3D Snapshot: Invertible Embedding of 3D Neural Representations in a Single Image

Yuqin Lu, Bailin Deng *Member, IEEE*, Zhixuan Zhong, Tianle Zhang, Yuhui Quan, Hongmin Cai, Shengfeng He, *Senior Member, IEEE*

Abstract—3D neural rendering enables photo-realistic reconstruction of a specific scene by encoding discontinuous inputs into a neural representation. Despite the remarkable rendering results, the storage of network parameters is not transmission-friendly and not extendable to metaverse applications. In this paper, we propose an invertible neural rendering approach that enables generating an interactive 3D model from a single image (*i.e.*, 3D Snapshot). Our idea is to distill a pre-trained neural rendering model (*e.g.*, NeRF) into a visualizable image form that can then be easily inverted back to a neural network. To this end, we first present a neural image distillation method to optimize three neural planes for representing the original neural rendering model. However, this representation is noisy and visually meaningless. We thus propose a dynamic invertible neural network to embed this noisy representation into a plausible image representation of the scene. We demonstrate promising reconstruction quality quantitatively and qualitatively, by comparing to the original neural rendering model, as well as video-based invertible methods. On the other hand, our method can store dozens of NeRFs with a compact restoration network (5MB), and embedding each 3D scene takes up only 160KB of storage. More importantly, our approach is the first solution that allows embedding a neural rendering model into image representations, which enables applications like creating an interactive 3D model from a printed image in the metaverse.

Index Terms—Neural representations, invertible image processing.

1 Introduction

Reconstructing a 3D scene from a sparse set of input images is an essential problem in computer vision and graphics, with a wide range of applications like the metaverse, computer games, and content creation. Recent advances in 3D neural rendering, especially NeRF [1] and its variants [2], [3], enable photo-realistic synthesis of novel views in complex scenarios. The main principle of this line of research is to use a deep network to represent the geometry and appearance of the scene. Despite the impressive reconstruction quality, a deep neural network is not an ideal way for "storing" the scene in some applications. First, it often involves a large number of parameters, which is not transmission- and storage-friendly. In addition, the model itself is an imperceivable information carrier that needs to be connected with other visual forms (*e.g.*, QR code) in applications such as metaverse. Although previous works attempted to decompose [3] or compress [4] a NeRF into a lightweight representation, the derived tensor components cannot be used for display purposes, while the resulting data size is not optimal for large-scale storage.

In this paper, we propose an invertible neural rendering method to conceal the 3D scene in an image representation, to meet both the storage and visual perception requirements. We transform an arbitrary neural rendering model to a visualizable and invertible image form in two stages, *neural image distillation* and *3D snapshot embedding*. The former decomposes the input model into three neural planes, each representing the geometry and appearance in X, Y, and Z dimensions, respectively. The original neural rendering model can be easily reconstructed by the three neural planes, and we call the combination of them a *neural image*. As this intermediate representation is noisy and not visualizable, we further embed it into a 3D snapshot in the second stage. However, noisy images are known to be challenging for information hiding [5]. We resolve this problem by concealing information in the frequency domain, using a dynamic invertible neural network (DINN) where we inject dynamicity in two aspects. First, we set half of the output channels (3 out of 6) as constants to reduce the output data size, and dynamically update them for each batch to maximize its representation power for various scenes. Second, inspired by Dropout [6], we introduce random noises to the constant channels to tolerate the differences across 3D scenes. In this way, the remaining 3 channels form a 3D snapshot that preserves the 3D scene appearance, while being embedded with the neural image information.

Our method can transform an arbitrary neural rendering model into a 3D snapshot. Extensive experiments show that our inversion achieves comparable performance to the original model. Moreover, when compared with video-based invertible methods using continuous views as the input, our model shows promising results while being interactive. In addition, our method can serve as a data storage tool, enabling the storage of dozens of NeRF-based 3D representations

The work is supported by the Guangdong Natural Science Funds for Distinguished Young Scholar (No. 2023B1515020097) and the National Research Foundation Singapore under the AI Singapore Programme (No. AISG3-GV-2023-011). (Shengfeng He is the corresponding author.)

Yuqin Lu, Zhixuan Zhong, Tianle Zhang, Yuhui Quan, and Hongmin Cai are with the School of Computer Science and Engineering, South China University of Technology, Guangzhou, China; Yuqin Lu, Zhixuan Zhong, and Tianle Zhang are also with the School of Computing and Information Systems, Singapore Management University, Singapore. E-mail: spaceluyq@gmail.com, zxzhong20@gmail.com, terryjoy0111@gmail.com, csyhquan@scut.edu.cn, hmcai@scut.edu.cn.

Bailin Deng is with the School of Computer Science and Informatics, Cardiff University, UK. E-mail: DengB3@cardiff.ac.uk.

Shengfeng He is with the School of Computing and Information Systems, Singapore Management University, Singapore. E-mail: shengfenghe@smu.edu.sg.

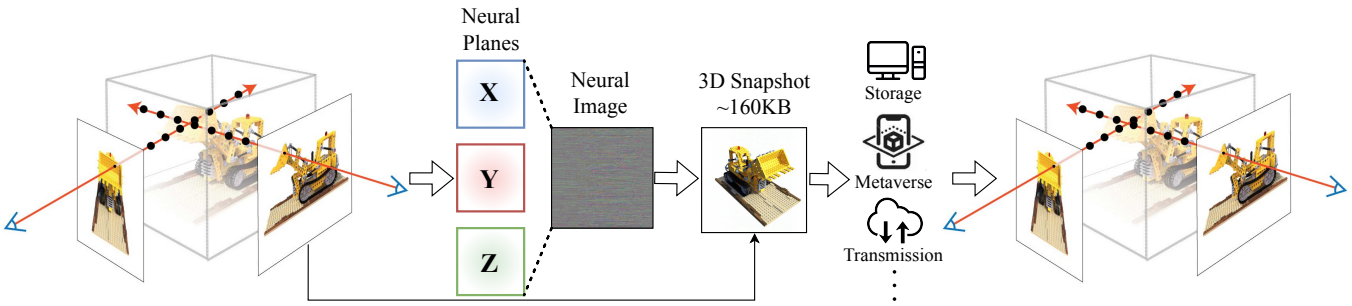


Fig. 1: We propose 3D Snapshot, an invertible neural rendering carrier that allows accurate 3D recomposition from a single image. We can convert an arbitrary neural rendering model into a distilled *neural image* representation, such that it can be effectively embedded into a small (around 160 KB) and visualizable 3D snapshot which allows inverting back to a 3D model.

using a compact network that only requires approximately 5MB of storage. This allows each scene to be represented as a 3D snapshot of 160 KB, approximately 25 times smaller than the state-of-the-art lightweight NeRF. Finally, our 3D snapshot can be easily printed on paper or on a wall to facilitate the reconstruction of an interactive 3D model in the metaverse. Fig. 1 illustrates the pipeline of our method. In summary, our contribution is threefold:

- We propose the first invertible neural rendering method that allows reconstructing an interactive 3D model from a single image.
- We present a neural image distillation method to approximate an arbitrary neural rendering model, and a dynamic invertible neural network that can embed the noisy intermediate result into a visualizable image.
- Our method can serve as a data storage tool and encode 50 scenes as snapshots (160K each) stored in a general network (5MB), reducing the storage requirement by up to 25× compared to existing lightweight NeRFs. Our snapshots provide visualizable carriers of the scenes that can be used to reconstruct the original 3D scenes.

