StegaNeRF: Embedding Invisible Information within Neural Radiance Fields

Chenxin Li1*, Brandon Y. Feng2*, Zhiwen Fan3*, Panwang Pan4, Zhangyang Wang3
1Chinese University of Hong Kong, 2University of Maryland, 3University of Texas at Austin, 4ByteDance

Abstract

Recent advancements in neural rendering have paved the way for a future marked by the widespread distribution of visual data through the sharing of Neural Radiance Field (NeRF) model weights. However, while established techniques exist for embedding ownership or copyright information within conventional visual data such as images and videos, the challenges posed by the emerging NeRF format have remained unaddressed. In this paper, we introduce StegaNeRF, an innovative approach for steganographic information embedding within NeRF renderings. We have meticulously developed an optimization framework that enables precise retrieval of hidden information from images generated by NeRF, while ensuring the original visual quality of the rendered images to remain intact. Through rigorous experimentation, we assess the efficacy of our methodology across various potential deployment scenarios. Furthermore, we delve into the insights gleaned from our analysis. StegaNeRF represents an initial foray into the intriguing realm of infusing NeRF renderings with customizable, imperceptible, and recoverable information, all while minimizing any discernible impact on the rendered images. For more details, please visit our project page: https://xggnet.github.io/StegaNeRF/

1. Introduction

Implicit neural representation (INR) is an emerging concept where the network describes the data through its weights [14, 24, 40, 42, 45, 58, 65]. After training, the INR weights can then be used for content distribution, streaming, and even downstream inference tasks, all without sending or storing the original data. Arguably the most prominent INR is Neural Radiance Fields (NeRF) [42], where a network learns a continuous function mapping spatial coordinates to density and color. Due to its lightweight size and superb quality, NeRF has immense potential for 3D content representation in future vision and graphics applications.

*Equal contribution

While there is a plethora of work dedicated towards better quality [4, 63, 72, 91], faster rendering [43, 56, 61, 71, 83], and sparse view reconstruction [10, 11, 44, 57, 80, 81, 85, 88], in this paper, we look beyond the horizon and explore a new question: Can we achieve steganography with NeRF?

Established digital steganography method [12] focus on embedding hidden messages in 2D images. The recent growth of deep learning and social media platforms further gives rise to many practical use cases of image steganography. As countless images and videos are shared online and even used to train deep learning models, 2D steganography methods [2, 3] allow users and data providers to protect copyright, embed ownership, and prevent content abuse.

Now, with the ongoing advances in 3D representations powered by NeRF, we envision a future where people frequently share their captured 3D content online just as they are currently sharing 2D images and videos online. Moreover, we are curious to explore the following research questions: 1. Injecting information into 2D images for copyright or ownership identification is common, but can we preserve such information when people share and render 3D scenes through NeRF? 2. NeRF can represent large-scale real-world scenes with training images taken by different people, but can we preserve these multiple source identities in the NeRF renderings to reflect the collaborative efforts required to reconstruct these 3D scenes? 3. Common image steganography methods embed either a hidden image or a message string into a given image, but can we allow different modalities of the hidden signal in NeRF steganography?
Driven by these questions, we formulate a framework to embed customizable, imperceptible, and recoverable information in NeRF renderings without sacrificing the visual quality. Fig. 1 presents an overview of our proposed framework, dubbed StegaNeRF. Unlike the traditional image steganography that embeds hidden signals only into a specific source image, we wish to recover the same intended hidden signal from NeRF rendered at arbitrary viewpoints.

Despite many established works on 2D steganography and hidden watermarks for image and video [2, 3, 9, 25, 36, 64], naively applying 2D steganography on the NeRF training images is not practical, since the embedded information easily gets lost in the actual NeRF renderings. In contrast, our framework enables reliable extraction of the hidden signal from NeRF renderings. During NeRF training, we jointly optimize a decoder network to extract hidden information from the NeRF renderings. To minimize the negative impact on the visual quality of NeRF, we identify weights with low impact on rendering and introduce a gradient masking strategy to steer the hidden steganographic information towards those low-importance weights. Extensive experimental results validate StegaNeRF balances between the rendering quality of novel views and the high-fidelity transmission of the concealed information.

