Vision Transformer for NeRF-Based View Synthesis from a Single Input Image

Kai-En Lin[1], Lin Yen-Chen[2], Wei-Sheng Lai[3],
Tsung-Yi Lin[4⋆], Yi-Chang Shih[3], and Ravi Ramamoorthi[1]

[1] UC San Diego
[2] MIT
[3] Google
[4] NVIDIA

Abstract. Although neural radiance fields (NeRF) have shown impressive advances for novel view synthesis, most methods typically require multiple input images of the same scene with accurate camera poses. In this work, we seek to substantially reduce the inputs to a *single unposed image*. Existing approaches condition on local image features to reconstruct a 3D object, but often render blurry predictions at viewpoints that are far away from the source view. To address this issue, we propose to leverage both the global and local features to form an expressive 3D representation. The global features are learned from a vision transformer, while the local features are extracted from a 2D convolutional network. To synthesize a novel view, we train a multilayer perceptron (MLP) network conditioned on the learned 3D representation to perform volume rendering. This novel 3D representation allows the network to reconstruct unseen regions without enforcing constraints like symmetry or canonical coordinate systems. Our method can render novel views from only a single input image and generalize across multiple object categories using a single model. Quantitative and qualitative evaluations demonstrate that the proposed method achieves state-of-the-art performance and renders richer details than existing approaches.
`https://cseweb.ucsd.edu/%7eviscomp/projects/VisionNeRF/`

Keywords: image-based rendering, transformer, novel view synthesis

1 Introduction

We study the problem of novel view synthesis from a *single unposed image*. Recent advances [37,39,55] on this problem typically infer the 3D shape and appearance by first extracting image features from the input image. Then, given a query 3D point, they project it onto the image plane and condition on the corresponding pixel's features for predicting the color and density. These image-conditioned models work well for rendering target views that are close to the input view. However, the rendering quality degrades significantly when target

⋆ Work done while at Google.

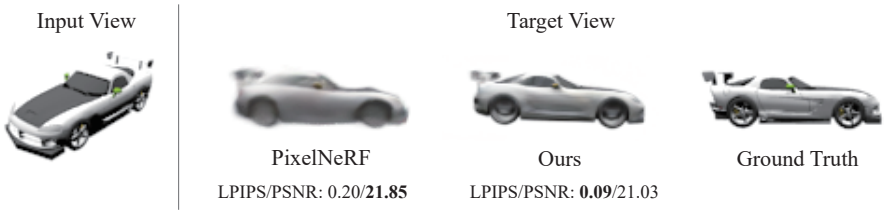

Fig. 1. Novel view synthesis in occluded regions. The visual quality of image-conditioned model (e.g., PixelNeRF [55]) degrades significantly when pixels in the target view are not visible in the input. To address this issue, we propose to incorporate both global features from vision transformer (ViT) and local appearance features from convolutional networks to achieve better rendering quality in occluded regions. Note that LPIPS [56] (lower is better) reflects the perceptual similarity better than PSNR.

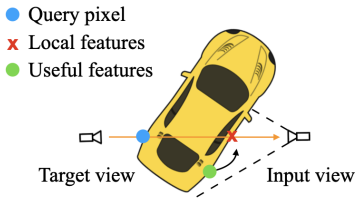

Fig. 2. The challenge of image-conditioned models in the presence of self-occlusion. To render a car's occluded wheel (blue dot) in the target view, image-conditioned models, e.g., PixelNeRF [55], query features along the ray, which corresponds to the car's window in the input view (red cross). Our method uses self-attention to learn long-range dependencies, which is able to find the most related features in the source view (green dot) for rendering a clear target view.

views move further and thus consist of many pixels that are not observed from the input view (see Fig. 1). We hypothesize that this is caused by the incorrectly-conditioned features in the presence of self-occlusion. As illustrated in Fig. 2, when the query pixel in the target view (e.g., the car's wheel) is not visible in the input view, image-conditioned models may use incorrect features from the other surface (e.g., the car's window) to predict the target view.

To tackle this issue, we propose a novel approach that utilizes the recent advances in vision transformer (ViT) and neural radiance fields (NeRF) to learn a better 3D representation. We first lift the input 2D image into feature tokens and apply ViT to learn global information. Subsequently, the feature tokens are unflattened and resampled into multi-level feature maps which allow the network to capture global information in a coarse-to-fine manner. In addition, we adopt a 2D convolutional neural network (CNN) to extract local features that capture details and appearance from the input image. Finally, we render the novel viewpoints using the volumetric rendering technique [29]. Our method is able to render unseen regions with more accurate structure and finer details.

We train and evaluate our method on the ShapeNet dataset [5], which has 13 object categories. Our method can generalize well across multiple categories and be applied to real input images. Quantitative and qualitative comparisons demonstrate that the proposed method performs favorably against existing approaches, e.g., SRN [43], PixelNeRF [55] and FE-NVS [15], and generates more visually appealing results. We summarize our contributions as follows:

Table 1. Comparisons with recent novel-view synthesis methods. Our method takes as input a single image to perform novel view synthesis. Different from methods that assume an object-centered coordinate system, we infer the 3D representation in viewer-centered coordinate system and thus do not require the camera pose of the input. Additionally, our method is able to generalize to multiple categories using a single model. We extract local image features using 2D CNN and retrieve global information using a ViT encoder to synthesize faithful and appealing details on occluded regions (see Figure 1).

	NeRF [29]	PIFu [39]	PixelNeRF [55]	CodeNeRF [18]	NeRFormer [37]	FE-NVS [15]	**Ours**
Single-view input	✗	✓	✓	✗	✗	✓	✓
Viewer-centered coordinate	✗	✓	✓	✗	✓	✓	✓
Cross-category generalization	✗	✓	✓	✗	✓	✓	✓
Image features	✗	✓	✓	✗	✓	✓	✓
Global features	✗	✗	✗	✓	✗	✗	✓

- We introduce a NeRF-based rendering method that can synthesize novel views from a single unposed image.
- We propose a novel 3D representation that integrates global and local information using vision transformer and 2D CNN.
- We demonstrate state-of-the-art performance against existing approaches on category-specific and category-agnostic datasets as well as real input images.

2 Related work

Our work takes inspiration from previous methods on novel view synthesis and recent advances in transformers. In this section, we briefly review previous papers on these topics.

2.1 Novel View Synthesis

Novel view synthesis is a long-standing topic for vision and graphics research. Earlier works in view interpolation [6] and light fields [14,24] establish the groundwork for image-based rendering. Later works utilize proxy geometry [4,8] and layered representations [40,45] to better represent the 3D scene and synthesize novel views. Since deep learning becomes the mainstream research focus, there has been a plethora of learning-based methods [12,13,20,25,26,28,42,58] and single-input view synthesis algorithms [31,39,41,52,53,54]. These approaches exploit the differentiable rendering pipeline to generate photorealistic results. An interesting breakthrough in image-based rendering is the surge of volumetric rendering methods based on the neural radiance fields (NeRF) [29]. NeRF encodes the 3D scene in a compact continuous 5D function, allowing photorealistic reconstruction of the given scene. Nonetheless, it requires tens or hundreds of input images and time-consuming optimization to overfit to the scene correctly. To address this problem, there have been several methods [37,46,49,55] utilizing

2D image features to improve the generalization. Another way to tackle generalization is to use pretrained networks with 1D latent code to represent the 3D shape, such as CodeNeRF [18]. Guo et al. [15] adopt a discrete 3D volume to represent the scene and achieve real-time rendering performance. Instead of relying on pure 1D, 2D, or 3D representations, we propose to learn a novel 3D representation that utilizes global information and local image features. We show a comparison of the proposed method to previous approaches [15,18,29,37,39,55] in Table 1.

2.2 Transformer

The transformer architecture [47] has brought significant advances in natural language processing (NLP). Variants based on self-attention have achieved state-of-the-art performance in many NLP [3,9] and vision [10,36,44] tasks. However, directly applying self-attention to an image requires each pixel to be attended to every other pixel, which significantly increases the computational costs and prohibits the self-attention to be applied to large image resolutions. To address this issue, several works [17,33,35,57] approximate self-attention by applying it to local patches of each query pixel. Recently, ViT [10] and follow-up works [36,50] demonstrated that applying a transformer to a sequence of patches (split from an image) can achieve competitive performance on discriminative tasks (e.g., image classification). Several works also adopt transformer for novel view synthesis. Wang et al. [48] include transformers in both the encoder and decoder for 3D reconstruction from multi-views. NeRF-ID [1] uses a transformer to sample 3D points along rays. Other approaches [19,37,49] use transformers to aggregate source view features extracted by a CNN. Our work is different from these methods as we focus on learning global image information using ViT. We demonstrate that ViT can learn 3D representation to achieve better reconstruction quality on unseen regions than the approaches based on 2D CNN.

3 Novel View Synthesis From a Single Image

Our goal is to infer a 3D representation from a single input image for novel view synthesis. We first discuss three different paradigms to learn such a 3D representation (Sec. 3.1). Then, we propose a hybrid representation to improve rendering quality on occluded regions, where we utilize a vision transformer [10] to encode global information (Sec. 3.2) and a 2D CNN to encode local appearance features (Sec. 3.3). Finally, we learn a NeRF [29] module that conditions the encoded features to render novel views via volume rendering (Sec. 3.4).

3.1 Synthesizing Occluded Regions with Limited Information

In this section, we identify and describe the common ways to reconstruct unseen regions in previous works (see Fig. 3 for illustration). Additionally, we analyze the

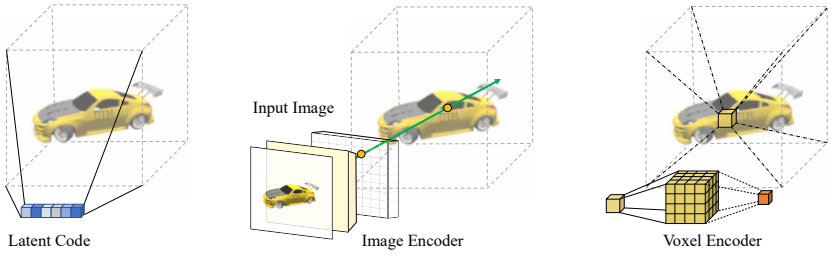

Fig. 3. Illustration of different representations for a 3D object. (a) 1D latent code-based approaches [7,11,18,27,30,32] encode the 3D object in an 1D vector. (b) 2D image-based methods [39,55] are conditioned on the per-pixel image features to reconstruct any 3D point. (c) 3D voxel-based approaches [15,26] treat a 3D object as a collection of voxels and apply 3D convolutions to generate RGBσ.

strengths and weaknesses of each method and propose a hybrid representation to address the critical issues in existing methods.

