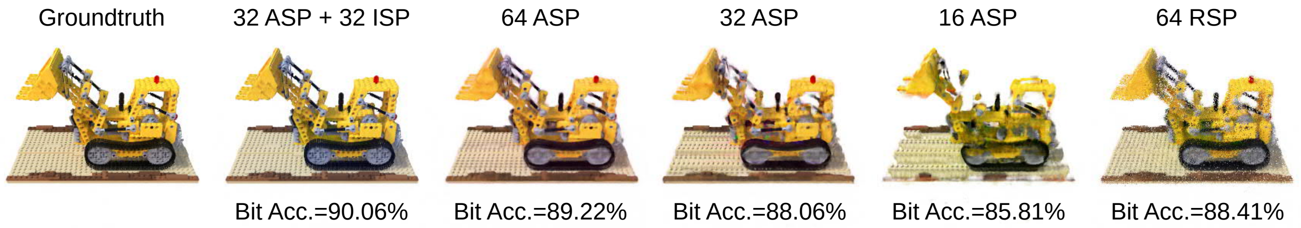

Figure 5: Comparisons for different rendering degradadtion in the inference phase. The message length is set to 16. We use average sampling points (ASP), importance sampling points (ISP), and random sampling points (RSP) in different rendering strategies. "32 ASP + 32 ISP" is a strategy employed in the training process, and message extraction also shows the highest bit accuracy. When sampling strategies are changed to other ones during inference, the message extraction still shows similar performance, which verifies the effectiveness of our distortion-resistant rendering.

Table 4: Comparisons for our full model, our model without Message Feature Field (MFF) and our model without Color Feature Field (CFF). The last row shows that our method achieves consistent performance even when different rendering scheme (DRS) is applied during testing.

	Bit Acc↑	PSNR ↑	SSIM ↑	LPIPS ↓
Ours	**100%**	**32.68**	**0.948**	**0.048**
W/o MFF	82.69%	20.46	0.552	0.285
W/o CFF	80.69%	21.06	0.612	0.187
DRS	**100%**	32.17	0.947	0.052

ever, when it comes to different distortions, its lower bit accuracies demonstrates the effectiveness of our distortion-resistant rendering during training.

Analysis for feature field. In the section, we further evaluate the effectiveness of color feature field and message feature field. We first remove the module for building color feature field and directly combine the color representation with the message features. In this case, the model performs poorly in preserving the visual quality of the rendered results. We further remove the module for building message feature field and combine the message directly with the color feature field. The results in Table 4 indicate that this may result in lower bit accuracy, which proves that messages are not embedded effectively.

Model robustness on rendering. Though we apply a normal volume rendering strategy for inference, the messages can also be effectively extracted using the a distortion rendering utilized in training phase. As shown in the last row of Table 4, the quantitative values with the distortion rendering are still similar to original results in the first row of Table 4, which further confirms the robustness of our proposed method.

The results for different sampling schemes are presented in Figure 5. Our distortion-resistant rendering employs 32 average sampling points and 32 importance sampling points during training. When different sampling strategies are applied in the inference phase, our method can also achieve high bit accuracy, which can validate the robustness of our method referring to different sampling strategies.

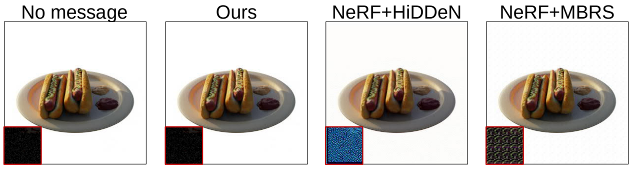

Figure 6: Comparisons for watermarking after rendering. The patch in the lower left corner shows the augmentation result by simply multiplying a factor 30. We use image inversion for better visualization

Compasion with NeRF+HiDDeN/MBRS [50, 14]. We also conduct an experiment to compare the our method with approaches by directy applying 2D watermaring method on rendered images, namely NeRF+HiDDeN [50] and NeRF+MBRS [14]. Although these methods can reach a high bit accuracy as reported in their papers, as shown in Figure 6, these methods can easily leave detectable traces especially in areas with lower geometry values, as they lack the consideration for 3D information during watermarking. Besides, they only consider the media in 2D domain and cannot protect the NeRF model weights.

6. Conclusions

In this paper, we propose a framework to create a copyright-embedded 3D implicit representation by embedding messages into model weights. In order to guarantee the invisibility of embedded information, we keep the geometry unchanged and construct a watermarked color representation to produce the message embedded color. The embedded message can be extracted by a CNN-based extractor from rendered images from any viewpoints, while keeping high reconstruction quality. Additionally, we introduce a distortion-resistant rendering scheme to enhance the robustness of our model under different types of distortion, including classical 2D degradation and different rendering strategies. The proposed method achieves a promising balance between bit accuracy and high visual quality in experimental evaluations.