2 Related Work

Neural Rendering and Scene Representations. Earlier works on neural rendering use neural networks to map continuous spatial coordinates to implicit shape representations, such as the signed distance [7], [8] or occupancy values [9], [10]. Recently, Neural Radiance Fields (NeRF) [1] represent a 3D scene implicitly as a continuous 5D function and apply volume rendering for view synthesis. Despite its high-quality rendering results and compactness, NeRF and many follow-up works [11]–[13] suffer from slow rendering and reconstruction due to their pure MLP-based design. Hence, different methods have been proposed to reduce the MLP computation by storing 3D features in an explicit data structure, such as voxel grids [14], [15], octrees [2], point clouds [16], triplanes [17], and multi-scale hashmap [18]. Unfortunately, this computational efficiency comes at the cost of large memory footprints for storing the features.

To reduce the model size while maintaining high computational efficiency, TensoRF [3] models the scene as a 4D tensor and factorizes it into low-rank components, dramatically lowering the memory consumption. CCNeRF [4] extends this work by further enabling extra flexible control of compression to trade off between the model size and the rendering quality. This compressibility is achieved by low-rank approximation of the scene tensor with a rank-residual learning strategy. In this paper, we also leverage the idea of factorization and compression, but go even further to compress a scene into a visualizable image that achieves almost 25-fold decrease of storage.

Steganography and Invertible Image Processing. Steganography involves concealing confidential data into a host acting as information carrier. Traditional techniques include utilizing the Least Significant Bits (LSB) [19], pixel value differencing (PVD) [20], histogram shifting [21], multiple bit-planes [22], and palettes [23] to hide images. In addition, deep learning models [24]–[29] have gained prominence for achieving enhanced performance and bolstering anti-steganalysis capabilities. Rather than concealing, invertible image processing transforms visual content (such as high-resolution images, dual-view images, and short videos) into an embedding image to facilitate data storage and transfer, with the embedding and restoration processes constituting a reversible task [30]–[37]. Xia *et al*. [30] propose an encoder-decoder network to convert a color image to an invertible grayscale image, and then invert the grayscale to color. Zhu *et al*. [32] embed a short video into an image with a motion embedding network, and convert it back to a video preview with a motion expansion network. Wu *et al*. [34] encode multiplane images into a single JPEG image which can be decoded to render novel views. Cheng *et al*. [35] proposed IICNet, the first generic framework based on invertible neural networks for solving invertible image processing tasks including image hiding, dual-view images embedding and spatial-temporal video embedding. However, these works are limited to handling video/image data, and cannot be easily generalized to invertible transformations between 3D scenes and embedding images. In this work, we focus on concealing a 3D scene into a carrier image in an invertible way, where the original scene can be almost losslessly reproduced to enable high-quality view synthesis.

3 Method

Given any pre-trained neural rendering model, our goal is to convert it into an intermediate representation that can fit to an invertible image embedding module. Specifically, we model the 3D scene as three neural planes that encode geometry and appearance along the X, Y and Z axes. Our

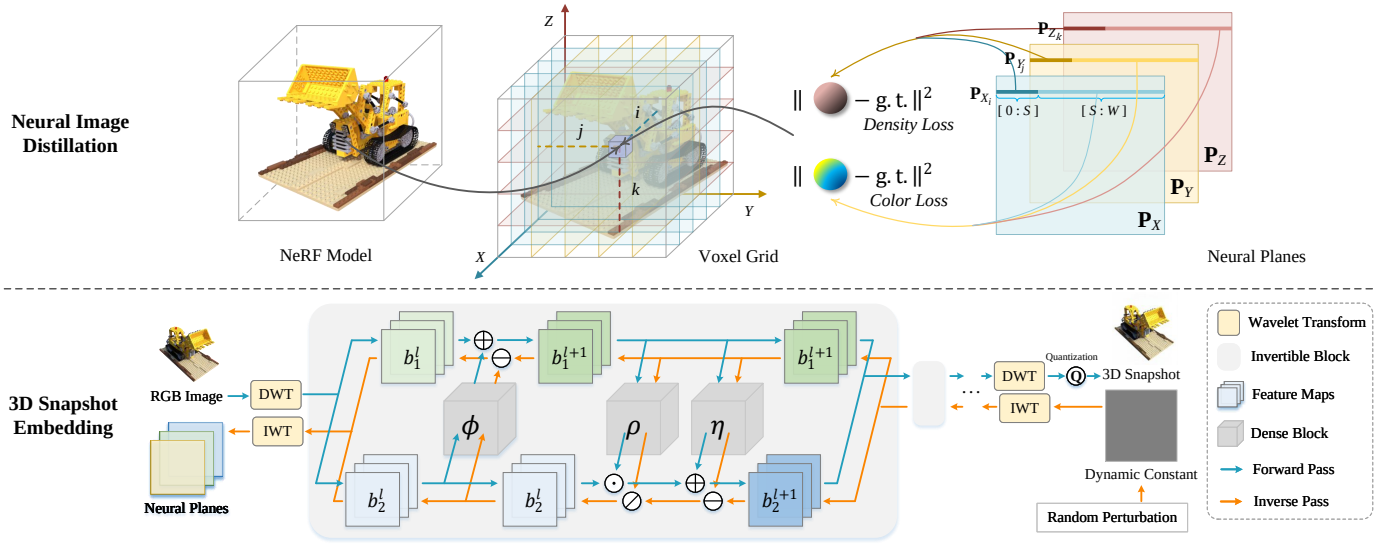


Fig. 2: Overview of our methods. Top: the neural image distillation module aims to distill a pre-trained NeRF model into a neural image consisting of three neural planes that encode variations of density and color along X, Y, Z axes. Bottom: the 3D snapshot embedding module takes the optimized neural planes and embeds them into an image with the proposed DINN framework to create a 3D snapshot, from which the embedded neural planes can be easily revealed in an invertible way.

representation is learned through querying the trained neural rendering model to collect the volume density and color at each location. These neural planes are then concatenated to form a neural image, which is further embedded into an image that depicts a specific view of the scene, creating a 3D snapshot that is easy to visualize and transmission-friendly. Furthermore, we utilize a dynamic invertible neural network (DINN) to enable bijective mapping and allow the neural planes to be easily restored from the 3D snapshot. The restored neural planes can recompose a voxel-based representation of the scene with little quality degradation, enabling synthesis of arbitrary novel views using classical volumetric rendering. Our pipeline is shown in Fig. 2.