StegaNeRF presents the first exploration of hiding information in NeRF models for ownership identification, and our contributions can be summarized as follows:

- We introduce the new problem of NeRF steganography and present the first effort to embed customizable, imperceptible, and recoverable information in NeRF.
- We propose an adaptive gradient masking strategy to steer the injected hidden information towards the less important NeRF weights, balancing the objectives of steganography and rendering quality.
- We empirically validate the proposed framework on a diverse set of 3D scenes with different camera layouts and scene characteristics, obtaining high recovery accuracy without sacrificing rendering quality.
- We explore various scenarios applying StegaNeRF for ownership identification, with the additional support of multiple identities and multi-modal signals.

3. Related Work

Neural Radiance Fields Following the success of NeRF [42], numerous techniques have been incorporated including ray re-parameterizations [4, 5], explicit spatial data structures [16, 23, 34, 43, 61, 84], caching and distillation [19, 22, 54, 69], ray-based representations [1, 15, 59], geometric primitives [33, 35], large-scale scenes [39, 63, 79, 89], generalizable architectures [60, 68, 73], and dynamic settings [18, 32, 46, 51, 78]. Unlike prior works, this paper explores the uncharted problem of embedding information in NeRF renderings, with critical implications for copyright protection and ownership preservation. As early-stage NeRF-based products already become available [37, 49] and more are expected, we believe now is the right time to open up the exploration of NeRF steganography.

Image Steganography Steganography hides intended signals as invisible watermarks (e.g., hyperlinks, images) within the cover media called carriers (e.g., images, video and audio) [12, 26]. Classical methods focus on seeking and altering the least significant bits (LSBs) [17, 47, 62, 76] and transforming domain techniques [7, 8, 52, 70]. Prior research also uses deep neural networks for steganography [2, 3, 21, 66]. Among them, DeepStega [2] conceals a hidden image within an equal-sized carrier image. Subsequent works [25, 36] use invertible networks to improve the performance of deep image steganography. Another line of work conceals information in other carrier media like audio [13, 20, 82] and video [38, 55, 75]. The above advances all play a critical part in the era when traditional media formats like images and videos are dominant. However, as MLP-based neural representations of 3D scenes are gaining momentum to become a major format of visual data, extending steganography to NeRF is bound to become an important problem in the upcoming future.

Lifting Steganography to 3D Prior to NeRF, meshes are commonly used to represent 3D shapes. Early pioneers apply steganography 3D meshes [6, 50] for copyright protection when meshes are exchanged and edited. More recent work [77] has also explored embedding multi-plane images within a JPEG-compressed image. Their problem can be regarded as a special case of 2D steganography, hiding multiple 2D images inside a single 2D image. In contrast, we try to hide a natural image into a 3D scene representation (NeRF), fundamentally differing from these prior methods where 2D images act as the carrier of hidden information.

3. Method

The goal of StegaNeRF is to inject customized (steganographic) information into the NeRF weights with imperceptible visual changes when rendering. Given a NeRF with weights θ and the information I to be injected, when we render with θ on any viewpoint, we hope the injected I can be recovered by a decoder Fψ with learnable weights ψ.

A seemingly obvious solution is to use prior image steganography methods to 1) inject I on training images, 2) train NeRF on those images with embedded information, 3) apply their provided Fψ to extract I from NeRF-rendered images. This approach has been successfully applied for GAN fingerprinting [86]. However, it fails in the NeRF context (see Fig. 3), where the subtle changes induced by prior
steganography methods easily get smoothed out, inhibiting the decoder from identifying the subtle patterns necessary for information recovery.

Such shortcomings of off-the-shelf steganographic methods are not surprising since they are developed for the traditional setting where 2D images are the ultimate form of visual information. In contrast, our problem setting involves the new concept of INR as the underlying representation, and 2D images are just the final output rendered by NeRF.

