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] Ravi Ramamoorthi[1]

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

Abstract

Although neural radiance fields (NeRF) have shown impressive advances in novel view synthesis, most methods 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 using local image features to reconstruct a 3D object often render blurry predictions at viewpoints distant from the source view. To address this, 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 multi-layer 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 renders novel views from just a single input image, and generalizes 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/`

1. Introduction

We study the problem of novel view synthesis from a *single unposed image*. Recent works [37, 39, 57] infer the 3D shape and appearance by projecting the input image features on the queried 3D point to predict the color and density. These image-conditioned models work well for rendering target views close to the input view. However, when target views move further, it causes significant occlusion from the input view, leading to dramatic degradation of the rendering quality, as shown in Fig. 1. We hypothesize that self-occlusion causes the incorrectly-conditioned features. As illustrated in Fig. 2, when the query pixel in the target view (*e.g.*, the car's wheel) is invisible from the input view, image-conditioned models incorrectly use the features from other surface (*e.g.*, the car's window) for the target view.

To tackle this issue, we propose a novel approach that utilizes the recent advances in vision transformer (ViT) [10]

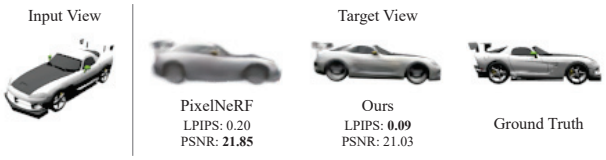


Figure 1. **Novel view synthesis in occluded regions.** The visual quality of image-conditioned model (*e.g.*, PixelNeRF [57]) degrades significantly when pixels in the target view are invisible from the input. We propose to incorporate both global features from vision transformer (ViT) and local appearance features from convolutional networks to achieve significantly better rendering quality with more details in the occluded regions. Note that LPIPS [58] (lower is better) reflects the perceptual similarity better than PSNR.

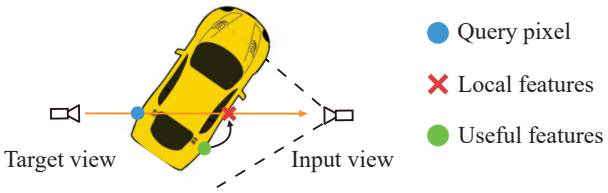


Figure 2. **The challenge of image-conditioned models in the presence of self-occlusion.** To render a car's occluded wheel (blue dot) in the target view, image-conditioned models, *e.g.*, PixelNeRF [57], query features along the ray, which corresponds to the car's window in the input view (red cross). 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.

*Work done while interning at Google.
†Work done while at Google.

and neural radiance fields (NeRF) [29] 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] including 13 object categories. Our method generalizes well across multiple categories, and works well on real-world images. Quantitative and qualitative comparisons demonstrate that our method performs favorably against existing approaches, *e.g*., SRN [44], PixelNeRF [57], FE-NVS [15], SRT [40], and FWD [4], and generates more visually appealing results. We summarize our contributions as follows:

- We introduce a NeRF-based rendering method that 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

2.1. Novel View Synthesis

Earlier works in view interpolation [6] and light fields [14, 23] establish the groundwork for image-based rendering. Later works utilize proxy geometry [3, 8] and layered representations [41, 46] to better represent the 3D scene and synthesize novel views. There has been a plethora of learning-based methods [12, 13, 19, 24, 26, 28, 43, 60] and single-input view synthesis algorithms [31, 39, 42, 54, 55, 56]. These approaches exploit the differentiable rendering pipeline to generate photorealistic results. Recently, neural radiance fields (NeRF) [29] 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 train a single scene. To address this problem, several methods [37, 48, 51, 57] utilize 2D image features to improve the generalization, or use pretrained networks with 1D latent code to represent the 3D shape, *e.g*. CodeNeRF [17]. 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. Table 1 compares the proposed method to previous approaches.