3.1 Preliminaries on Neural Radiance Fields

A neural radiance field (NeRF) [1] represents a scene with a 5D function f_Θ whose input is a 3D location $\mathbf{x} = (x, y, z)$ and a 2D viewing direction (θ, ϕ), and whose output is the corresponding color $\mathbf{c} = (r, g, b)$ and volume density σ. To predict the color $\hat{\mathcal{C}}$ of a pixel in image, it casts a camera ray $\mathbf{r}$ from the pixel into the scene and accumulate the colors of N sampled points along the ray based on their densities:

$$\hat{\mathcal{C}}(\mathbf{r}) = \sum_{i=1}^{N} T_i \big(1 - \exp(-\sigma_i \delta_i)\big)\mathbf{c}_i, \tag{1}$$

with $T_i = \exp\left(-\sum_{j=0}^{i-1} \sigma_j \delta_j\right)$ where δ_i represents the distance between the i-th and $(i+1)$-th samples, and the term $1 - \exp(-\sigma_i \delta_i)$ denotes the amount of light contributed by the i-th sample. With this differentiable volume rendering process, NeRF can be trained using only 2D image supervision by minimizing the difference between the predicted pixel colors the ground-truth from the image:

$$\mathcal{L}_{NeRF} = \sum_{\mathbf{r}} \left\| \mathcal{C}(\mathbf{r}) - \hat{\mathcal{C}}(\mathbf{r}) \right\|_2^2. \tag{2}$$

3.2 Neural Image Distillation

To embed a NeRF model into an image form, a straightforward idea is to transform it to an image-like structure and conceal it in a host image. To this end, we propose neural image distillation to represent NeRF as a neural image $\mathcal{I} \in \mathbb{R}^{3\times H\times W}$ consisting of three channels $\mathbf{P}_X, \mathbf{P}_Y, \mathbf{P}_Z \in \mathbb{R}^{H\times W}$. Each channel is associated with a neural plane that carries the geometry and appearance information along the X, Y and Z axis, respectively. To render views from a volumetric radiance field, we need to know the volume density and view-dependent color at each location of the scene. Since volume density and color are unrelated features, for each neural plane we associate the first dimension to spatial coordinates along its corresponding axis, and divide it along the second dimension into two slices $\mathbf{P}_M^\sigma \in \mathbb{R}^{H\times S}$ and $\mathbf{P}_M^\mathbf{c} \in \mathbb{R}^{H\times(W-S)}$ (with $M \in \{X, Y, Z\}$) to model the variations in density and color, respectively. We empirically set S to around $W/4$. Next, we describe how the volume density and view-dependent color are obtained from the neural planes and how they are optimized.

Volume Density. We model the volume density as a scalar function $\hat{\sigma}(\cdot)$ of the spatial location independent of the view direction. Inspired by the idea of factorization, we define the density function at (x, y, z) with our neural planes as:

$$\hat{\sigma}(x_i, y_j, z_k) = \sum_{n=0}^{S} (\mathbf{P}_X^\sigma)_{i,n} \cdot (\mathbf{P}_Y^\sigma)_{j,n} \cdot (\mathbf{P}_Y^\sigma)_{k,n}, \tag{3}$$

where (x_i, y_j, z_k) represents the 3D location corresponding to the (i, j, k) index for the first dimension of the planes. To optimize the density planes $\{\mathbf{P}_M^\sigma \mid M = X, Y, Z\}$, we introduce a loss function that penalizes the deviation between $\hat{\sigma}(x_i, y_j, z_k)$ and the ground-truth $\sigma(x_i, y_j, z_k)$ obtained from the pre-trained NeRF f_Θ for all combination of indices:

$$\mathcal{L}_\sigma = \sum_{i,j,k} \left\| \hat{\sigma}(x_i, y_j, z_k) - \sigma(x_i, y_j, z_k) \right\|^2. \tag{4}$$

View-dependent Color. Different from density, the color at a location may vary as the view direction changes. We follow [2], [15] and represent the color as linear combination of spherical harmonics (SH), which are bases for functions defined over the surface of a unit sphere. In this way, the view-dependent color at a location can be determined using the corresponding linear combination coefficients. Let n_c be the number of SH coefficients (27 for all RGB channels using spherical harmonics of degree 2). Then we further divide the color slice of each neural plane into n_c segments, one for each coefficient. We predict each coefficient $\hat{e}_p$ ($p = 1, \ldots, n_c$) in a similar way as the density value prediction:

$$\hat{e}_p(x_i, y_j, z_k) = \sum\nolimits_{n=(p-1)\cdot N}^{p\cdot N} (\mathbf{P}_X^c)_{i,n} \cdot (\mathbf{P}_Y^c)_{j,n} \cdot (\mathbf{P}_Z^c)_{k,n} \quad (5)$$

where $N = \frac{(W-S)}{n_c}$. Then the color $\hat{\mathbf{c}}$ at location (x_i, y_j, z_k) viewed from direction $\mathbf{d}$ is be determined using the SH functions $\{Y_s\}$ and the coefficients $\{\hat{e}_p\}$ via:

$$\hat{\mathbf{c}}(x_i, y_j, z_k, \mathbf{d}) = \sum\nolimits_{s=1}^{\ell} \begin{bmatrix} \hat{e}_s(x_i, y_j, z_k) \\ \hat{e}_{s+\ell}(x_i, y_j, z_k) \\ \hat{e}_{s+2\ell}(x_i, y_j, z_k) \end{bmatrix} Y_s(\mathbf{d}), \quad (6)$$

where $\ell = n_c/3$ denotes the number of coefficients for each individual RGB channel. We optimize the color plates $\{\mathbf{P}_M^{\mathbf{c}} \mid M = X, Y, Z\}$ using a loss function

$$\mathcal{L}_c = \sum\nolimits_{i,j,k} \sum\nolimits_{\mathbf{d}\in\mathcal{D}} \left\| \hat{\mathbf{c}}(x_i, y_j, z_k, \mathbf{d}) - \mathbf{c}(x_i, y_j, z_k, \mathbf{d}) \right\|^2 \quad (7)$$

where $\mathcal{D}$ is a set of sample directions, and $\mathbf{c}$ is the ground-truth color obtained from the trained NeRF f_Θ.

3.3 3D Snapshot Embedding

Given a neural image $\mathbf{I}$, we introduce a Dynamic Invertible Neural Network (DINN) to embed it into a host image $\mathbf{H}$ that depicts the scene from a specific viewpoint, and we call it 3D snapshot. We aim to establish a bijective mapping between $\mathbf{I}$ and $\mathbf{H}$ so that the neural image can be easily revealed from the 3D snapshot embedding. To this end, we adopt an invertible neural network (INN) to enable bijective mapping and further propose to inject dynamicity into the process to facilitate the restoration of neural images.