3.1. Two-Stage Optimization

Recognizing the formulation difference due to the emergence of implicit representation with NeRF, we move away from the traditional 2D image-based pipeline that trains an encoder network to inject subtle changes to the given 2D images. Instead, we incorporate the steganography objective into the gradient-based learning process of NeRF.

We re-design the training as a two-stage optimization. The first stage is the original NeRF training procedure, involving the standard photometric loss between the rendered and ground truth pixels to guarantee the visual quality. After finishing the weights update \( \theta_0 \) at the first stage, we dedicate the second stage to obtain the final NeRF weights \( \theta \) containing steganographic information. We introduce several techniques to achieve robust information recovery with imperceptible impact on the rendered images. The training workflow is depicted in Fig. 2 and Alg. 1.

3.2. Information Recovery

Let \( P \) denote the camera pose at which we render an image from a NeRF network. We want to recover \( I \) from the rendered image \( \theta(P) \). Importantly, we also want to avoid false positives on images \( \theta_0(P) \) rendered by the original network without steganography ability, even if the images rendered by \( \theta_0 \) and \( \theta \) look visually identical. Therefore, we optimize \( \theta \) to minimize the following contrastive loss terms:

\[
L_{\text{dec}}^+ = |F_\psi(\theta(P)) - I|, \quad L_{\text{dec}}^- = |F_\psi(\theta_0(P)) - \emptyset|, \quad (1)
\]

where \( \emptyset \) is an empty image with all zeros. Effectively, \( L_{\text{dec}}^+ \) serves as a regularization term, facilitating the decoder to recover the embedded image pattern based on renderings obtained from a StegaNeRF. In contrast, \( L_{\text{dec}}^- \) prevents the decoder from falsely producing any plausible image pattern when given renderings from a Standard NeRF that is without hidden signals.

Classifier-Guided Recovery The decoder \( F_\psi \) is easily implemented as a U-Net to decode \( I \) as the form of 2D images, but accurately recovering all the details in \( I \) might be challenging. Therefore, we additionally train a classifier
network $F_θ$ to solve the easier task of classifying whether the given NeRF rendering contains hidden information. $F_θ$ is optimized by the following cross-entropy loss [30, 67]:

$$L_{dec} = -\log(F_θ(\theta(P))) - \log(1 - F_θ(\theta_0(P))).$$

We then use its prediction to guide the process of decoding pixel-wise information by adding the classification output as input to $F_ψ$, such that $F_ψ(x) = F_ψ(x, F_θ(\theta_0(x)))$.

Although the above discussion focuses on hiding images, our framework can be easily extended to embed other modalities like strings, text, or even audio, all of which may be represented as 1D vectors. We can simply modify the architecture of $F_ψ$ to have a 1D prediction branch.

### 3.3. Preserving Perceptual Identity

Since we want to hide information without affecting the visual perception of the rendered output, an intuitive regularization is to penalize how much $\theta$ deviates from $\theta_0$. However, we find that naively summing the deviation penalty across all weights makes it difficult for the NeRF network to adjust its weights for the steganographic objective. Instead, motivated by the fact that INR weights are not equally important and exhibit strong sparsity [28, 87], we propose an adaptive gradient masking strategy to encode the hidden information on specific weights’ groups.

Formally, given the initial set of weights $\theta_0 \in \mathbb{R}^N$, we compute a soft importance mask $m \in \mathbb{R}^N$. Specifically, for each weight $w_i \in \theta_0$, we compute:

$$m_i = \frac{|w_i|^{-\alpha}}{\sum_i |w_i|^{-\alpha}}.$$  

where the magnitude $| \cdot |$ of each model weight $w_i$ serves as its individual importance indicator, and $\alpha > 0$ is a scaling factor of power controlling the relative distribution of importance across the weights. We mask the gradient as $\frac{\partial \mathcal{L}}{\partial \theta} \odot m$ when optimizing $\theta$ based on the total loss $\mathcal{L}$ in the second stage, where $\odot$ is a Hadamard product. Effectively, more significant weights are “masked out” to minimize the impact of steganographic learning on the rendered visual quality.