Limitations. Though our method has shown promising performance in claiming the ownership of Neural Radiance Fields, training a NeRF model is time-consuming. We will consider how to speed up the training process in our future work. Besides, though we have considered several designs to strengthen the system robustness, this standard may still be undermined when malicious users directly attack model weights, *ie*, the model weights are corrupted. We conduct a simple experiment by directly adding Gaussian noise (std = 0.01) to the model parameters, and the accuracy slightly decreases to 93.97% ($N_b = 8$). As this may also affect rendering quality, such model weights corruption may not be a priority for malicious users who intend to display the content. We will still actively consider how to handle such attacks in our future work.

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

References

[1] Mahdi Ahmadi, Alireza Norouzi, Nader Karimi, Shadrokh Samavi, and Ali Emami. Redmark: Framework for residual diffusion watermarking based on deep networks. *Expert Systems with Applications*, 2020.

[2] Shumeet Baluja. Hiding images in plain sight: Deep steganography. In *Advances in Neural Information Processing Systems*, 2017.

[3] Shumeet Baluja. Hiding images within images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019.

[4] Eric R Chan, Connor Z Lin, Matthew A Chan, Koki Nagano, Boxiao Pan, Shalini De Mello, Orazio Gallo, Leonidas J Guibas, Jonathan Tremblay, Sameh Khamis, et al. Efficient geometry-aware 3D generative adversarial networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022.

[5] Eric R Chan, Marco Monteiro, Petr Kellnhofer, Jiajun Wu, and Gordon Wetzstein. pi-GAN: Periodic implicit generative adversarial networks for 3D-aware image synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021.

[6] Anpei Chen, Zexiang Xu, Fuqiang Zhao, Xiaoshuai Zhang, Fanbo Xiang, Jingyi Yu, and Hao Su. MVSNeRF: Fast generalizable radiance field reconstruction from multi-view stereo. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021.

[7] Pei-Ze Chiang, Meng-Shiun Tsai, Hung-Yu Tseng, Wei-Sheng Lai, and Wei-Chen Chiu. Stylizing 3D scene via implicit representation and hypernetwork. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2022.

[8] Daniel Cotting, Tim Weyrich, Mark Pauly, and Markus Gross. Robust watermarking of point-sampled geometry. In *Proceedings Shape Modeling Applications, 2004.*, 2004.

[9] Sara Fridovich-Keil, Alex Yu, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. Plenoxels: Radiance fields without neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022.

[10] Mohamed Hamidi, Aladine Chetouani, Mohamed El Haziti, Mohammed El Hassouni, and Hocine Cherifi. Blind robust 3D mesh watermarking based on mesh saliency and wavelet transform for copyright protection. *Information*, 2019.

[11] Jong-Uk Hou, Do-Gon Kim, and Heung-Kyu Lee. Blind 3D mesh watermarking for 3D printed model by analyzing layering artifact. *IEEE Transactions on Information Forensics and Security*, 2017.

[12] Yi-Hua Huang, Yue He, Yu-Jie Yuan, Yu-Kun Lai, and Lin Gao. StylizedNeRF: Consistent 3D scene stylization as stylized NeRF via 2D-3D mutual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022.

[13] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, 2015.

[14] Zhaoyang Jia, Han Fang, and Weiming Zhang. MBRS: Enhancing robustness of DNN-based watermarking by mini-batch of real and simulated jpeg compression. In *Proceedings of the 29th ACM International Conference on Multimedia*, 2021.

[15] Junpeng Jing, Xin Deng, Mai Xu, Jianyi Wang, and Zhenyu Guan. HiNet: deep image hiding by invertible network. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021.

[16] Ki-Ryong Kwon, Seong-Geun Kwon, Suk-Hawn Lee, Tae-Su Kim, and Kuhn-Il Lee. Watermarking for 3D polygonal meshes using normal vector distributions of each patch. In *Proceedings 2003 International Conference on Image Processing*, 2003.

[17] Chih-Chin Lai and Cheng-Chih Tsai. Digital image watermarking using discrete wavelet transform and singular value decomposition. *IEEE Transactions on Instrumentation and Measurement*, 2010.

[18] David B Lindell, Julien NP Martel, and Gordon Wetzstein. Autoint: Automatic integration for fast neural volume rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021.

[19] Jing Liu, Yajie Yang, Douli Ma, Wenjuan He, and Yinghui Wang. A novel watermarking algorithm for three-dimensional point-cloud models based on vertex curvature. *International Journal of Distributed Sensor Networks*, 2019.

[20] Shaohui Liu, Yinda Zhang, Songyou Peng, Boxin Shi, Marc Pollefeys, and Zhaopeng Cui. DIST: Rendering deep implicit signed distance function with differentiable sphere tracing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020.

[21] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks:

Learning 3D reconstruction in function space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019.

[22] 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)*, 2019.

[23] 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. In *Proceedings of the European Conference on Computer Vision*, 2020.