2.2. Transformer

The transformer architecture [49] has brought significant advances in natural language processing (NLP). While self-attention and its variant have achieved state-of-the-art performance in many NLP [2, 9] and vision [10, 36, 45] tasks, directly applying self-attention to an image is prohibitively expensive, as it requires each pixel to be attended to every other pixel. Several works [16, 33, 35, 59] approximate self-attention by applying it to local patches of each query pixel. Recently,the vision transformer (ViT) [10] and follow-up works [36, 52] demonstrated that applying a transformer to a sequence of patches (split from an image) achieves competitive performance on discriminative tasks (*e.g*., image classification). Wang et al. [50] 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 [18, 37, 51] 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. In our experiment, ViT encodes image features that achieves higher reconstruction quality on unseen regions than previous CNN-based approaches. SRT [40] uses a fully transformer-based framework to encode and decode 3D information. It learns the 3D scene information as a set of latent code, while our work adopts radiance field as the scene representation. SRT uses a transformer to decode the set of latent code, whereas our method uses the per-pixel information from a set of feature maps, thus having an explicit mapping between the input image and the 3D point query. Sec. 4.2 shows that our proposed method achieves favorable results over SRT in PSNR and SSIM metrics.

3. Novel View Synthesis From a Single Image

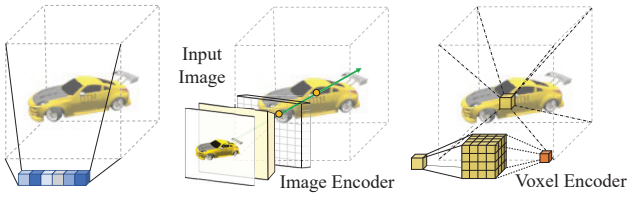
Our goal is to infer a 3D representation from a single input image for novel view synthesis. We first discuss three different paradigms to learn such a 3D representation (Sec. 3.1). Then, we propose a hybrid representation to improve rendering quality on occluded regions, where we utilize a ViT to encode 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 for novel view synthesis (Sec. 3.4).

3.1. Synthesizing Occluded Regions

In this section, we describe how previous works and our method reconstruct unseen regions illustrated in Fig. 3. Additionally, we analyze the strengths and weaknesses of each method, and propose a hybrid representation to address the

	NeRF [29]	PIFu [39]	PixelNeRF [57]	CodeNeRF [17]	NeRFormer [37]	FE-NVS [15]	SRT [40]	FWD [4]	**Ours**
Single-view input	✗	✓	✓	✗	✗	✓	✓	✓	✓
Viewer-centered coordinate	✗	✓	✓	✗	✓	✓	✓	✓	✓
Cross-category generalization	✗	✓	✓	✗	✓	✓	✓	✓	✓
Image features	✗	✓	✓	✗	✓	✓	✗	✓	✓
Global features	✗	✗	✗	✓	✗	✗	✓	✗	✓

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 Fig. 1).



(a) 1D latent code (b) 2D image features (c) 3D volume features

Figure 3. **Illustration of different representations for a 3D object.** (a) 1D latent code-based approaches [7, 11, 17, 27, 30, 32] encode the 3D object in an 1D vector. (b) 2D image-based methods [39, 57] 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 color and density vector RGBσ.

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. If a 3D point $\mathbf{x}$ 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. If $\mathbf{x}$ is occluded, we resort to information other than the color at the projection $\pi(\mathbf{x})$. There are three possible solutions to gather such information.

1D latent code. Existing methods encode 3D and appearance prior through a 1D global latent vector $\mathbf{z}$ [7, 11, 17, 27, 30, 32, 38], and decode the color $\mathbf{c}$ and density σ through CNN as the following, shown in Fig. 3(a) [17]:

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

where $\mathbf{x}$ and $\mathbf{d}$ denotes the the spatially-varying sampling position and viewing direction. Since different 3D points share the same latent code, the inductive bias is limited.

2D spatially-variant image feature. There are many interests around image-conditioned methods, such as PIFu [39] and PixelNeRF [57], due to the flexibility and high-quality results around the input views. These approaches are more computationally efficient as they operate in the 2D image space rather than 3D voxels, as illustrated in Fig. 3(b). As a representative example, PixelNeRF defines 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 $\mathbf{d}_c$ is the ray direction.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 encourages better rendering quality in visible areas, and is more computationally efficient. However, it often generates blurry predictions in unseen parts shown in Fig. 1.