We retain the standard photometric error in steganography learning to maintain NeRF rendering fidelity: $\mathcal{L}_{rgb} = (\theta(P) - \theta_0(P))$. The overall training loss at the second stage can be formulated as follows:

$$\mathcal{L} = \lambda_0 L_{dec} + \lambda_1 L_{dec}^+ + \lambda_2 L_{dec}^- + \lambda_3 L_{rgb}.$$  

### 4. Experiments

#### 4.1. Implementation Details.

**Dataset.** We use common datasets LLFF [41] and NeRF-Synthetic [42], with forward scenes \{flower, fern, fortress, room\} from LLFF and 360° scenes \{lego, drums, chair\} from NeRF-Synthetic. We further experiment on the Brandenburg Gate scene from NeRF-W dataset [39], with over 800 views of in-the-wild collected online.

**Training.** On LLFF and NeRF-Synthetic, we adopt Plenoxels [83] as the NeRF backbone architecture for efficiency. On NeRF-W, we use the available PyTorch implementation [53]. The first stage of training is performed according to the standard recipes of those implementations. We then perform the second stage of steganography training for 55 epochs unless otherwise noted. On LLFF and NeRF-W, we downsize the training images by 4 times following common practice, and we use the original size on NeRF-Synthetic.

For hyper-parameters in Eq. (4), we set the weight of NeRF reconstruction error $\lambda_3 = 1$ for all experiments. We set $\lambda_0 = 0.01, \lambda_1 = 0.5, \lambda_2 = 0.5$ for all scenes on LLFF dataset, and $\lambda_0 = 0.1, \lambda_1 = 1, \lambda_2 = 1$ for the scenes in NeRF-Synthetic, and $\lambda_0 = 0.05, \lambda_1 = 1, \lambda_2 = 1$ for NeRF-W. For Eq. (3), we set $\alpha = 3$ for all scenes. We run experiments on one NVIDIA A100 GPU.

**Evaluation.** We assume a typical authentication scenario shown in Alg. 2. We consider the recovery quality of the embedded information, including the metrics of classification accuracy (Acc.) of the classifier, and the structural sim-
Figure 4. Results on the NeRF-Synthetic dataset. Within each column, we show the StegaNeRF rendering, residual error from the initial NeRF rendering, and the recovered hidden image. We show the SSIM for the StegaNeRF renderings and the recovered hidden images.

Table 1. Quantitative results of StegaNeRF rendering and hidden information recovery. Standard NeRF is the initial NeRF $\theta_0$ with standard training (upper-bound rendering performance). Prior 2D steganography fails after NeRF training while StegaNeRF successfully embeds and recovers hidden information with minimal impact on the rendering quality. Results are averaged over the selected LLFF and NeRF-Synthetic scenes.

<table>
<thead>
<tr>
<th>Method</th>
<th>NeRF Rendering</th>
<th>Hidden Recovery</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PSNR ↑</td>
<td>SSIM ↑</td>
</tr>
<tr>
<td>Standard NeRF</td>
<td>28.15</td>
<td>0.8575</td>
</tr>
<tr>
<td>LSB [9]</td>
<td>28.13</td>
<td>0.8569</td>
</tr>
<tr>
<td>DeepStega [2]</td>
<td>26.94</td>
<td>0.8432</td>
</tr>
<tr>
<td>StegaNeRF (Ours)</td>
<td>28.13</td>
<td>0.8363</td>
</tr>
</tbody>
</table>

Table 2. Quantitative results of NeRF rendering and hidden information recovery. We consider two conditions, embedding each scene with a common hidden image (One-for-All) or a scene-specific hidden image (One-for-One). We report the quality difference compared to the single-scene settings as $\Delta_{SSIM}^{(10^{-2})}$. Results are averaged over the selected LLFF scenes.