Given a single image $\mathbf{I}_s$ at camera s, our task is to synthesize novel view $\mathbf{I}_t$ at camera t. For a 3D point $\mathbf{x}$ in the scene, if it is visible in the source image, we can directly use the color $\mathbf{I}_s(\pi(\mathbf{x}))$, where π denotes the projection to source view, to represent the point as seen by a novel viewpoint. However, if it is occluded, we have to resort to information other than the color at the projection $\pi(\mathbf{x})$. Typically, there are three possible solutions to gather such information.

1D latent code-based approaches. First, a common way [7,11,18,27,30,32,38] to represent and reconstruct shape and appearance from a single image is to exploit a 1D global latent code $\mathbf{z}$ (shown in Fig. 3(a)). The latent code can incorporate shape information and priors in a compact vector representation, and it is later decoded by neural networks to generate explicit color and density. As demonstrated in Jang et al. [18], the color and density can be acquired by

$$(\sigma, \mathbf{c}) = \mathcal{F}_{\text{1D}}(\mathbf{z}; \mathbf{x}; \mathbf{d}). \tag{1}$$

The spatially-varying input in this case is the sample position $\mathbf{x}$ and viewing direction $\mathbf{d}$ in the canonical coordinate. Intuitively, every 3D point has the same latent code input and thus there is less inductive bias compared to other methods. As a result, this method is good for (a) recovering unseen regions and (b) shape/appearance editing by changing the latent code. However, this method requires (a) ground truth camera poses to achieve better rendering quality and (b) time-consuming optimization to find the correct latent code for representing an object.

2D image-based approaches. There has been a lot of interest around image-conditioned methods, such as PIFu [39] and PixelNeRF [55], due to the flexibility and high-quality results around the input views. Moreover, they are more computationally efficient as they operate in the 2D image space rather than 3D

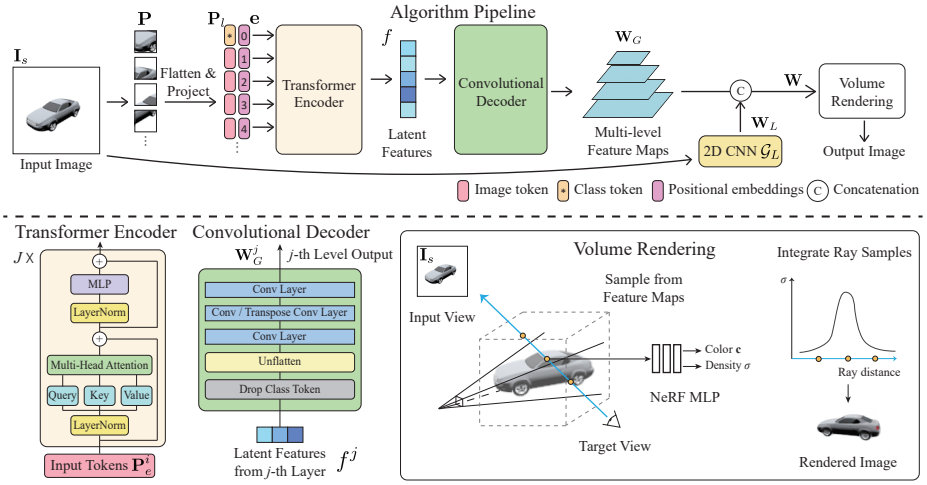

Fig. 4. Overview of our rendering pipeline. We first divide an input image into $N = 16 \times 16$ patches $\mathbf{P}$. Each patch is flattened and linearly projected to an image token $\mathbf{P}_l$. The transformer encoder takes the image tokens and learnable positional embeddings $\mathbf{e}$ as input to extract global information as a set of latent features f (Sec. 3.2). Then, we decode the latent feature into multi-level feature maps $\mathbf{W}_G$ using a convolutional decoder. In addition to global features, we use another 2D CNN $\mathcal{G}_L$ to obtain local image features (Sec. 3.3). Finally, we sample the features for volume rendering using the NeRF MLP (Sec. 3.4).

voxels, as illustrated in Fig. 3(b). Taking PixelNeRF [55] as an example, we can define the output as

$$(\sigma, \mathbf{c}) = \mathcal{F}_{\text{2D}}(\mathbf{W}(\pi(\mathbf{x})); \mathbf{x}_c; \mathbf{d}_c), \tag{2}$$

where $\mathbf{x}_c$ is the 3D position and $\mathbf{d}_c$ the ray direction in camera coordinates. In this case, the spatial information is encoded inside the feature map $\mathbf{W}$ when it is extracted by an image encoder. Consequently, any 3D point along a ray $\mathbf{x}_t \in \mathbf{r}$ would share the same feature $\mathbf{W}(\pi(\mathbf{x}_t))$. This representation (a) encourages better rendering quality in visible areas and (b) is more computationally efficient. However, it often (a) generates blurry predictions in unseen parts as shown in Fig. 1 and (b) has limited receptive fields.

3D volume-based approaches. To utilize 3D locality, another way is to treat the object as a set of voxels in 3D space and apply 3D convolutions to reconstruct unseen areas (see Fig. 3(c)). The voxel grid can be constructed by unprojecting 2D images or feature maps to a 3D volume [15]. For each 3D point, we have features $\mathbf{W}(\pi(\mathbf{x}))$ and 3D location $\mathbf{x}$. The 3D CNN can utilize information from neighboring voxels to infer geometry and appearance at $\mathbf{x}$ as follows

$$(\sigma, \mathbf{c}) = \mathcal{F}_{\text{3D}}(\mathbf{W}(\pi(\mathbf{x}_n)); \mathbf{x}_n), \tag{3}$$

where $\mathbf{x}_n$ denotes the set of neighboring voxels of $\mathbf{x}$. The benefits of this method are (a) faster rendering time and (b) using 3D prior to rendering unseen geometry. On the other hand, it suffers from (a) limited rendering resolution due to the voxel size and (b) limited receptive fields.

Our approach. We observe that the 1D approach enjoys a holistic view on the object and is able to encode the overall shape in a compact format. The 2D method offers better visual quality around input views, while the 3D method provides a way to refine the shape. However, volume-based methods are more computationally-intensive and require more memory when increasing the grid size. Our method is more closer to 2D image-based approaches, while sharing the spirit with the 1D latent-code based methods to encode global information. Specifically, we utilize (i) a ViT architecture and its fully-connected networks to learn global information, and (ii) a 2D CNN module to extract local image features. Recent success in vision transformer [10,36] shows the efficacy of using ViT to learn for long-range dependencies between features. Thus, our local and global hybrid representation allows for more flexibility and better reconstruction quality in the unseen regions. Unlike CodeNeRF [18] and DISN [53], our method does not require a canonical coordinate system to utilize the global features. Our method enjoys benefits of the high-resolution image features from 2D-CNN while not suffering from small receptive fields due to the use of ViT encoder.

3.2 Vision Transformer (ViT) as Global Feature Encoder

We adopt the image-based approach that conditions on per-pixel feature $\mathbf{W}$ for rendering. We divide $\mathbf{W}$ into two parts: (i) global feature maps $\mathbf{W}_G$ and (ii) local feature maps $\mathbf{W}_L$. In this section, we describe how we obtain $\mathbf{W}_G$ with a vision transformer. Our model takes as an input a single image $\mathbf{I}_s \in \mathbb{R}^{H\times W\times 3}$, where H and W are the image height and width, respectively.

Flatten and project. As shown in Fig. 4, the image $\mathbf{I}_s$ is first reshaped into a sequence of flattened 2D patches $\mathbf{P} \in \mathbb{R}^{N\times P^2\times 3}$, where $N = \frac{HW}{P^2}$ is the number of patches, and P denotes the patch size [10]. As the transformer takes a latent vector of size D, we project the patches with a trainable linear layer to produce $\mathbf{P}_l \in \mathbb{R}^{N\times D}$. In previous ViT work [10], a learnable class token is usually concatenated to the image tokens to incorporate global information that is not grounded in the input image. In our case, we treat the class token as a "background" token to represent features that are not shown in the image. Consequently, we have $N+1$ tokens in total, denoted as $\mathbf{P}_l^0, \mathbf{P}_l^1, ..., \mathbf{P}_l^{N+1}$. We also add learnable positional embeddings $\mathbf{e}$ to distinguish between different spatial patches:

$$\mathbf{P}_e^i = \mathbf{P}_l^i + \mathbf{e}^i. \tag{4}$$

Transformer encoder. The image tokens $\{\mathbf{P}_e^0, \mathbf{P}_e^1, ..., \mathbf{P}_e^{N+1}\}$ undergo J transformer layers to generate latent features f^j, where j denotes the output of the j-th transformer layer. The transformer layer is composed of multiheaded self-attention (MSA) and MLP layers [10]. The MSA block performs self-attention

on the images and extracts information by comparing a pair of tokens. Therefore, the transformer encoder has a global receptive field in all the layers, which can easily learn long-range dependency between different image patches [10,36].