Framework. As shown in Fig. 2, our 3D snapshot embedding framework includes a hiding process and a revealing process. These two processes can be effectively carried out in the same network due to the adoption of INN where the forward and backward operations can easily be reversed. Since the optimized neural images are complicated and semantically not meaningful, it is difficult to use off-the-shelf INNs to effectively hide them in a host. We thus propose a Dynamic Invertible Neural Network (DINN) to tailor INN for embedding neural images. Mathematically, we regard the hiding and revealing operations as a pair of inverse problems, and formulate the processes as:

$$(\mathbf{E}, \mathbf{C}) = \mathcal{F}(\mathbf{H}, \mathbf{I}), \quad \mathbf{E}' = \mathcal{Q}(\mathbf{E}), \quad (\hat{\mathbf{H}}, \hat{\mathbf{I}}) = \mathcal{F}^{-1}(\mathbf{E}', \mathbf{C}'),$$

where $\mathcal{F}$ is an invertible function, $\mathbf{E}$ and $\mathbf{C}$ have the same dimension as $\mathbf{H}$ and $\mathbf{I}$ respectively, and $\mathcal{Q}$ is the quantization operation. $\mathbf{E}'$, $\mathbf{C}'$, $\hat{\mathbf{H}}$, and $\hat{\mathbf{I}}$ are the embedding 3D snapshot, dynamic constant, restored host image, and restored neural image, respectively. Following [38]–[40], we represent $\mathcal{F}$ by a succession of invertible blocks, each containing two affine coupling layers. In the l-th block, the input b_l is divided into b_1^l and b_2^l along the channel dimension and the corresponding output is b_1^{l+1} and b_2^{l+1}. Then the forward pass can be represented as:

$$b_1^{l+1} = b_1^l + \phi(b_2^l), \quad b_2^{l+1} = b_2^l \odot \exp(\rho(b_1^{l+1})) + \eta(b_1^{l+1}),$$

where $\phi(\cdot)$, $\rho(\cdot)$ and $\eta(\cdot)$ are arbitrary functions. For the backward pass, b_1^l and b_2^l can be easily recovered by:

$$b_2^l = (b_2^{l+1} - \eta(b_1^{l+1})) \odot \exp(-\rho(b_1^{l+1})), \quad b_1^l = b_1^{l+1} - \phi(b_2^l).$$

The design of an invertible block is illustrated in Fig. 2. Note that the $\exp(\cdot)$ computation is omitted in the figure.

Wavelet Domain Hiding. The visible noise and lack of meaning in neural images can result in texture-copying artifacts and distorted embeddings when directly hidden in the spatial domain. To address this issue, we propose a frequency-domain hiding process, particularly in the high-frequency domain, which significantly enhances the visual quality of the embedding images. Specifically, we use the discrete wavelet transform (DWT) to extract low- and high-frequency wavelet sub-bands before passing them through the invertible blocks. We enforce a reconstruction loss that penalizes more in the low-frequency sub-band, forcing the network to hide information in the high-frequency domain. As wavelet transform is bidirectional symmetric [41], we can obtain the embedding images with the inverse wavelet transform (IWT) without compromising the network's invertibility. In this paper, we employ the Haar wavelet kernel for its simplicity and effectiveness to perform DWT and IWT.

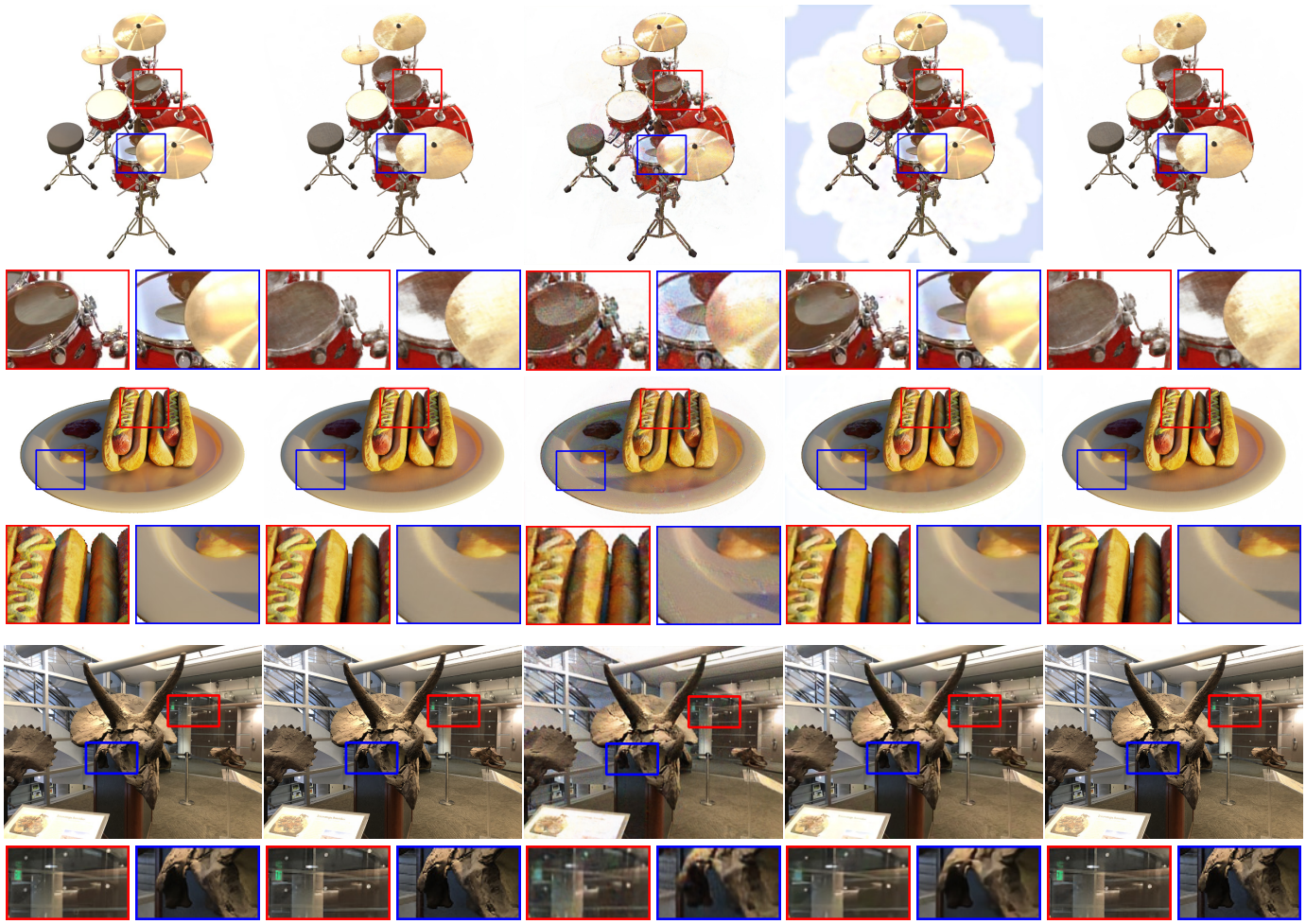
Dynamic Adaptation. The optimized neural images are noisy and semantically not meaningful, which makes the concealing tasks challenging [5]. Thus we inject dynamicity during the training process to adapt the network to noisy inputs. The dynamic adaptation is applied in two aspects. First, the last half of the output channels are dynamically updated constants, which facilitate the embedding by accumulating information across different neural images. Additionally, random noises are introduced to the constant channels in the backward pass, which makes the constant channels unreliable and thus forces the network to place useful information in the embedding channels. Specifically, $\mathbf{C}'$ is initially fixed, and then dynamically updated after a certain iteration count k via:

$$\mathbf{C}'_i = 0.9 \cdot \mathbf{C}'_{i-1} + 0.1 \cdot \mathbf{C}, \quad \text{for } i \geq k. \quad (8)$$

Quantization. A quantization operation is inevitable when saving the embedding in an image format (*e.g.* JPG or PNG) that only allows 8 bits per pixel per channel. We follow [42] and add uniform noises during training, and perform integer rounding to quantize the embedding image during inference. The quantized image is clamped between 0 and 255.

Loss Functions. As we aim at both concealing our neural image into a host for a 3D snapshot as well as recovering the neural image from it, we enforce loss functions on both ends. For embedding, we require the 3D snapshot to resemble the host image while carrying information of the neural image, especially in high-frequency domain:

$$\mathcal{L}_e = \|\mathbf{E} - \mathbf{H}\|_2^2, \qquad \mathcal{L}_{freq} = \|\mathcal{T}(\mathbf{E})_{Lf} - \mathcal{T}(\mathbf{H})_{Lf}\|_2, \quad (9)$$



(a) GT (b) Original (c) Video Snapshot (d) IICNet (e) Ours

Fig. 3: Reconstruction comparisons with the original NeRF model (b), and other invertible video methods (c & d).

where $\mathcal{T}(\cdot)_{Lf}$ represents the operation of extracting low-frequency sub-bands after DWT. For restoration, we require the restored host image and neural image to both match the original ones. Following [5], [35], [36], we use the L_1 distance as the restoration loss:

$$\mathcal{L}_{r1} = \|\hat{\mathbf{H}} - \mathbf{H}\|_1, \qquad \mathcal{L}_{r2} = \|\hat{\mathbf{I}} - \mathbf{I}\|_1. \tag{10}$$

In summary, our final loss function is:

$$\mathcal{L}_{total} = \lambda_1 \mathcal{L}_e + \lambda_2 \mathcal{L}_{freq} + \lambda_3 \mathcal{L}_{r1} + \lambda_4 \mathcal{L}_{r2}, \tag{11}$$

where λ_1, λ_2, λ_3, and λ_4 are balancing weights.

4 Experiments

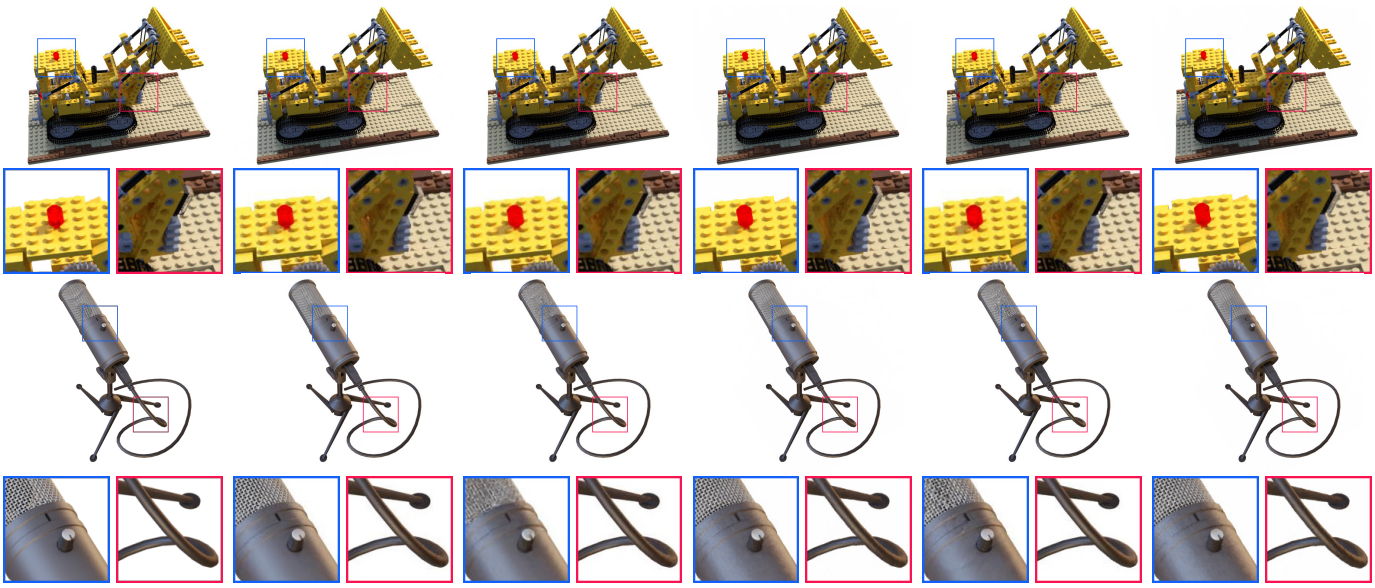
4.1 Experimental Setup

Implementation Details. We implement our method using Pytorch [43] and evaluate it on a single Nvidia GeForce RTX 3090. In our experiments, we set the resolution of our neural images as $H = W = 800$. The hyper-parameters in Eq. (11) are set as $\lambda_1 = 1.0$, $\lambda_2 = 2.0$, and $\lambda_3 = 0.5$, $\lambda_4 = 10.0$. It takes about 13 hours to train our model for 15,000 epochs.

Datasets and Competitors. We use two synthetic and three real-scene datasets, including NeRF-synthetic [1], Synthetic-NSVF [14], LLFF [44], Tanks and Temples [45], and DTU MVS dataset [46]. The synthetic datasets both contain eight scenes with realistic images of size 800×800 rendered from various viewpoints. Each scene includes 300 views with provided camera poses distributed in the upper hemisphere, with 100 views for training and 200 for testing. We collect 50 high-quality scenes from the above datasets to demonstrate the capability of our model in recovering plausible views from a single embedding image. The training and testing view split for each dataset all follows their original papers. Note that the Synthetic-NSVF dataset is excluded from training and used as the unseen scenes to validate our generalizability. We select TensoRF [3] as the original model.