<table>
<thead>
<tr>
<th>Setting</th>
<th>NeRF Rendering</th>
<th>Hidden Recovery</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PSNR↑</td>
<td>SSIM↑</td>
</tr>
<tr>
<td>One-for-All</td>
<td>24.99</td>
<td>0.8013</td>
</tr>
<tr>
<td>One-for-One</td>
<td>24.99</td>
<td>0.8016</td>
</tr>
<tr>
<td>One-for-All</td>
<td>27.90</td>
<td>0.8513</td>
</tr>
<tr>
<td>One-for-One</td>
<td>27.90</td>
<td>0.8515</td>
</tr>
<tr>
<td>One-for-All</td>
<td>30.27</td>
<td>0.8498</td>
</tr>
<tr>
<td>One-for-One</td>
<td>30.12</td>
<td>0.8480</td>
</tr>
</tbody>
</table>

Table 1. Quantitative results of StegaNeRF rendering and hidden information recovery. Standard NeRF is the initial NeRF $\theta_0$ with standard training (upper-bound rendering performance). Prior 2D steganography fails after NeRF training while StegaNeRF successfully embeds and recovers hidden information with minimal impact on the rendering quality. Results are averaged over the selected LLFF and NeRF-Synthetic scenes.

4.2. Case I: Embedding in a Single Scene.

We first explore the preliminary case of ownership identification on a specific NeRF scene. We select random images from ImageNet [27] as the hidden information to be injected to NeRF renderings.

Failure of 2D Baseline  Due to the lack of prior study on NeRF steganography, we consider a baseline from 2D image steganography by training NeRF from scratch with the watermarked images. We implement two off-the-shelf steganography methods including a traditional machine learning approach called Least Significant Bit (LSB [9]), and a deep learning pipeline as DeepStega [2]. An ideal case is that the embedded information can be recovered from the synthesized novel views, indicating the successful NeRF steganography. However, as can be seen in Fig. 3, the embedded information containing the hidden images cannot be recovered from the NeRF renderings. By analyzing the residual maps of training views (between GT training views and the watermarked) and novel views (between GT novel views and the actual rendering), we observe the subtle residuals to recover the hidden information are smoothed out in NeRF renderings, and similar failures occur on other datasets as well. Therefore, for the rest of our study, we mainly focus on analyzing our new framework that performs direct steganography on 3D NeRF.
Results Tab. 1 contains quantitative results on LLFF and NeRF-Synthetic scenes. While NeRF trained by 2D steganography methods hardly recovers the embedded information, StegaNeRF accurately recovers the hidden image with minimal impact on the rendering quality measured by PSNR. Fig. 4 provides qualitative results of StegaNeRF on three embedded images on the NeRF-Synthetic scenes. An interesting observation is the common regions where hidden information (high residual error) emerges in the renderings, e.g., the right rear side of lego and left handrail of chair. We also notice that, within each scene, these regions are persistent across multiple viewpoints.

4.3. Case II: Embedding in Multiple Scenes.

Settings We extend our steganography scheme to embed information within multi-scene settings at once. Specially, we use three LLFF scenes \{flower, fern, fortress\} to test the two sub-settings of multi-scene embedding with: (1) One-for-All, a common hidden image and (2) One-for-One scene-specific hidden images. All scenes share the same decoder \(F_{\psi}\) and classifier \(F_{\psi}^c\). The difference between the two sub-settings is the number of hidden images that \(F_{\psi}\) and \(F_{\psi}^c\) need to identify and recover. We sample one scene for every training epoch, and due to the increased data amount, we increase training epochs until convergence.

Results Tab. 2 provides quantitative results on the multi-scene setting. The performance drop compared to single-scene training is sometimes noticeable, but it is not surprising due to the inherent requirement of per-scene fitting for our NeRF framework. Fig. 5 presents visual comparisons of multi-scene steganography against single-scene scheme.


Settings Constructing NeRF of large-scale cultural landmarks is a promising application, and community contributions are crucial to form the training images. Since every NeRF prediction is indebted to some particular training images, it would be meaningful to somehow identify the contributing users in the rendered output. Specifically, we present a proof-of-concept based on the following scenario: Given a collection of user-uploaded training images, our task is to apply StegaNeRF to ensure the final NeRF renderings hide subtle information about the relevant identities whose uploaded images help generate the current rendering.