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 $\mathbf{x}$. This method is faster in rendering, and leverages 3D prior to rendering unseen geometry. On the other hand, it suffers from limited rendering resolution due to the voxel size and 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 refines the shape. However, volume-based methods are more computationally-intensive and require more memory when increasing the grid size. Our method combines the advantage of 2D-based method that condition on local image features, and 1D-based methods that encode global information. Specifically, we utilize (i) a ViT architecture and its

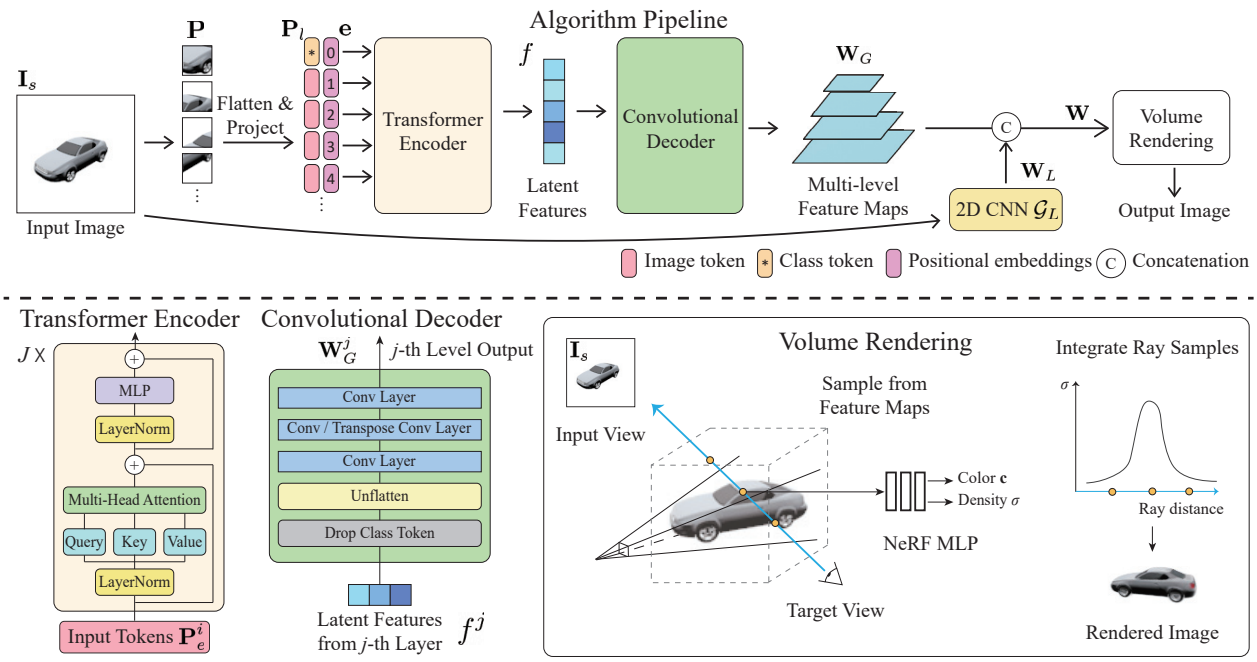


Figure 4. **Overview of our rendering pipeline.** We first divide an input image into $N = 8 \times 8$ patches $\mathbf{P}$. Each patch is flattened and linearly projected to an image token $\mathbf{P}_l$. The transformer encoder takes the image tokens and learnable positional embeddings $\mathbf{e}$ 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).

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 the long-range dependencies between features. Thus, our local and global hybrid representation allows for more flexibility and better visual quality in the unseen regions. Unlike CodeNeRF [17] and DISN [55], our method does not require a canonical coordinate system to utilize the global features. Our method enjoys the benefits of high-resolution image features from 2D-CNN, while improving the receptive fields through ViT encoder.

3.2. Global Features from Vision Transformer

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 $\mathbf{P}_l^0, \mathbf{P}_l^1, ..., \mathbf{P}_l^N$. We also add learnable positional embeddings $\mathbf{e}$ to distinguish between different spatial patches: $\mathbf{P}_e^i = \mathbf{P}_l^i + \mathbf{e}^i$.