Convolutional decoder. After generating a set of latent features $f = \{f^0, ..., f^J\}, f^j \in \mathbb{R}^D$, our algorithm then utilizes a convolutional decoder to promote the latent features into multi-level feature maps. These multi-level feature maps extract coarse-to-fine global information and allow us to concatenate with the local appearance features in the final rendering stage (see Sec. 3.3). To generate the feature maps, we first drop the class token. The class token is useful during the self-attention stage but does not have physical meaning when unflattened [36]. Consequently, we define the operation as $\text{Drop} : \mathbb{R}^{(N+1)\times D} \rightarrow \mathbb{R}^{N\times D}$. After dropping the class token, we unflatten the image by $\text{Unflatten} : \mathbb{R}^{N\times D} \rightarrow \mathbb{R}^{\frac{H}{P}\times\frac{W}{P}\times D}$. Now we have a set of feature patches $\mathbf{P}_f = \{\mathbf{P}_f^0, ..., \mathbf{P}_f^J\}$, where $\mathbf{P}_f^j \in \mathbb{R}^{\frac{H}{P}\times\frac{W}{P}\times D}$. We then construct the multi-level feature maps with a set of convolutional decoders as in Fig. 4. The convolutional decoders are defined as $\text{Decode} : \mathbb{R}^{\frac{H}{P}\times\frac{W}{P}\times D} \rightarrow \mathbb{R}^{H^j\times W^j\times D^j}$, where the feature patches are (i) first convolved with a 1×1 convolution layer, (ii) resampled with a strided convolution or transposed convolution to have size $H^j \times W^j$, and (iii) convolved with a 3×3 convolution layer to have D^j channels. We can describe the feature maps as

$$\mathbf{W}_G^j = (\text{Decode} \circ \text{Unflatten} \circ \text{Drop})(f^j), \text{where } j = 0, 1, ..., J. \tag{5}$$

3.3 Convolutional Networks as Local Feature Encoder

We empirically find that only using the global information from ViT compromises the rendering quality of target views that are close to the input view, e.g., the color and appearance are inconsistent (see Fig. 12). To alleviate this problem, we introduce an additional 2D CNN module $\mathcal{G}_L$ to extract local image features, which can improve the color and appearance consistency in the visible regions. The local features can be represented as

$$\mathbf{W}_L = \mathcal{G}_L(\mathbf{I}_s), \mathcal{G}_L : \mathbb{R}^{H\times W\times C} \rightarrow \mathbb{R}^{\frac{H}{2}\times\frac{W}{2}\times D_L}, \tag{6}$$

where D_L is the output dimension of $\mathcal{G}_L$.

Finally, we use a convolutional layer $\mathcal{G}$ to fuse the information from both global feature $\mathbf{W}_G$ and local feature $\mathbf{W}_L$ and generate the hybrid feature map:

$$\mathbf{W} = \mathcal{G}(\mathbf{W}_G^0, \mathbf{W}_G^1, ..., \mathbf{W}_G^J; \mathbf{W}_L) \tag{7}$$

3.4 Volumetric Rendering with NeRF

Once we obtain the hybrid features $\mathbf{W}$, we can adopt the volumetric rendering [29] to render a target view conditioned on $\mathbf{W}$. We start by sampling a ray $\mathbf{r}(t) = \mathbf{o} + t\mathbf{d}$ from the target viewpoint, where $\mathbf{o}$ is the origin of the ray, $\mathbf{d}$ is the ray direction, and t is the distance from the origin. Note that t is bounded by

near plane t_{near} and far plane t_{far}. Along the ray, we first pick equally distributed samples between the bounds $[t_{\text{near}}, t_{\text{far}}]$. We denote a 3D sample location as $\mathbf{x}$, which can be projected onto the source image with coordinate $\pi(\mathbf{x})$ with known camera parameters. We then extract the per-pixel feature as $\mathbf{W}(\pi(\mathbf{x}))$. The NeRF MLP module takes as input the per-pixel feature $\mathbf{W}(\pi(\mathbf{x}))$, 3D sample location in camera coordinate $\mathbf{x}_c$ and viewing direction $\mathbf{d}_c$. Note that we encode $\mathbf{x}_c$ with positional encoding γ:

$$\gamma(p) = (\sin(2^0\pi p), \cos(2^0\pi p), ..., \sin(2^{M-1}\pi p), \cos(2^{M-1}\pi p)), \tag{8}$$

where M is the number of frequency bases. We set $M = 10$ in all our experiments. The MLP outputs color $\mathbf{c}$ and density σ, which can be written as:

$$(\sigma, \mathbf{c}) = \text{MLP}(\gamma(\mathbf{x}_c); \mathbf{d}_c; \mathbf{W}(\pi(\mathbf{x}))). \tag{9}$$

Finally, we render the target view into a 2D image via

$$\hat{\mathbf{C}}(\mathbf{r}) = \int_{t_n}^{t_f} T(t)\sigma(t)\mathbf{c}(t)dt, \tag{10}$$

where $T(t) = \exp(-\int_{t_n}^{t} \sigma(\mathbf{r}(s))ds)$ is the accumulated transmittance along the ray from t_n to t. Here we approximate the integral with quadrature [29].

We adopt a L2 norm loss to compare the rendered pixel $\hat{\mathbf{C}}(\mathbf{r})$ against the ground-truth pixel:

$$\mathcal{L} = \sum_{\mathbf{r}} ||\hat{\mathbf{C}}(\mathbf{r}) - \mathbf{C}(\mathbf{r})||_2^2. \tag{11}$$

Implementation Details. We implement our method using PyTorch [34]. The ViT module is initialized from the pretrained weights of [51] and fine-tuned with the training. The 2D CNN module $\mathcal{G}_L$ has three ResBlocks. The detailed architecture of the entire model is provided in the supplementary material. We train our model on 16 NVIDIA A100 GPUs, where the training converges at 500K iterations. We set the learning rate to be $1e-4$ for the MLP and $1e-5$ for ViT and the CNN. To improve training stability, we use a linear warm-up schedule to increase the learning rate linearly from 0 for the first 10k steps (please see our supplementary material for more details). As NeRF is trained with batches of rays, we use 512 rays for 1 instance and the batch size of 8. We will release our source code upon acceptance.

4 Experimental Results

To evaluate our method, we conduct experiments on category-specific view synthesis (Sec. 4.1) and category-agnostic view synthesis (Sec. 4.2). Furthermore, we show the qualitative results of our method on real input images (Sec. 4.3). We also provide ablation studies to analyze the key components in our method (Sec. 4.4). Finally, we discuss the limitations and potential future work (Sec. 4.5).

Fig. 5. Category-specific view synthesis on Chairs. The results of SRN and PixelNeRF are often too blurry, especially on the legs that are not visible in the input views. Our method can generate novel views with clearer structures and sharper edges.

4.1 Category-specific View Synthesis

In this section, we evaluate our method on the same experimental setup and data as SRN [43]. The dataset consists of 6591 chairs and 3514 cars in total, which are split into training, validation, and test sets. For each object in the training set, 50 views lying on a sphere around the object are selected to render with simple lighting. For testing, the objects in the test set are rendered from 251 views on an archimedean spiral with the same illumination as training. During the evaluation, the 64-th view is selected as the input view and all other 250 views are used as target views. All images are in 128×128 resolution. We compare our method with SRN [43], PixelNeRF [55][5], CodeNeRF [18][6] and FE-NVS [15][7].

As shown in Table 2, our method achieves state-of-the-art performance against existing approaches in terms of PSNR, SSIM, and LPIPS [56]. On the chair dataset, our method shows significant improvement on all three metrics. As shown in Fig. 5, our rendered results have better appearance and clearer structures, while SRN [43] and PixelNeRF [55] have blurry predictions on the chair legs. On the car dataset, we obtain the best LPIPS and SSIM scores. While PixelNeRF [55] has the highest PSNR, their results are overly-blurry with smooth textures, as shown in Fig. 6. In contrast, our predictions have finer details and reveal more details such as the windows, lights, and wheels. Note that we do not compare visual results with CodeNeRF [18] as their pre-generated results are not publicly available, and their source code does not support inference without camera poses. FE-NVS [15] does not provide source code or pre-generate results

[5] LPIPS is calculated from the results provided by the authors on request.
[6] LPIPS and code for unposed inference are not available.
[7] LPIPS is provided by the authors on request.

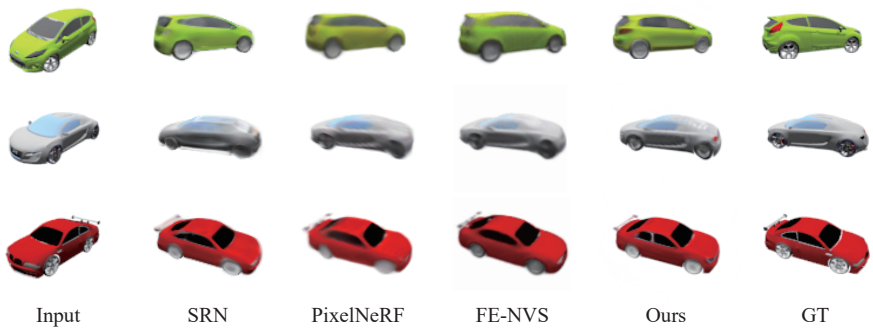

Fig. 6. Category-specific view synthesis on Cars. Our method can generate sharper car structure and richer details, such as the rear lights and windows in the first row, the wheels and door in the second row, and the windows in the third row.

Table 2. Category-specific view synthesis on the ShapeNet dataset. Our method performs favorably against other approaches, especially on LPIPS. Note that while PixelNeRF has higher PSNR on the cars dataset, their results look blurry (see Fig. 6).

	Chairs			Cars		
Methods	PSNR(↑)	SSIM(↑)	LPIPS(↓)	PSNR(↑)	SSIM(↑)	LPIPS(↓)
SRN [43]	22.89	0.89	0.104	22.25	0.89	0.129
PixelNeRF [55]	23.72	0.91	0.128	**23.17**	0.90	0.146
CodeNeRF [18]	22.39	0.87	0.166	22.73	0.89	0.128
FE-NVS [15]	23.21	0.92	0.077	22.83	**0.91**	0.099
Ours	**24.48**	**0.93**	**0.062**	22.88	**0.91**	**0.095**

as well. However, we try our best to obtain high-resolution screenshots from their paper and compare with their results on the same view.

4.2 Category-agnostic View Synthesis

Our method is able to generalize across multiple object categories using a single model. We follow the training/test splits of the ShapeNet dataset defined in NMR [21] and choose 1 view as input while the other 23 views as target in both training and evaluation. There are 30642 objects for training and 8762 objects for evaluation (from 13 categories). The image resolution is 64×64.

Table 3 shows the quantitative results, where our method achieves the state-of-the-art performance against SRN [43], PixelNeRF [55], and FE-NVS [15] on all 13 categories. The results demonstrate that our hybrid representation is more expressive than the locally-conditioned models or 3D voxel methods. The visual comparisons in Fig. 7 also show that our method reconstructs finer object structure and details. More visual comparisons are provided in the supplementary material.

Table 3. Category-agnostic view synthesis on the NMR dataset. Our method achieves the state-of-the-art performance across all 13 categories using a single model.