To simulate this multi-user scenario in the public NeRF-W dataset, we randomly select \(M\) anchor views with different viewing positions, and then find the \(K\) nearest neighbour views to each anchor to form their respective clusters. We set \(M = 3, K = 20\) in experiments. We assume a common contributor identity for each cluster, and we want the NeRF rendering to contain customized hidden information about those \(M\) contributors when we render within the spatial regions spanned by their own cluster. Thus, our classifier network \(F_{\psi}^c\) is modified to output \(M\)-class cluster predictions and another class for identity outside of the \(M\) clusters. Since the decoder should extract no information for views outside of those \(M\) clusters to prevent false positives, we also compute \(\mathcal{L}_{\text{dec}}^1\) and \(\mathcal{L}_{\text{dec}}^2\) for those poses.

Results We employ the same network backbone as NeRF-W [66] to handle in-the-wild images with different time and lighting effects. Fig. 6 presents qualitative results of embedding multiple identities in a collaborative large-scale NeRF.
Figure 6. Qualitative results of NeRF steganography on NeRF-W dataset. Within each block we show the StegaNeRF rendering, the initial NeRF rendering, the residual error between StegaNeRF and initial, and the recovered hidden image. At the bottom right, we provide the SSIM computed with their respective ground truth. “Unmarked Identity” denotes the remaining training views not related to any of the considered users, and the decoder does not recover meaningful information from renderings at those poses.

Figure 7. Multi-modal hidden information. We show StegaNeRF rendering, recovered hidden images, audio (waveform and spectrum), and text. Metrics for hidden information in each modality are labeled respectively. For the StegaNeRF rendering, we report both SSIM and the relative change from initial NeRF renderings.

4.5. Case IV: Embedding Multi-Modal Information.

Settings We further show the potential of StegaNeRF in embedding multi-modal hidden information, such as images, audio, and text. We modify the decoder network to build a modal-specific decoder for each modality.

Results Fig. 7 shows recovered multi-modal embedded signals in the trex and horns scenes from LLFF. Evidently, the StegaNeRF framework can easily extend to embed multi-modal information with high recovery performance without sacrificing the rendering quality.

4.6. Ablation Studies

The effect of removing each component of StegaNeRF is presented in Tab. 3. Standard NeRF uses the initial NeRF rendering. No Classifier removes the classifier $F^C$, while No Classifier Guided retains the classification task (hence impact on NeRF rendering) but does not condition the decoder on classifier output. No Gradient Masking removes the proposed masking strategy (Sec. 3.3), and No Soft Masking uses the binary mask $[29, 31]$ with a threshold of 0.5. Our StegaNeRF makes a good balance between the rendering quality and the decoding accuracy. Fig. 8 shows the visual impact of removing each component. It appears that when any component is removed, the performance drops accordingly, revealing the effectiveness of our design.

Fig. 9 shows the impact of varying gradient masking strategies. We can see a different degree of affected by ratio variation on hidden recovery and rendering quality. Fig. 10 reports the performance of StegaNeRF against the common perturbations including JPEG compression and Gaussian noise. The lines show the average accuracies across selected scenes and the shaded regions indicate the range of 0.5 standard deviation. It appears that the hidden recovery of StegaNeRF is robust to various JPEG compression ratios and Gaussian blur degradation.
Figure 8. Impact on visual quality when changing different components of the proposed StegaNeRF framework as in Tab. 3. We show the SSIM for the renderings and the recovered hidden images.

Figure 9. Analysis of gradient masking. (a) Varying $\alpha$ from Eq. (3). (b) Varying portions of masked out weights from the gradient updates at steganography learning. We provide the SSIM of rendered views (blue) and recovered hidden images (green).

Figure 10. Analysis of robustness over (a) JPEG compression and (b) Gaussian blur. We provide the SSIM of rendered views (blue) and recovered hidden images (green).