Transformer encoder. The tokens $\{\mathbf{P}_e^0, \mathbf{P}_e^1, ..., \mathbf{P}_e^N\}$ 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 $\mathcal{O} : \mathbb{R}^{(N+1)\times D} \rightarrow \mathbb{R}^{N\times D}$. After dropping the class token, we unflatten the image by $\mathcal{U} : \mathbb{R}^{N\times D} \rightarrow \mathbb{R}^{\frac{H}{P}\times\frac{W}{P}\times D}$. 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 $\mathcal{D} : \mathbb{R}^{\frac{H}{P}\times\frac{W}{P}\times D} \rightarrow \mathbb{R}^{H^j\times W^j\times D^j}$, 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,

$$\mathbf{W}_G^j = (\mathcal{D}\circ\mathcal{U}\circ\mathcal{O})(f^j), \text{where } j \in \{0, 1, ..., J\}. \quad (4)$$

3.3. Local Features from Convolutional Networks

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. 9). 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

$$\mathbf{W}_L = \mathcal{G}_L(\mathbf{I}_s), \mathcal{G}_L : \mathbb{R}^{H\times W\times C} \rightarrow \mathbb{R}^{\frac{H}{2}\times\frac{W}{2}\times D_L}, \quad (5)$$

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:

$$\mathbf{W} = \mathcal{G}(\mathbf{W}_G^0, \mathbf{W}_G^1, ..., \mathbf{W}_G^J; \mathbf{W}_L) \quad (6)$$

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 $\mathbf{d}_c$. We encode $\mathbf{x}_c$ with positional encoding γ:

$$\begin{aligned}\gamma(p) = (&\sin(2^0\pi p), \cos(2^0\pi p), ...,\\ &\sin(2^{M-1}\pi p), \cos(2^{M-1}\pi p)),\end{aligned} \quad (7)$$

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:

$$(\sigma, \mathbf{c}) = \text{MLP}(\gamma(\mathbf{x}_c); \mathbf{d}_c; \mathbf{W}(\pi(\mathbf{x}))). \quad (8)$$

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

$$\hat{\mathbf{C}}(\mathbf{r}) = \int_{t_n}^{t_f} T(t)\sigma(t)\mathbf{c}(t)dt, \quad (9)$$

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:

$$\mathcal{L} = \sum_{\mathbf{r}} ||\hat{\mathbf{C}}(\mathbf{r}) - \mathbf{C}(\mathbf{r})||_2^2. \quad (10)$$

Implementation details. We implement our method using PyTorch [34]. The ViT module is initialized from the pretrained weights of [53] 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 10^{-4} for the MLP and 10^{-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. We train the model with 512 rays for 1 object and a batch size of 8.

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). Sec. 4.3 shows the qualitative results of our method on real input images. Sec. 4.4 provides ablation studies to analyze the key components in our method. Sec. 4.5 replaces the ViT with different backbones and show the efficacy of using ViT features. Finally, we discuss the limitations and future work (Sec. 4.6).

4.1. Category-specific View Synthesis

We evaluate our method on the same experimental setup and data as SRN [44]. 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

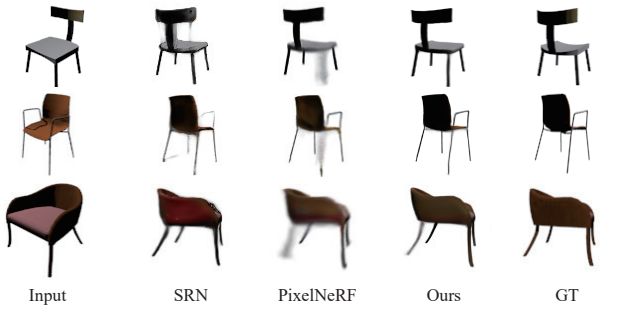


Figure 5. **Category-specific view synthesis on Chairs.** The results of SRN and PixelNeRF are often too blurry, especially on the legs that are not visible in the input views. Our method can generate novel views with clearer structures and sharper edges.

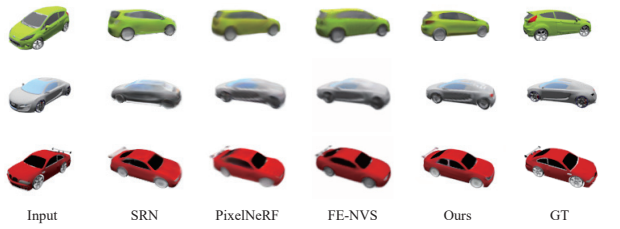


Figure 6. **Category-specific view synthesis on Cars.** Our method can generate sharper car structure and richer details, such as the rear lights and windows in the first row, the wheels and door in the second row, and the windows in the third row.

other 250 views are used as target views. The image resolution is 128×128. We compare our method with SRN [44], PixelNeRF [57][1], CodeNeRF [17][2] and FE-NVS [15][3].