Since our method is the first invertible neural rendering method, we compare it with two video-based invertible methods, Video Snapshot [32] and IICNet [35]. We follow their settings and select nine views for each scene to form a short video. We embed these input frames to an image using these two video-based invertible methods and recover all frames from the embedding image. Our method generates the same viewpoints to these frames and is compared with the invertible quality to them.

Metrics. To evaluate the success of reconstructing a 3D scene, we adopt the widely-used metrics, PSNR, SSIM, and LPIPS [47]. PSNR and SSIM indicate the pixel error between our recovered results and ground-truth, but they cannot



(a) GT (b) NeRF (c) DVGO (d) TensoRF (e) Plenoxels (f) Ours

Fig. 4: Comparisons with NeRF models on novel view synthesis.

TABLE 1: Quantitative comparison in reconstruction and size on 50 training scenes. Last two variants are directly tested or after 5-mins finetuning on 6 unseen scenes.

Method	NeRF	TensoRF	Plenoxels	PlenOctrees	Instant-NGP
PSNR	31.01	31.56	31.71	31.71	**33.18**
Size(MB)	5×50	3.9×50	778×50	1976×50	63.3×50
Method	DVGO	CCNeRF	**Ours**	Ours (Unseen)	Ours (FT)
PSNR	31.95	30.55	28.82	26.12	27.89
Size(MB)	612×50	4.4×50	**5+(0.16×50)**	5+(0.16×6)	5+(0.16×6)

measure semantic similarity. LPIPS is a perceptual-based metric that measures the perceptual similarity.

4.2 Results

Qualitative results. We show qualitative comparisons against the baseline methods in Fig. 3. For our method, the reconstruction results are rendered by first extracting the embedded neural image from our 3D snapshot and then using volume rendering to construct views from the restored neural image. For video-based invertible methods, we treat the views as successive frames of a video and directly embed them into one image (the intermediate frame of the input video), from which we decode all the embedded views. Our model can reconstruct views with minimal degradation in image quality when compared to both the ground-truth and the original model. Despite the challenge of embedding noisy neural images, our approach effectively preserves the overall geometry and appearance of the restored scenes. Notably, while Video snapshot and IICNet can produce plausible results, they are limited in their ability to handle a large number of inputs, whereas our method can theoretically synthesize views from arbitrary viewpoints since we embed a scene representation instead of images. In addition, we show comparisons with serveral NeRF-based methods for novel view synthesis in Fig. 4, and more results can be found in our supplementary materials.

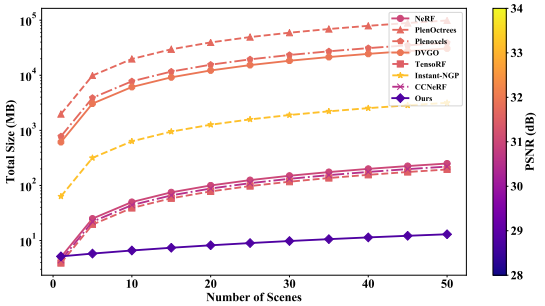


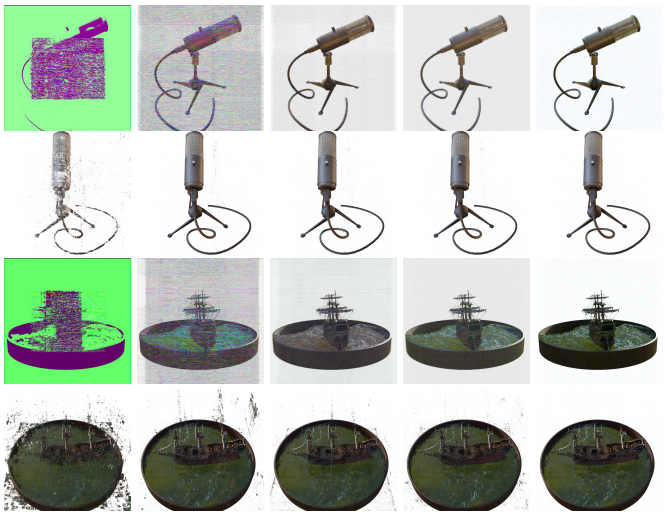
Fig. 5: Our method achieves around 25x smaller storage size and the efficiency increases as the number of scenes grows.

TABLE 2: Quantitative comparisons with video invertible methods. Note that we can restore an interactive 3D model.

	PSNR ↑	SSIM ↑	LPIPS ↓
VS [32]	28.12	0.85	0.118
IICNet [35]	26.45	0.81	0.120
Ours	**29.04**	**0.89**	**0.082**

Quantitative results. We conduct two sets of quantitative comparisons to evaluate our performance. Firstly, we compare our model with other NeRF models in terms of reconstruction quality and model size. Table 1 demonstrates that our model achieves comparable performance to existing NeRF models. Regarding model size, we can store all 3D scenes (50 available scenes in our settings) in a single INN network, which occupies approximately 5MB of storage, along with the 3D snapshot that facilitates restoring the corresponding scene. Each individual image requires an additional storage space of approximately 160KB, and our efficiency increases as the number of scenes grows. Fig. 5 presents a direct comparison of our model size with others.

Secondly, in Table 2, we compare our method with video-based invertible methods. We observe that our approach achieves comparable results to video-based invertible methods. However, video-based invertible methods can only restore a video, whereas our model can recover views from



(a) Enc.-Dec. (b) w/o FH (c) w/o DC (d) w/o DN (e) Ours

Fig. 6: Ablation evaluation on the embedding and reconstruction factors of our method. Odd rows are 3D snapshots and even rows are restorations.