5. More Discussions

How useful is steganography for NeRF? Although NeRF-based 3D content has yet to become mainstream, we believe it will play a major future role not only for social platforms, but also for 3D vision research and applications. As people upload their personal NeRF models online for viewing purposes, NeRF steganography for ownership identification is apparently an important feature. Moreover, future 3D vision research will likely demand large-scale NeRF datasets of real-world images, and in this context, NeRF steganography can be a crucial tool for responsible and ethical uses of training data and deployed models.

Hidden or Memorized? StegaNeRF follows the standard setup of NeRF which prescribes per-scene training. Therefore, it may be reasonable to wonder whether the decoder is trained to “memorize” the hidden image without it really being injected into NeRF. To address this concern, we perform an experiment where the NeRF weights are frozen after the first stage and only update the decoder of StegaNeRF. If the decoder actually memorizes the image by itself, then we should expect good recovery performance under this setup. However, as shown in Tab. 4, with NeRF Frozen, we observe a significant performance degradation. Therefore, we can verify that the decoder alone cannot trivially memorize
the hidden image, and the information is actually hidden in the NeRF renderings. Furthermore, we extend StegaNeRF to generalize across different scenes and hidden images while sharing the same universal decoder. Fig. 11a shows the rendering and decoding quality of embedding the different numbers of hidden images to a scene (flower). Fig. 11b shows the performance of embedding a common image into a varying number of scenes from the LLFF dataset. While the performance slightly decreases in the generalized setting, it still remains at a comparable level.

Why not just share the rendered images instead of NeRF weights? Directly sharing the NeRF model gives end users the freedom to render at continuous viewpoints, which fundamentally differs from the traditional setting centered around discrete 2D image frames. It would not be realistic to assume people will stick with sharing 2D images instead of 3D weights, and therefore, it is necessary for the research community to see beyond 2D steganography and explore 3D NeRF steganography.

Robustness: As shown in Fig. 12, in addition to using a resolution of 128 × 128 for hidden images in most of our experiments, we also tested resolutions ranging from 96 × 96 to 224 × 224. We found that increasing the size of embedded images had no noticeable impact on the performance of StegaNeRF, with only a 0.6% drop in rendering PSNR and a 1.2% reduction in hidden SSIM. We also demonstrated the robustness of StegaNeRF to different sampling approaches during rendering, such as avoiding sampling in empty areas [43, 48]. Instead of considering all the points on a ray during rendering, we disregard the contribution of points to the integration along the entire ray if their density falls below a threshold, like 0.001. As shown in Tab. 4, replacing the conventional rendering strategy with No Empty Sampling has minimal effect on the performance of StegaNeRF, revealing our robustness to different rendering fashions.

![Figure 12. Ablation on the resolutions of embedded images by the average results over the selected scenes from the LLFF dataset.](image)

Limitations: The current implementation is limited by the computational efficiency of NeRF, since the steganographic training stage requires performing gradient descent updates through the NeRF model. Another limitation is that, the recovery accuracy of the injected hidden information varies across different 3D scenes. More systematic analysis is required to understand the underlying factors that make some scenes easier to hide information than others.

6. Conclusion

With the progression of NeRF-based content, now serving as a pivotal element in diverse 3D applications, it is incumbent upon the research community to proactively tackle the imminent requirement to embed information within NeRF models as they are exchanged and disseminated across online platforms. This study introduces StegaNeRF, a pioneering framework for the automated embedding of steganographic information within NeRF renderings. We outline the essential components essential for achieving this functionality, unveiling potential challenges and revealing insights that can be instrumental for subsequent developers. By shedding light on the prospect of ownership identification within the realm of NeRF, this paper underscores the need for increased attention and concerted efforts toward addressing related challenges and avenues.

References


[16] Guy Gafni, Justus Thies, Michael Zollhofer, and Matthias Wenxue Cui, Shaohui Liu, Feng Jiang, Yongliang Liu, and


[19] Stephan J Garbin, Marek Kowalski, Matthew Johnson, Jamie Brandon Yushan Feng and Amitabh Varshney. SIGNET: Guy Gafni, Justus Thies, Michael Zollhofer, and Matthias


[37] Luma AI. https://lumalabs.ai/. 2