Metrics	Methods	plane	bench	cbnt.	car	chair	disp.	lamp	spkr.	rifle	sofa	table	phone	boat	average
PSNR(↑)	SRN	26.62	22.20	23.42	24.40	21.85	19.07	22.17	21.04	24.95	23.65	22.45	20.87	25.86	23.28
	PixelNeRF	29.76	26.35	27.72	27.58	23.84	24.22	28.58	24.44	30.60	26.94	25.59	27.13	29.18	26.80
	FE-NVS	30.15	27.01	28.77	27.74	24.13	24.13	28.19	24.85	30.23	27.32	26.18	27.25	28.91	27.08
	Ours	**32.34**	**29.15**	**31.01**	**29.51**	**25.41**	**25.77**	**29.41**	**26.09**	**31.83**	**28.89**	**27.96**	**29.21**	**30.31**	**28.76**
SSIM(↑)	SRN	0.901	0.837	0.831	0.897	0.814	0.744	0.801	0.779	0.913	0.851	0.828	0.811	0.898	0.849
	PixelNeRF	0.947	0.911	0.910	0.942	0.858	0.867	0.913	0.855	0.968	0.908	0.898	0.922	0.939	0.910
	FE-NVS	0.957	0.930	0.925	0.948	0.877	0.871	0.916	0.869	0.970	0.920	0.914	0.926	0.941	0.920
	Ours	**0.965**	**0.944**	**0.937**	**0.958**	**0.892**	**0.891**	**0.925**	**0.877**	**0.974**	**0.930**	**0.929**	**0.936**	**0.950**	**0.933**
LPIPS(↓)	SRN	0.111	0.150	0.147	0.115	0.152	0.197	0.210	0.178	0.111	0.129	0.135	0.165	0.134	0.139
	PixelNeRF	0.084	0.116	0.105	0.095	0.146	0.129	0.114	0.141	0.066	0.116	0.098	0.097	0.111	0.108
	FE-NVS	0.061	0.080	0.076	0.085	0.103	0.105	0.091	0.116	0.048	0.081	0.071	0.080	0.094	0.082
	Ours	**0.042**	**0.067**	**0.065**	**0.059**	**0.084**	**0.086**	**0.073**	**0.103**	**0.046**	**0.068**	**0.055**	**0.068**	**0.072**	**0.065**

Table 4. Ablation studies. We start from a baseline model that uses ViT to extract global features. While PSNR/SSIM are slightly lower than PixelNeRF, our results have much better LPIPS scores and sharper details (see Figure 1). By using a 3-layer CNN to extract local features, our performance on the car dataset is improved, and the rendered images have more faithful appearances to the input views (see Figure 12). By adding the viewing direction in volume rendering, the performance is improved significantly. Finally, by replacing the 3-layer CNN with ResBlocks, we see more fine details and better object structure in Figure 12.

	Chairs			Cars		
Method	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
PixelNeRF	23.72	0.91	0.128	23.17	0.90	0.146
Baseline (ViT only)	23.45	0.92	0.076	21.95	0.89	0.113
+ $\mathcal{G}_L$ (3-layer CNN)	23.42	0.92	0.075	22.42	0.90	0.103
+ Viewing Direction	**24.53**	**0.93**	**0.062**	22.70	**0.91**	0.097
ResBlocks for $\mathcal{G}_L$	24.48	**0.93**	**0.062**	**22.88**	**0.91**	**0.095**

4.3 View Synthesis on Real Images

Our method can also generalize to real images. We use our model trained on the ShapeNet car dataset to test on real car images from the Stanford cars dataset [23]. We use an image segmentation model [22] to remove the background. Note that our method does not require any camera pose as input, which is often difficult to obtain from real images. We compare our results with PixelNeRF in Fig. 8. In the occluded regions, PixelNeRF suffers from blurry predictions as pointed out by the arrows. In contrast, our method is able to reconstruct the entire shape and keep details such as headlights and side mirrors.

4.4 Ablation Studies

We conduct ablation studies to validate the efficacy of key components in our method. We start from the baseline method that uses the ViT to extract global features only. While ViT encodes the high-level global information, it may not preserve the color and appearance from the input view due to the low-resolution

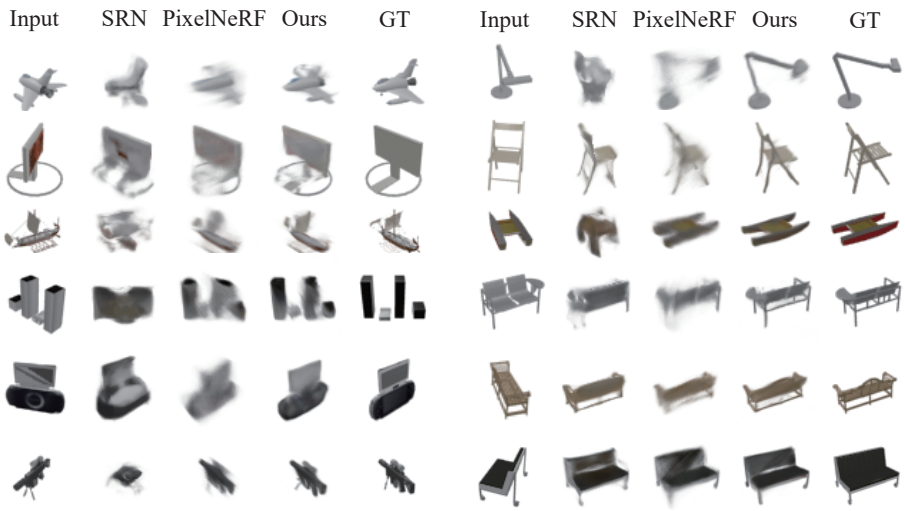

Fig. 7. Visual comparison of category-agnostic view synthesis. The results of SRN [43] and PixelNeRF [55] are often too blurry and contain smearing artifacts. In contrast, our results are sharper with more fine details. The visual results of all 13 categories are provided in the supplementary material.

latent embeddings. The rendered results may not have consistent appearances to the input view on non-occluded regions, as shown in the second column in Fig. 12. By introducing G_L (using a simple 3-layer CNN) to extract local image features, the rendered car looks closer to the input view (top of the third column in Fig. 12). However, we can see that the chair's back is still blurry (bottom of the third column in Fig. 12). Next, we add the viewing direction as input to the NeRF MLP, which significantly improves the sharpness (bottom of the 4-th column in Fig. 12) and reveals more details such as the rear mirror of the car (top of the 4-th column in Fig. 12). Finally, we adopt a more complex ResBlocks design in G_L, which further improves the geometry shape of the car and chair (the 5-th column in Fig. 12). Table 4 also reports the quantitative results of these design decisions on both datasets.

4.5 Limitations and Future Work

First, our method does not utilize geometry priors such as symmetry [52]. For example, in the car dataset, some decals on the car are symmetrical and can be reused for the unseen side. However, it remains a question on how to select the symmetry plane or find the canonical space for such a prior. Another limitation is that we do not fully utilize the high-level semantics of the objects. A semantic understanding on the smaller components could help reconstruct the unseen areas much better. For example, a car has four wheels. Given the partial observation, it is possible to use semantic knowledge to recover the unseen

Fig. 8. Results on real input images. Our method is able to generate visually-pleasing results even trained on a synthetic dataset. Conversely, PixelNeRF fails to keep the finer details. Note the side mirrors and headlamps of the bottom right inset.

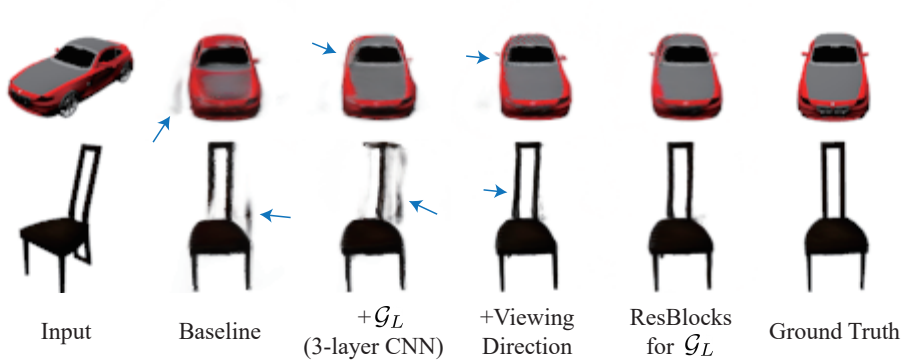

Fig. 9. Effects of different components. The baseline model (ViT only) can render realistic images, but the local appearance and color may not look similar to the input view. By extracting local features with a 3-layer CNN, the rendered car shows more faithful colors to the input. With the viewing direction in volume rendering, our model can improve fine structures such as the left mirror of the car and the back of the chair. Finally, replacing the 3-layer CNN with ResBlocks can further refine the details and geometry structure of the rendered objects.

components. Lastly, GAN-based methods can be helpful in generating texture in occluded parts of the object. Integrating locally-conditioned models with GAN loss training remains a challenging problem for future research.

5 Conclusions

In this work, we present a NeRF-based algorithm for novel view synthesis from a single unposed image. We utilize vision transformer in conjunction with convolutional networks to extract global and local features as 3D representations. This hybrid representation shows promising performance in recovering the occluded shapes and appearance. Additionally, we show that vision transformer can be used to generate global information without enforcing a canonical coordinate system (which requires camera pose estimation). We believe that our work has shed light on future research to synthesize faithful 3D content with local and global image features, and we hope that it could bring about more exciting advances in the frontier for immersive 3D content.

6 Acknowledgement

This work was supported in part by a Qualcomm FMA Fellowship, ONR grant N000142012529, ONR grant N000141912293, NSF grant 1730158, a grant from Google, and an Amazon Research award. We also acknowledge gifts from Adobe, the Ronald L. Graham Chair, and the UC San Diego Center for Visual Computing.

Supplementary Material

A Video Results

We include video results as a webpage (`index.html`) in the supplementary file. In each video, we compare the input, SRN [43], PixelNeRF [55], ours and ground-truth. The renderings of SRN and PixelNeRF are provided by the authors of PixelNeRF on request.

B Network Architecture

Transformer encoder. We adopt the pretrained ViT-b16 model from Wightman [51] as our transformer encoder. The transformer encoder has 12 layers (J=12), where each layer uses the LayerNorm [2] for normalization and GELU [16] for activation. The positional embedding is resized to the same size as the input image (e.g., 128×128 for the ShapeNet dataset, and 64×64 for the NMR dataset).