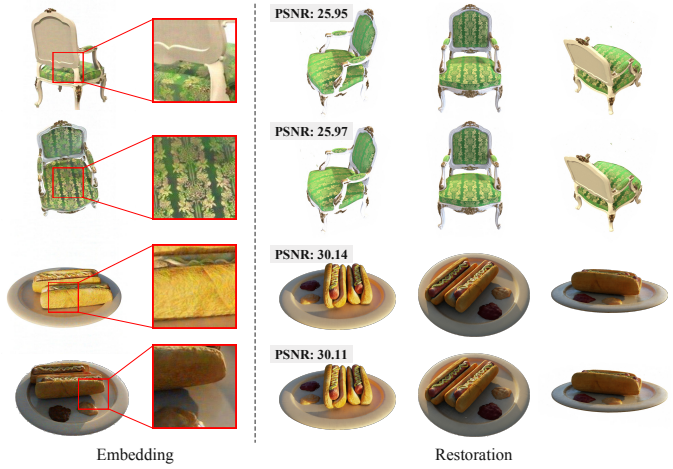


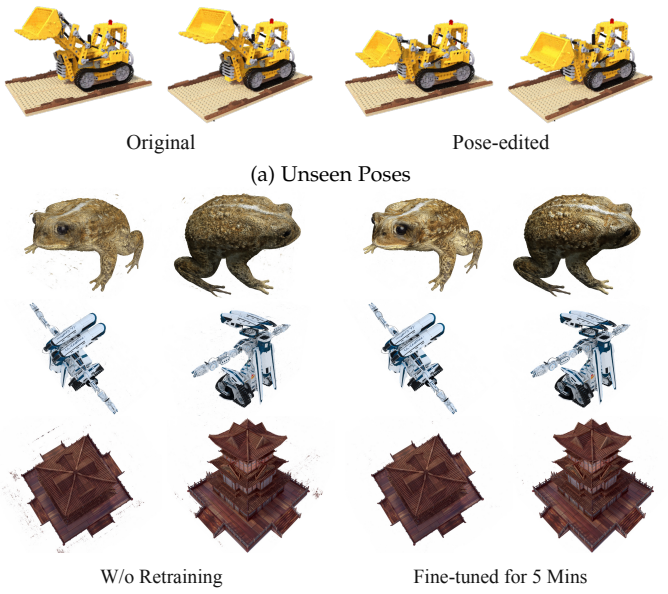
Fig. 7: Different choice of host images and their restoration results. PSNR value is shown in the first reconstruction.

the 3D model. This property is especially important, as it enables us to recover views from arbitrary viewpoints, unlike video-based methods that are limited to restoring views from specific camera positions.

Ablation Study. We conduct a series of experiments to evaluate the effectiveness of different components in our method by designing four variants for comparison:

- Encoder-Decoder: an encoder-decoder architecture that directly embeds neural images in the hosts;
- w/o Frequency Hiding: operating the hiding process in spatial domain instead of frequency domain;
- w/o Dynamic Constant: disabling the dynamic update of the constant channel;
- w/o Dynamic Noise: not perturbing the constant channel during training.

The results in Fig. 6 show that our dynamic setting can help boost the reconstruction performance, as the dynamic constant provides embedding information in the form of constant noise and dynamic noise enables it to tolerate the



(a) Unseen Poses

(b) Unseen Scenes

Fig. 8: Testing on unseen cases.

differences between scenes. In contrast, using the encoder-decoder structure, we cannot successfully embed and reconstruct the 3D model, indicating that the noisy neural image is challenging to embed with a vanilla network.

Choice of Host Image. The host image serves both as a carrier that embeds the 3D scene and as a preview of the scene. We have conducted experiments to investigate the impact of the choice of host images on the reconstruction quality. Fig. 7 shows the reconstruction results using different host images. The minor difference in the PSNR value indicates that the choice of host images has negligible impact on the reconstruction quality. We provide more details and more results in the supplementary materials.

Network Capacity and Generalizability. Once the network is properly trained, we are able to embed and reconstruct all NeRF models in the training set using a single model. Although we only apply it to 50 high-quality scenes in this paper, the underlying principle of our network allows for more models to be encoded.

Moreover, we aim to explore the generalizability of our model by investigating its ability to embed and invert unseen scenes. We conducted two experiments to validate this. The first experiment, depicted in Fig. 8a, involves a simpler scenario where the object is present in the training set but appears in a different pose (edited using NeRF-Editing [48]). Our model is capable of perfectly embedding and reconstructing the unseen 3D poses, indicating that it can be used to embed spatio-temporal neural rendering models into video 3D snapshots.

The second experiment, shown in Fig. 8b, involves completely unseen scenes. Our model can reconstruct a 3D model with minor artifacts and less details, mainly due to the limited availability of training data. However, it is worth noting that fine-tuning the model for only 5 minutes enables us to incorporate the previously unseen scene into the model for better reconstruction. The quantitative results for 6 unseen scenes from Synthetic-NSVF [14] are presented in Tab. 1. The performance is still comparable for unseen scenario and gets

even better after 5-minute fine-tuning.

5 Conclusion

We propose a novel approach to embed a pre-trained neural rendering model into a plausible image representation in an invertible way, which can be used to recover the 3D scene if needed. Our key idea is to optimize three neural planes that encode the information from the original neural model along three dimensions, and then utilize a dynamic invertible neural network to embed such representation into a plausible carrier image. Its ability to embed neural 3D scenes into images also enables potential further applications in relevant domains such as data storage, VR, and AR.

References

[1] B. Mildenhall, P. Srinivasan, M. Tancik, J. Barron, R. Ramamoorthi, and R. Ng, "Nerf: Representing scenes as neural radiance fields for view synthesis," in *ECCV*, 2020.

[2] A. Yu, R. Li, M. Tancik, H. Li, R. Ng, and A. Kanazawa, "Plenoctrees for real-time rendering of neural radiance fields," in *ICCV*, 2021, pp. 5752–5761.

[3] A. Chen, Z. Xu, A. Geiger, J. Yu, and H. Su, "Tensorf: Tensorial radiance fields," in *ECCV*, 2022.

[4] J. Tang, X. Chen, J. Wang, and G. Zeng, "Compressible-composable nerf via rank-residual decomposition," in *NeurIPS*, 2022.

[5] S.-P. Lu, R. Wang, T. Zhong, and P. L. Rosin, "Large-capacity image steganography based on invertible neural networks," in *CVPR*, 2021, pp. 10 816–10 825.

[6] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *JMLR*, vol. 15, no. 1, pp. 1929–1958, 2014.

[7] C. Jiang, A. Sud, A. Makadia, J. Huang, M. Nießner, T. Funkhouser *et al.*, "Local implicit grid representations for 3d scenes," in *CVPR*, 2020, pp. 6001–6010.

[8] J. J. Park, P. Florence, J. Straub, R. Newcombe, and S. Lovegrove, "Deepsdf: Learning continuous signed distance functions for shape representation," in *CVPR*, 2019, pp. 165–174.

[9] L. Mescheder, M. Oechsle, M. Niemeyer, S. Nowozin, and A. Geiger, "Occupancy networks: Learning 3d reconstruction in function space," in *CVPR*, 2019, pp. 4460–4470.

[10] K. Genova, F. Cole, A. Sud, A. Sarna, and T. Funkhouser, "Local deep implicit functions for 3d shape," in *CVPR*, 2020, pp. 4857–4866.

[11] J. T. Barron, B. Mildenhall, M. Tancik, P. Hedman, R. Martin-Brualla, and P. P. Srinivasan, "Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields," in *ICCV*, 2021, pp. 5855–5864.

[12] K. Zhang, G. Riegler, N. Snavely, and V. Koltun, "Nerf++: Analyzing and improving neural radiance fields," *arXiv preprint arXiv:2010.07492*, 2020.

[13] C. Reiser, S. Peng, Y. Liao, and A. Geiger, "Kilonerf: Speeding up neural radiance fields with thousands of tiny mlps," in *ICCV*, 2021, pp. 14 335–14 345.

[14] L. Liu, J. Gu, K. Zaw Lin, T.-S. Chua, and C. Theobalt, "Neural sparse voxel fields," *NeurIPS*, vol. 33, pp. 15 651–15 663, 2020.