[24] Thomas Neff, Pascal Stadlbauer, Mathias Parger, Andreas Kurz, Joerg H. Mueller, Chakravarty R. Alla Chaitanya, Anton S. Kaplanyan, and Markus Steinberger. DONeRF: Towards Real-Time Rendering of Compact Neural Radiance Fields using Depth Oracle Networks. *Computer Graphics Forum*, 2021.

[25] Michael Niemeyer, Lars Mescheder, Michael Oechsle, and Andreas Geiger. Differentiable volumetric rendering: Learning implicit 3D representations without 3D supervision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020.

[26] Ryutarou Ohbuchi, Akio Mukaiyama, and Shigeo Takahashi. A frequency-domain approach to watermarking 3D shapes. In *Computer Graphics Forum*, 2002.

[27] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. DeepSDF: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019.

[28] Songyou Peng, Michael Niemeyer, Lars Mescheder, Marc Pollefeys, and Andreas Geiger. Convolutional occupancy networks. In *Proceedings of the European Conference on Computer Vision*, 2020.

[29] Emil Praun, Hugues Hoppe, and Adam Finkelstein. Robust mesh watermarking. In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques*, 1999.

[30] Shunsuke Saito, Zeng Huang, Ryota Natsume, Shigeo Morishima, Angjoo Kanazawa, and Hao Li. Pifu: Pixel-aligned implicit function for high-resolution clothed human digitization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019.

[31] Katja Schwarz, Yiyi Liao, Michael Niemeyer, and Andreas Geiger. GRAF: Generative radiance fields for 3D-aware image synthesis. *Advances in Neural Information Processing Systems*, 2020.

[32] Vincent Sitzmann, Michael Zollhöfer, and Gordon Wetzstein. Scene representation networks: Continuous 3D-structure-aware neural scene representations. *Advances in Neural Information Processing Systems*, 2019.

[33] Jeongho Son, Dongkyu Kim, Hak-Yeol Choi, Han-Ul Jang, and Sunghee Choi. Perceptual 3D watermarking using mesh saliency. In *International Conference on Information Science and Applications*, 2017.

[34] Matthew Tancik, Ben Mildenhall, and Ren Ng. Stegastamp: Invisible hyperlinks in physical photographs. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020.

[35] R.G. van Schyndel, A.Z. Tirkel, and C.F. Osborne. A digital watermark. In *Proceedings of 1st International Conference on Image Processing*, 1994.

[36] Can Wang, Ruixiang Jiang, Menglei Chai, Mingming He, Dongdong Chen, and Jing Liao. NeRF-Art: Text-driven neural radiance fields stylization. *arXiv preprint arXiv:2212.08070*, 2022.

[37] Xinyu Weng, Yongzhi Li, Lu Chi, and Yadong Mu. High-capacity convolutional video steganography with temporal residual modeling. In *Proceedings of the International Conference on Multimedia Retrieval*, 2019.

[38] Eric Wengrowski and Kristin Dana. Light field messaging with deep photographic steganography. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019.

[39] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3D ShapeNets: A deep representation for volumetric shapes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

[40] Yinghao Xu, Sida Peng, Ceyuan Yang, Yujun Shen, and Bolei Zhou. 3D-aware image synthesis via learning structural and textural representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022.

[41] Peng Yang, Yingjie Lao, and Ping Li. Robust watermarking for deep neural networks via bi-level optimization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021.

[42] Lior Yariv, Yoni Kasten, Dror Moran, Meirav Galun, Matan Atzmon, Basri Ronen, and Yaron Lipman. Multiview neural surface reconstruction by disentangling geometry and appearance. *Advances in Neural Information Processing Systems*, 2020.

[43] Innfarn Yoo, Huiwen Chang, Xiyang Luo, Ondrej Stava, Ce Liu, Peyman Milanfar, and Feng Yang. Deep 3D-to-2D watermarking: Embedding messages in 3D meshes and extracting them from 2D renderings. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022.

[44] Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. PixelNeRF: Neural radiance fields from one or few images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021.

[45] Chaoning Zhang, Philipp Benz, Adil Karjauv, Geng Sun, and In So Kweon. Udh: Universal deep hiding for steganography, watermarking, and light field messaging. *Advances in Neural Information Processing Systems*, 2020.

[46] Kevin Alex Zhang, Lei Xu, Alfredo Cuesta-Infante, and Kalyan Veeramachaneni. Robust invisible video watermarking with attention. *arXiv preprint arXiv:1909.01285*, 2019.

[47] 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*, 2018.

[48] Chengxuan Zhu, Renjie Wan, and Boxin Shi. Neural transmitted radiance fields. In *Advances in Neural Information Processing Systems*, 2022.

[49] Chengxuan Zhu, Renjie Wan, Yunkai Tang, and Boxin Shi. Occlusion-free scene recovery via neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023.

[50] Jiren Zhu, Russell Kaplan, Justin Johnson, and Li Fei-Fei. HiDDeN: Hiding data with deep networks. In *Proceedings of the European Conference on Computer Vision*, 2018.