Convolutional decoder. We provide the architecture details of our convolutional decoder in Table 5. Our convolutional decoder takes the tokens f^3, f^6, f^9, and f^{12} [36] as input and generate multi-level features $\mathbf{W}_G^3, \mathbf{W}_G^6, \mathbf{W}_G^9, \mathbf{W}_G^{12}$, as shown in Fig. 4 of the main paper. We then resize all feature maps $\mathbf{W}_G$ to the size of $\frac{H}{2} \times \frac{W}{2}$ via bilinear interpolation. Finally, we adopt a two-layer CNN module (see Table 6) to generate the global feature representation $\mathbf{W}'_G$.

Table 5. Architecture of the convolutional decoder. Conv denotes convolution layer. **TransConv** denotes transposed convolution layer.

$\#j$	layer	kernel	stride	dilation	in	out	activation	input
3	Conv0-0	1×1	1	1	768	96	N/A	f^3
	TransConv0-1	4×4	4	1	96	96	N/A	Conv0-0
	Conv0-2	3×3	1	1	96	512	N/A	TransConv0-1
6	Conv1-0	1×1	1	1	768	192	N/A	f^6
	TransConv1-1	2×2	2	1	192	192	N/A	Conv1-0
	Conv1-2	3×3	1	1	192	512	N/A	TransConv1-1
9	Conv2-0	1×1	1	1	768	384	N/A	f^9
	Conv2-1	3×3	1	1	384	512	N/A	Conv2-0
12	Conv3-0	1×1	1	1	768	768	N/A	f^{12}
	Conv3-1	3×3	2	1	768	768	N/A	Conv3-0
	Conv3-2	3×3	1	1	768	512	N/A	Conv3-1

2D CNN $\mathcal{G}_L$. We use the ResBlocks in Fig. 10 to generate the local feature representation $\mathbf{W}_L$. The local feature $\mathbf{W}_L$ and global feature $\mathbf{W}'_G$ are then concatenated as our hybrid feature representation $\mathbf{W}$.

Table 6. Architecture of the last two layers for generating global feature representation $\mathbf{W}'_G$.

Layer	kernel	stride	dilation	in	out	activation	input
Conv0	3×3	1	1	1024	512	ReLU	$\mathbf{W}_G$
Conv1	3×3	1	1	512	256	ReLU	conv0

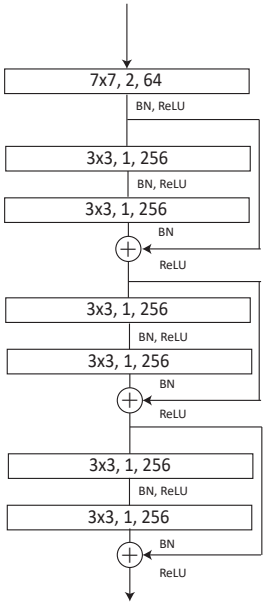

Fig. 10. Illustration of the ResBlocks. We use three ResBlocks as our 2D CNN module to extract local feature representation from the input image. The numbers in each layer denote the kernel size, stride, and output channels, respectively. BN denotes BatchNorm, and ReLU is the nonlinear activation function.

NeRF MLP. We utilize the hierarchical rendering [29] to include detailed geometry and appearance. We use 6 ResNet blocks with width 512 for both coarse and fine stages to process the global and local features.

C Implementation Details

Learning rate. We set the initial learning rate to be $1e-4$ for the MLP and $1e-5$ for ViT and the 2D CNN module. To improve training stability, we use a warm-up schedule to increase the learning rate linearly from 0 for the first 10k steps. We decay the learning rate decay by a factor of 0.1 at 450k steps. Our learning rate schedule for the NeRF MLP is plotted in Fig. 11.

Sampling strategy. We use a different bounding box sampling strategy than PixelNeRF's design. In PixelNeRF, only foreground pixels are sampled during

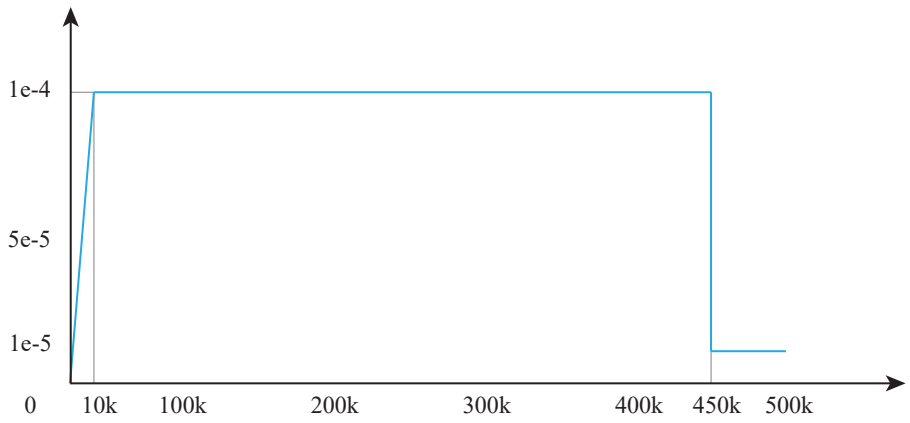

Fig. 11. Illustration of the learning rate strategy for the MLP module. We first linearly increase the learning rate to $1e-4$ for the first 10k step. Then, we set a learning rate decay of scale 0.1 at 450k steps.

the training, while the background pixels are ignored. We find that such a sampling strategy makes training unstable, especially for the ViT module. To solve this issue, we take 20% of the samples from the entire image (including background) and 80% of the samples within a bounding box of the valid pixels. We show a rendered result with our modified sampling strategy in Fig. 12.

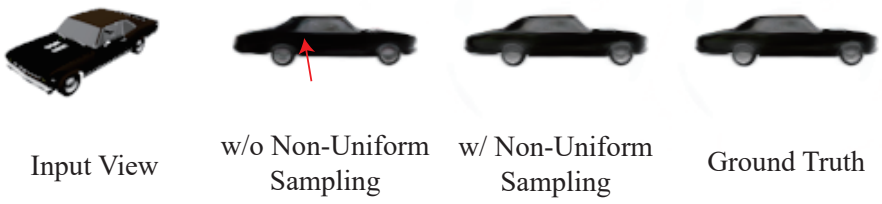

Fig. 12. Ablation studies on sampling strategy.

D Experiment Configurations

We provide the detailed experiment setups in Sec. 4 of the main paper.

D.1 Category-specific View Synthesis.

SRN [43] and PixelNeRF [55]: We obtain the PSNR and SSIM from Table 2 in [55]. We use the pre-generated results provided by the authors of [55] (on request) to calculate the LPIPS score.

CodeNeRF [18]: We obtain the PSNR and SSIM of the unposed input from Table 2 in [18]. As the pre-generated results are not available, and the source code of CodeNeRF does not provide optimization stages for unposed inputs, we are not able to calculate the LPIPS score.

FE-NVS [15]: We obtain the PSNR and SSIM of the unposed input from Table 1 in [15]. We obtain the LPIPS score from the authors on request. As FE-NVS does not provide source code to reproduce any qualitative results, we crop the high-resolution images from their paper to show the comparison in Fig. 6 of the main paper.

Note that all the methods except CodeNeRF uses view 64 as input, while CodeNeRF uses view 82 as input for evaluation in their paper. As the source code for CodeNeRF does not include unposed inference, we are not able to generate the full evaluation using view 64.

D.2 Category-agnostic View Synthesis

Similar to category-specific view synthesis, we obtain the numbers of SRN [43] and PixelNeRF [55] from Table 4 of [55]. We obtain the numbers of FE-NVS [15] from Table 4 of [15]. Qualitative results for SRN and PixelNeRF are generated using the pre-generated results from [55].

D.3 View Synthesis on Real Images

We use the real car images provided by PixelNeRF and the Stanford Cars dataset [23] for evaluation. We remove the background using the PointRend [22] segmentation and resize the images to 128×128 resolution. We assume the input camera pose is an identity matrix and apply rotation matrices to simulate 360° views. We use the source code and pre-trained model of PixelNeRF to generate the results for PixelNeRF, where we synthesize the renderings at the same camera poses.

E Additional Visual Results

We include extra results on category-specific view synthesis in Fig. 13 and 14, results on category-agnostic view synthesis in Fig. 15-21. Note that we choose the target views where most pixels are not visible in the input view to better compare the rendering quality of each method on occluded regions. The video results (in GIF format) are also provided in the supplementary files.

Fig. 13. Category-specific view synthesis results on cars.

Fig. 14. Category-specific view synthesis results on chairs.