[15] S. Fridovich-Keil, A. Yu, M. Tancik, Q. Chen, B. Recht, and A. Kanazawa, "Plenoxels: Radiance fields without neural networks," in *CVPR*, 2022, pp. 5501–5510.

[16] Q. Xu, Z. Xu, J. Philip, S. Bi, Z. Shu, K. Sunkavalli, and U. Neumann, "Point-nerf: Point-based neural radiance fields," in *CVPR*, 2022, pp. 5438–5448.

[17] E. R. Chan, C. Z. Lin, M. A. Chan, K. Nagano, B. Pan, S. De Mello, O. Gallo, L. J. Guibas, J. Tremblay, S. Khamis, T. Karras, and G. Wetzstein, "Efficient geometry-aware 3d generative adversarial networks," in *CVPR*, June 2022, pp. 16 123–16 133.

[18] T. Müller, A. Evans, C. Schied, and A. Keller, "Instant neural graphics primitives with a multiresolution hash encoding," *arXiv preprint arXiv:2201.05989*, 2022.

[19] C.-K. Chan and L.-M. Cheng, "Hiding data in images by simple lsb substitution," *Pattern recognition*, vol. 37, no. 3, pp. 469–474, 2004.

[20] F. Pan, J. Li, and X. Yang, "Image steganography method based on pvd and modulus function," in *ICECC*, 2011, pp. 282–284.

[21] P. Tsai, Y.-C. Hu, and H.-L. Yeh, "Reversible image hiding scheme using predictive coding and histogram shifting," *Signal processing*, vol. 89, no. 6, pp. 1129–1143, 2009.

[22] B. C. Nguyen, S. M. Yoon, and H.-K. Lee, "Multi bit plane image steganography," in *IWDW Workshop*, 2006, pp. 61–70.

[23] M. Niimi, H. Noda, E. Kawaguchi, and R. O. Eason, "High capacity and secure digital steganography to palette-based images," in *ICIP*, vol. 2. IEEE, 2002, pp. II–II.

[24] H. Shi, J. Dong, W. Wang, Y. Qian, and X. Zhang, "Ssgan: Secure steganography based on generative adversarial networks," in *PCM*, 2018, pp. 534–544.

[25] W. Tang, S. Tan, B. Li, and J. Huang, "Automatic steganographic distortion learning using a generative adversarial network," *IEEE Signal Processing Letters*, vol. 24, no. 10, pp. 1547–1551, 2017.

[26] J. Yang, D. Ruan, J. Huang, X. Kang, and Y.-Q. Shi, "An embedding cost learning framework using gan," *IEEE TIFS*, vol. 15, pp. 839–851, 2019.

[27] W. Tang, B. Li, S. Tan, M. Barni, and J. Huang, "Cnn-based adversarial embedding for image steganography," *IEEE TIFS*, vol. 14, no. 8, pp. 2074–2087, 2019.

[28] S. Baluja, "Hiding images within images," *IEEE TPAMI*, vol. 42, no. 7, pp. 1685–1697, 2019.

[29] J. Zhu, R. Kaplan, J. Johnson, and L. Fei-Fei, "Hidden: Hiding data with deep networks," in *ECCV*, 2018, pp. 657–672.

[30] M. Xia, X. Liu, and T.-T. Wong, "Invertible grayscale," *ACM TOG*, vol. 37, no. 6, pp. 1–10, 2018.

[31] Y. Du, Y. Xu, T. Ye, Q. Wen, C. Xiao, J. Dong, G. Han, and S. He, "Invertible grayscale with sparsity enforcing priors," *ACM TOMM*, vol. 17, no. 3, 2021.

[32] Q. Zhu, C. Han, G. Han, T.-T. Wong, and S. He, "Video snapshot: Single image motion expansion via invertible motion embedding," *IEEE TPAMI*, vol. 43, no. 12, pp. 4491–4504, 2020.

[33] Y. Xing, Z. Qian, and Q. Chen, "Invertible image signal processing," in *CVPR*, 2021, pp. 6287–6296.

[34] Y. Wu, G. Meng, and Q. Chen, "Embedding novel views in a single jpeg image," in *ICCV*, 2021, pp. 14 519–14 527.

[35] K. L. Cheng, Y. Xie, and Q. Chen, "Iicnet: A generic framework for reversible image conversion," in *ICCV*, 2021, pp. 1991–2000.

[36] M. Xiao, S. Zheng, C. Liu, Y. Wang, D. He, G. Ke, J. Bian, Z. Lin, and T.-Y. Liu, "Invertible image rescaling," in *ECCV*, 2020, pp. 126–144.

[37] Z. Zhong, L. Chai, Y. Zhou, B. Deng, J. Pan, and S. He, "Faithful extreme rescaling via generative prior reciprocated invertible representations," in *CVPR*, 2022, pp. 5708–5717.

[38] L. Ardizzone, J. Kruse, C. Rother, and U. Köthe, "Analyzing inverse problems with invertible neural networks," in *ICLR*, 2018.

[39] L. Dinh, D. Krueger, and Y. Bengio, "Nice: Non-linear independent components estimation," *arXiv preprint arXiv:1410.8516*, 2014.

[40] L. Dinh, J. Sohl-Dickstein, and S. Bengio, "Density estimation using real nvp," *arXiv preprint arXiv:1605.08803*, 2016.

[41] S. Mallat, "A theory for multiresolution signal decomposition: the wavelet representation," *IEEE TPAMI*, vol. 11, no. 7, pp. 674–693, 1989.

[42] J. Ballé, V. Laparra, and E. P. Simoncelli, "End-to-end optimized image compression," *arXiv preprint arXiv:1611.01704*, 2016.

[43] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *NeurIPS*, 2019, pp. 8024–8035.

[44] B. Mildenhall, P. P. Srinivasan, R. Ortiz-Cayon, N. K. Kalantari, R. Ramamoorthi, R. Ng, and A. Kar, "Local light field fusion: Practical view synthesis with prescriptive sampling guidelines," *ACM TOG*, vol. 38, no. 4, pp. 1–14, 2019.

[45] A. Knapitsch, J. Park, Q.-Y. Zhou, and V. Koltun, "Tanks and temples: Benchmarking large-scale scene reconstruction," *ACM TOG*, vol. 36, no. 4, pp. 1–13, 2017.

[46] R. Jensen, A. Dahl, G. Vogiatzis, E. Tola, and H. Aanæs, "Large scale multi-view stereopsis evaluation," in *CVPR*, 2014, pp. 406–413.

[47] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang, "The unreasonable effectiveness of deep features as a perceptual metric," in *CVPR*, 2018, pp. 586–595.

[48] Y.-J. Yuan, Y.-T. Sun, Y.-K. Lai, Y. Ma, R. Jia, and L. Gao, "Nerf-editing: geometry editing of neural radiance fields," in *CVPR*, 2022, pp. 18 353–18 364.