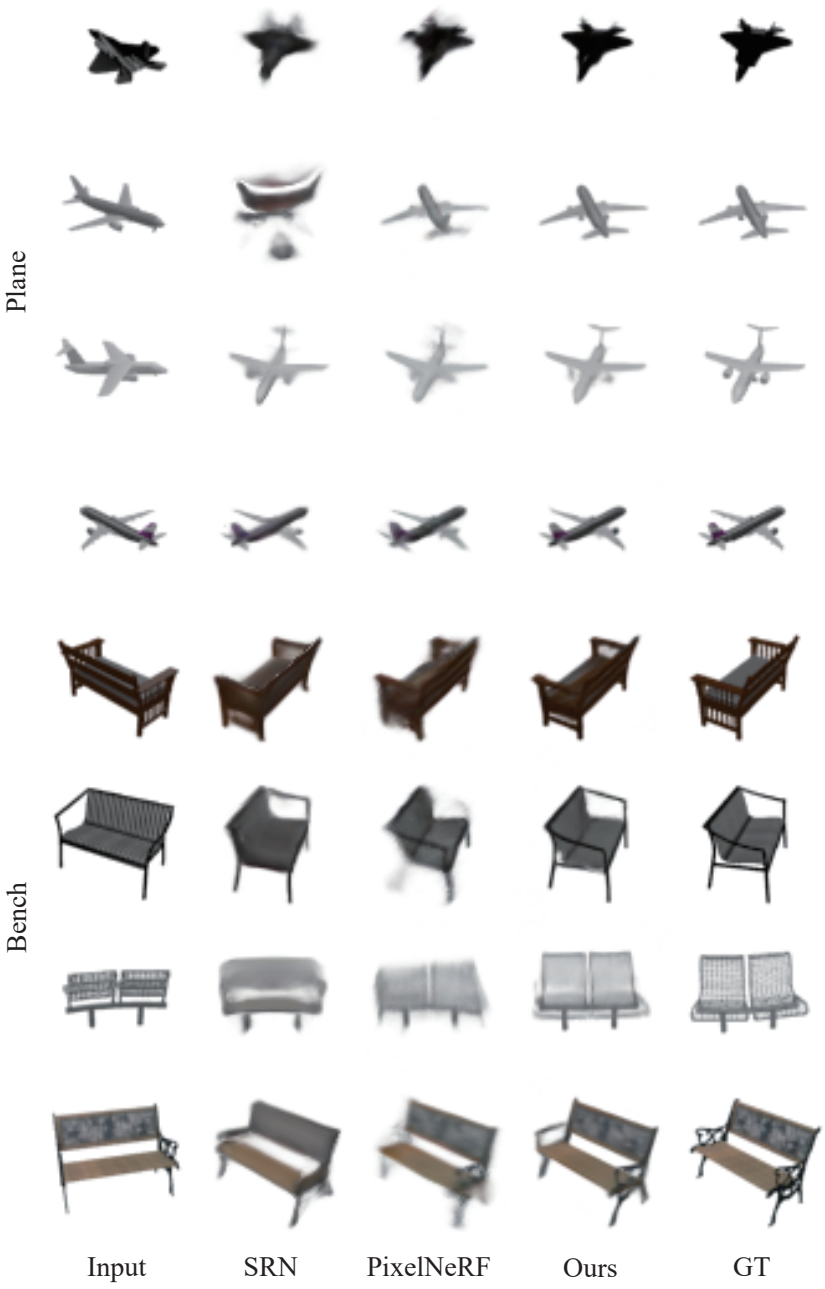

Fig. 15. Category-agnostic view synthesis results on the NMR dataset.

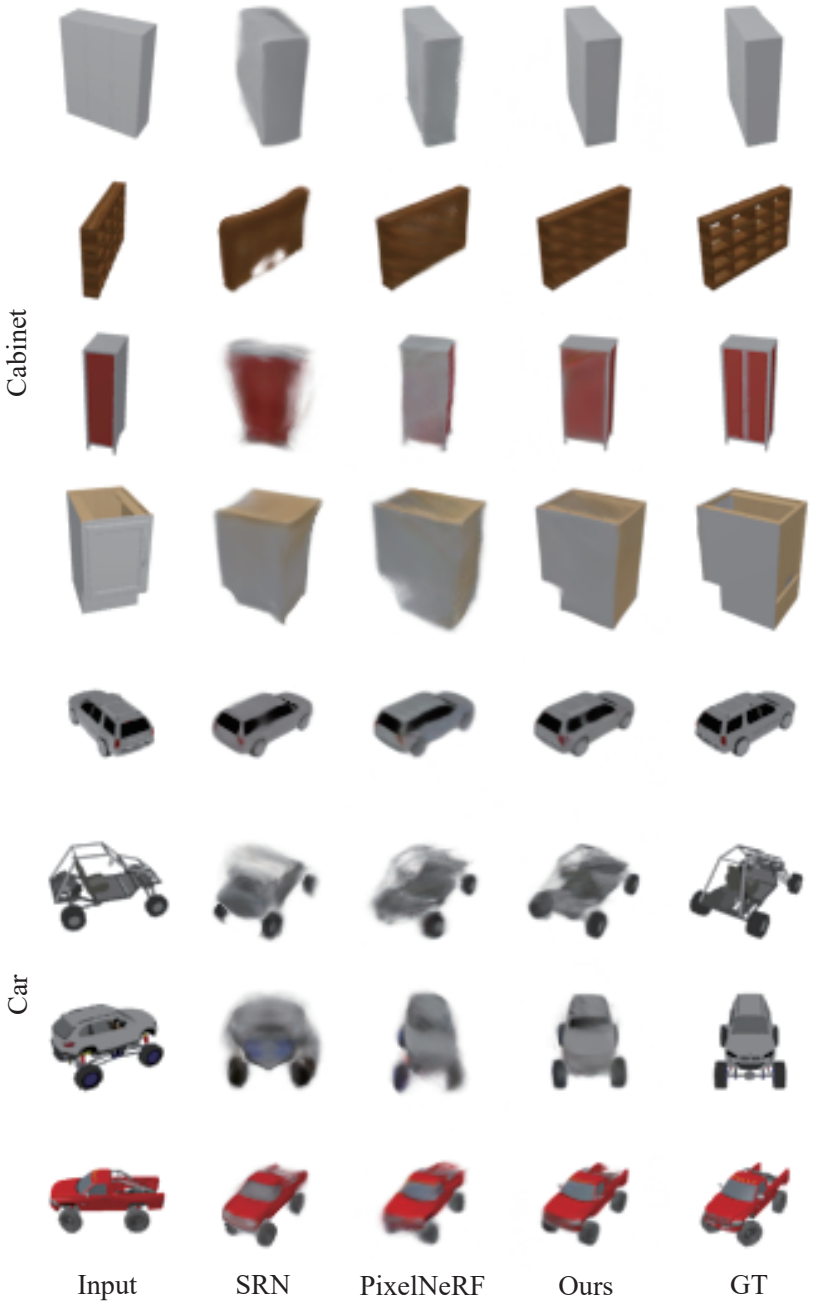

Fig. 16. Category-agnostic view synthesis results on the NMR dataset.

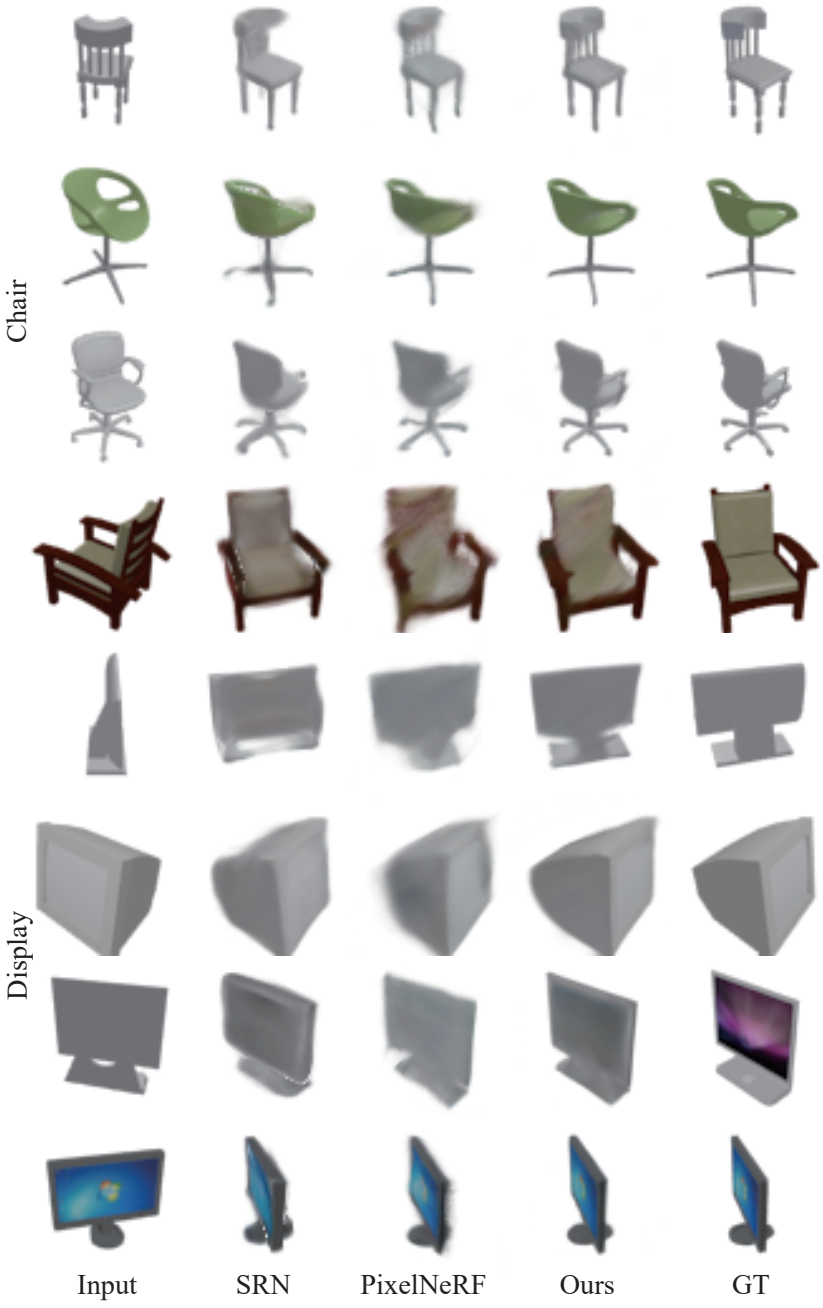

Fig. 17. Category-agnostic view synthesis results on the NMR dataset.

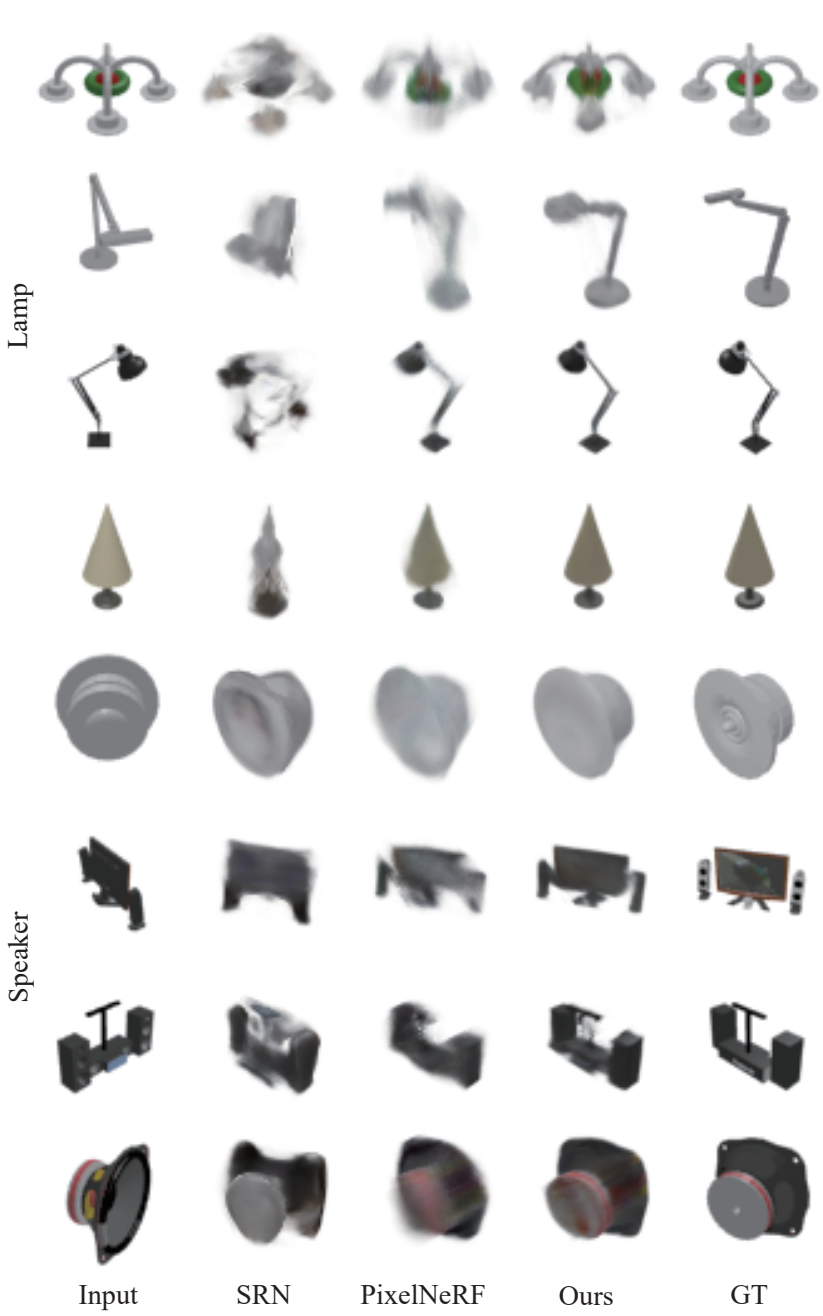

Fig. 18. Category-agnostic view synthesis results on the NMR dataset.

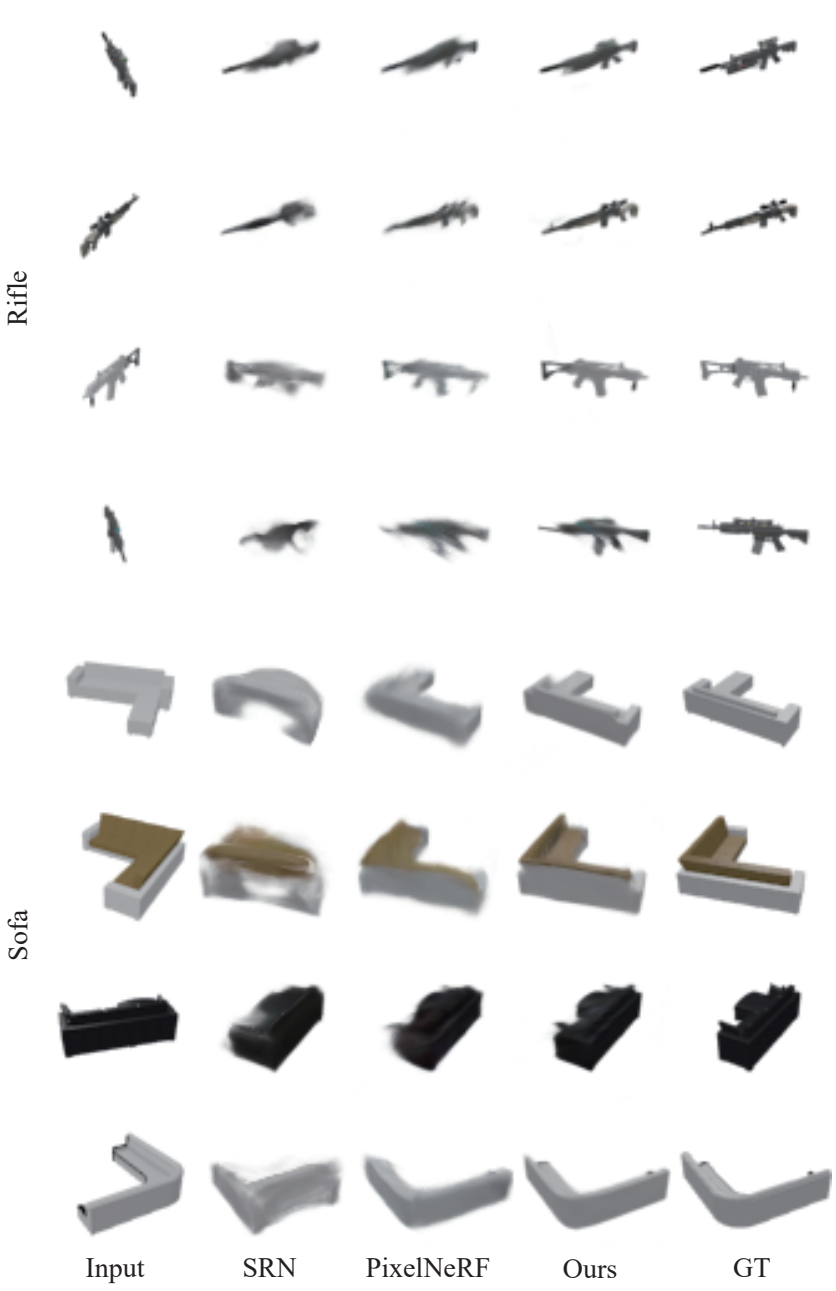

Fig. 19. Category-agnostic view synthesis results on the NMR dataset.

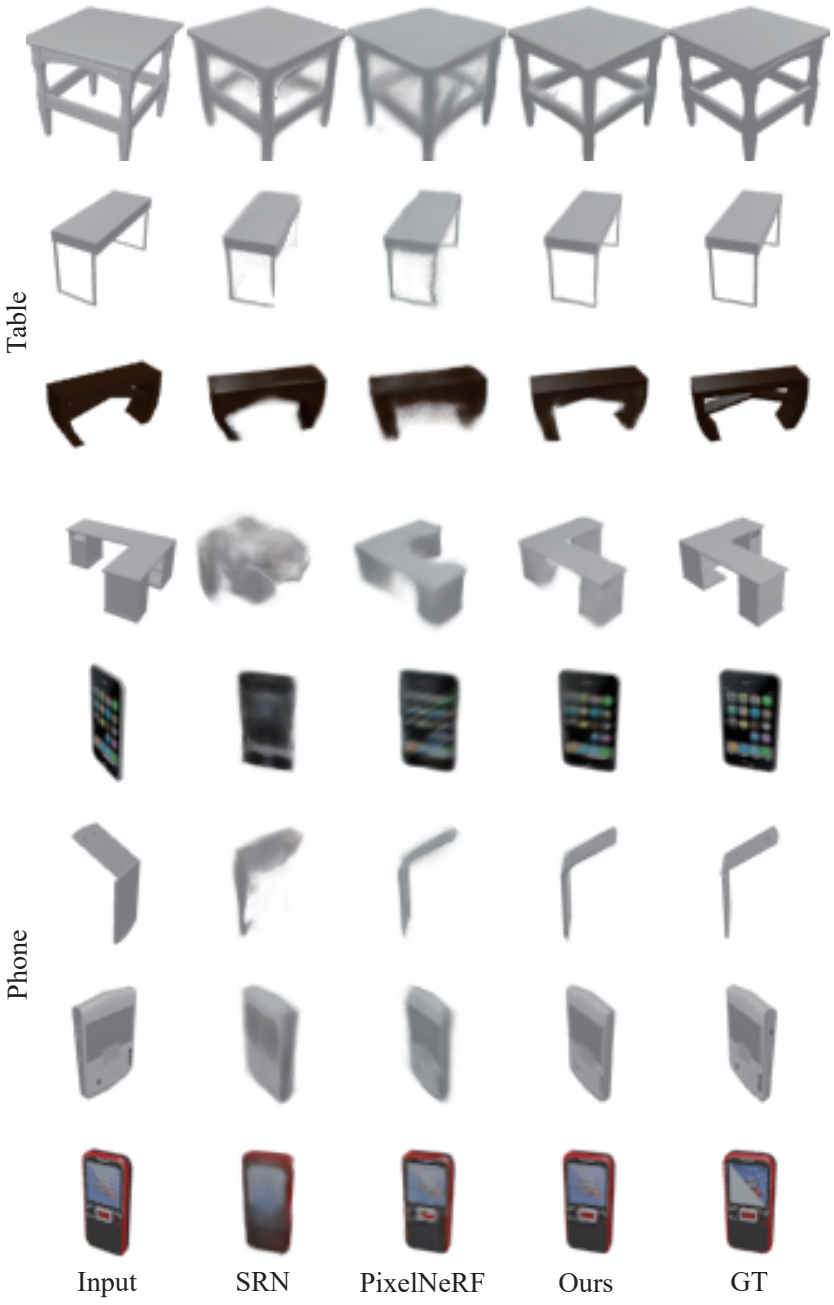

Fig. 20. Category-agnostic view synthesis results on the NMR dataset.

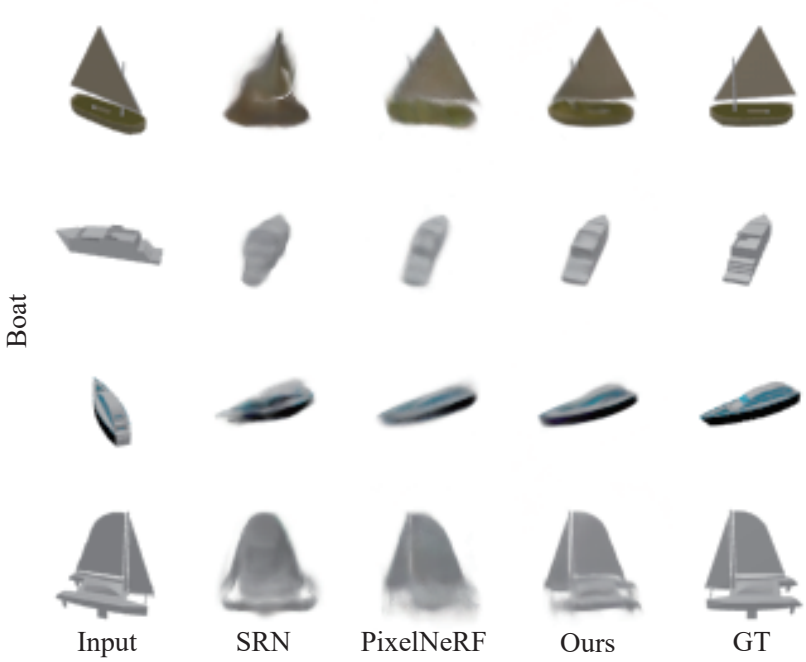

Fig. 21. Category-agnostic view synthesis results on the NMR dataset.

References

1. Arandjelović, R., Zisserman, A.: NeRF in detail: Learning to sample for view synthesis. arXiv:2106.05264 (2021)
2. Ba, J.L., Kiros, J.R., Hinton, G.E.: Layer normalization. arXiv preprint arXiv:1607.06450 (2016)
3. Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J.D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A.: Language models are few-shot learners. In: NeurIPS (2020)
4. Buehler, C., Bosse, M., McMillan, L., Gortler, S., Cohen, M.: Unstructured lumigraph rendering. In: Proceedings of the 28th annual conference on Computer graphics and interactive techniques (2001)
5. Chang, A.X., Funkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., Su, H., Xiao, J., Yi, L., Yu, F.: ShapeNet: An Information-Rich 3D Model Repository. arXiv:1512.03012 (2015)
6. Chen, S.E., Williams, L.: View interpolation for image synthesis. In: Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques (1993)
7. Chen, Z., Zhang, H.: Learning implicit fields for generative shape modeling (2019)
8. Debevec, P.E., Taylor, C.J., Malik, J.: Modeling and rendering architecture from photographs: A hybrid geometry-and image-based approach. In: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques (1996)
9. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: BERT: Pre-training of deep bidirectional transformers for language understanding. arXiv:1810.04805 (2018)
10. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., Houlsby, N.: An image is worth 16x16 words: Transformers for image recognition at scale. In: ICLR (2021)
11. Dupont, E., Miguel Angel, B., Colburn, A., Sankar, A., Guestrin, C., Susskind, J., Shan, Q.: Equivariant neural rendering. In: ICML (2020)
12. Flynn, J., Broxton, M., Debevec, P., DuVall, M., Fyffe, G., Overbeck, R., Snavely, N., Tucker, R.: DeepView: View synthesis with learned gradient descent. In: CVPR (2019)
13. Flynn, J., Neulander, I., Philbin, J., Snavely, N.: DeepStereo: Learning to predict new views from the world's imagery. In: ICCV (2016)
14. Gortler, S.J., Grzeszczuk, R., Szeliski, R., Cohen, M.F.: The lumigraph. In: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques (1996)
15. Guo, P., Bautista, M.A., Colburn, A., Yang, L., Ulbricht, D., Susskind, J.M., Shan, Q.: Fast and explicit neural view synthesis. In: WACV (2022)
16. Hendrycks, D., Gimpel, K.: Gaussian error linear units (gelus). arXiv preprint arXiv:1606.08415 (2016)
17. Hu, H., Zhang, Z., Xie, Z., Lin, S.: Local relation networks for image recognition. In: ICCV (2019)
18. Jang, W., Agapito, L.: CodeNeRF: Disentangled neural radiance fields for object categories. In: ICCV (2021)
19. Johari, M.M., Lepoittevin, Y., Fleuret, F.: GeoNeRF: Generalizing nerf with geometry priors. arXiv:2111.13539 (2021)
20. Kalantari, N.K., Wang, T.C., Ramamoorthi, R.: Learning-based view synthesis for light field cameras. ACM TOG **35**(6), 1–10 (2016)

21. Kato, H., Ushiku, Y., Harada, T.: Neural 3D mesh renderer. In: CVPR (2018)
22. Kirillov, A., Wu, Y., He, K., Girshick, R.: PointRend: Image segmentation as rendering. In: CVPR (2020)
23. Krause, J., Stark, M., Deng, J., Fei-Fei, L.: 3D object representations for fine-grained categorization. In: 4th International IEEE Workshop on 3D Representation and Recognition (3dRR-13). Sydney, Australia (2013)
24. Levoy, M., Hanrahan, P.: Light field rendering. In: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques (1996)
25. Lin, K.E., Xiao, L., Liu, F., Yang, G., Ramamoorthi, R.: Deep 3D mask volume for view synthesis of dynamic scenes. In: ICCV (2021)
26. Lombardi, S., Simon, T., Saragih, J., Schwartz, G., Lehrmann, A., Sheikh, Y.: Neural volumes: Learning dynamic renderable volumes from images. ACM TOG **38**(4) (2019)
27. Mescheder, L., Oechsle, M., Niemeyer, M., Nowozin, S., Geiger, A.: Occupancy networks: Learning 3D reconstruction in function space. In: CVPR (2019)
28. Mildenhall, B., Srinivasan, P.P., Ortiz-Cayon, R., Kalantari, N.K., Ramamoorthi, R., Ng, R., Kar, A.: Local light field fusion: Practical view synthesis with prescriptive sampling guidelines. ACM TOG (2019)
29. Mildenhall, B., Srinivasan, P.P., Tancik, M., Barron, J.T., Ramamoorthi, R., Ng, R.: NeRF: Representing scenes as neural radiance fields for view synthesis. In: ECCV (2020)
30. Niemeyer, M., Mescheder, L., Oechsle, M., Geiger, A.: Differentiable volumetric rendering: Learning implicit 3D representations without 3D supervision. In: CVPR (2020)
31. Niklaus, S., Mai, L., Yang, J., Liu, F.: 3D ken burns effect from a single image. ACM TOG **38**(6), 1–15 (2019)
32. Park, J.J., Florence, P., Straub, J., Newcombe, R., Lovegrove, S.: Deepsdf: Learning continuous signed distance functions for shape representation. In: CVPR (2019)
33. Parmar, N., Vaswani, A., Uszkoreit, J., Kaiser, L., Shazeer, N., Ku, A., Tran, D.: Image transformer. In: ICML (2018)
34. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., Chintala, S.: Pytorch: An imperative style, high-performance deep learning library. In: NeurIPS (2019)
35. Ramachandran, P., Parmar, N., Vaswani, A., Bello, I., Levskaya, A., Shlens, J.: Stand-alone self-attention in vision models. In: NeurIPS (2019)
36. Ranftl, R., Bochkovskiy, A., Koltun, V.: Vision transformers for dense prediction. In: ICCV (2021)
37. Reizenstein, J., Shapovalov, R., Henzler, P., Sbordone, L., Labatut, P., Novotny, D.: Common objects in 3D: Large-scale learning and evaluation of real-life 3D category reconstruction. In: ICCV (2021)
38. Rematas, K., Martin-Brualla, R., Ferrari, V.: Sharf: Shape-conditioned radiance fields from a single view. arXiv:2102.08860 (2021)
39. Saito, S., , Huang, Z., Natsume, R., Morishima, S., Kanazawa, A., Li, H.: Pifu: Pixel-aligned implicit function for high-resolution clothed human digitization. arXiv:1905.05172 (2019)
40. Shade, J., Gortler, S., He, L.w., Szeliski, R.: Layered depth images. In: Proceedings of the 25th annual conference on Computer graphics and interactive techniques (1998)

41. Shih, M.L., Su, S.Y., Kopf, J., Huang, J.B.: 3D photography using context-aware layered depth inpainting. In: CVPR (2020)
42. Sitzmann, V., Thies, J., Heide, F., Nießner, M., Wetzstein, G., Zollhofer, M.: DeepVoxels: Learning persistent 3D feature embeddings. In: CVPR (2019)
43. Sitzmann, V., Zollhöfer, M., Wetzstein, G.: Scene representation networks: Continuous 3D-structure-aware neural scene representations. In: NeurIPS (2019)
44. Strudel, R., Garcia, R., Laptev, I., Schmid, C.: Segmenter: Transformer for semantic segmentation. In: ICCV (2021)
45. Szeliski, R., Golland, P.: Stereo matching with transparency and matting. vol. 32, pp. 45–61 (July 1999)
46. Trevithick, A., Yang, B.: GRF: Learning a general radiance field for 3D scene representation and rendering. In: arXiv:2010.04595 (2020)
47. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L.u., Polosukhin, I.: In: Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R. (eds.) NeurIPS (2017)
48. Wang, D., Cui, X., Chen, X., Zou, Z., Shi, T., Salcudean, S., Wang, Z.J., Ward, R.: Multi-view 3D reconstruction with transformers. In: ICCV (2021)
49. Wang, Q., Wang, Z., Genova, K., Srinivasan, P., Zhou, H., Barron, J.T., Martin-Brualla, R., Snavely, N., Funkhouser, T.: IBRNet: Learning multi-view image-based rendering. In: CVPR (2021)
50. Wang, W., Xie, E., Li, X., Fan, D.P., Song, K., Liang, D., Lu, T., Luo, P., Shao, L.: Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. In: ICCV (2021)
51. Wightman, R.: Pytorch image models. `https://github.com/rwightman/pytorch-image-models` (2019)
52. Wu, S., Rupprecht, C., Vedaldi, A.: Unsupervised learning of probably symmetric deformable 3D objects from images in the wild. In: CVPR (2020)
53. Xu, Q., Wang, W., Ceylan, D., Mech, R., Neumann, U.: Disn: Deep implicit surface network for high-quality single-view 3d reconstruction. In: NeurIPS (2019)
54. Xu, Z., Bi, S., Sunkavalli, K., Hadap, S., Su, H., Ramamoorthi, R.: Deep view synthesis from sparse photometric images. ACM TOG **38**(4) (Jul 2019)
55. Yu, A., Ye, V., Tancik, M., Kanazawa, A.: pixelNeRF: Neural radiance fields from one or few images. In: CVPR (2021)
56. Zhang, R., Isola, P., Efros, A.A., Shechtman, E., Wang, O.: The unreasonable effectiveness of deep features as a perceptual metric. In: CVPR (2018)
57. Zhao, H., Jia, J., Koltun, V.: Exploring self-attention for image recognition. In: CVPR (2020)
58. Zhou, T., Tucker, R., Flynn, J., Fyffe, G., Snavely, N.: Stereo magnification: Learning view synthesis using multiplane images. In: ACM SIGGRAPH (2018)