As shown in Table 2, our method achieves state-of-the-art performance against existing approaches in terms of PSNR, SSIM, and LPIPS [58]. 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 [44] and PixelNeRF [57] have blurry predictions on the chair legs. On the car dataset, we obtain the best LPIPS and SSIM scores. While PixelNeRF [57] 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 [17] 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 as well. However, we try our best to obtain high-resolution screenshots from their paper and compare with their results on the same view.

[1]LPIPS is calculated from the results provided by the authors.
[2]LPIPS and code for unposed inference are not available.
[3]LPIPS is provided by the authors on request.

Methods	Chairs PSNR(↑)	Chairs SSIM(↑)	Chairs LPIPS(↓)	Cars PSNR(↑)	Cars SSIM(↑)	Cars LPIPS(↓)
SRN [44]	22.89	0.89	0.104	22.25	0.89	0.129
PixelNeRF [57]	23.72	0.91	0.128	23.17	0.90	0.146
CodeNeRF [17]	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	0.077	22.88	0.91	0.084

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).

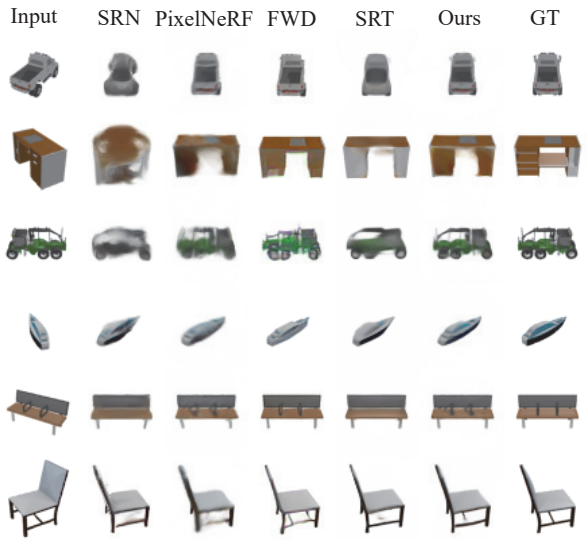


Figure 7. **Visual comparison of category-agnostic view synthesis.** The results of SRN [44], PixelNeRF [57] and SRT [40] are often too blurry and contain smearing artifacts. In contrast, our results are sharper with more fine details. FWD [4] produces distorted renderings at far viewpoints because the depth is not as accurate for occluded regions. The visual results of all 13 categories are provided in the supplementary material.

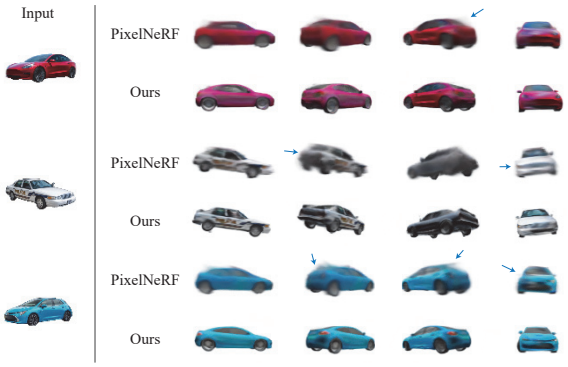


Figure 8. **Results on real input images.** Our method is able to generate visually-pleasing results even trained on a synthetic dataset. Conversely, PixelNeRF fails to keep the finer details. Note the side mirrors and headlamps of the bottom right inset.

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
	FWD	30.01	26.16	28.49	27.01	23.44	24.00	27.84	24.45	30.40	26.76	25.91	27.61	28.69	26.66
	SRT	31.47	28.45	30.40	28.21	24.69	24.58	28.56	25.61	30.09	28.11	27.42	28.28	29.18	27.87
	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
	FWD	0.952	0.914	0.918	0.939	0.857	0.867	0.906	0.857	0.968	0.909	0.906	0.924	0.936	0.911
	SRT	0.954	0.925	0.920	0.937	0.861	0.855	0.904	0.854	0.962	0.911	0.909	0.918	0.930	0.912
	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
	FWD	0.034	0.055	0.056	0.042	0.081	0.079	0.062	0.091	0.026	0.054	0.049	0.056	0.052	0.055
	SRT	0.050	0.068	0.058	0.062	0.085	0.087	0.082	0.096	0.045	0.066	0.055	0.059	0.079	0.066
	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 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.

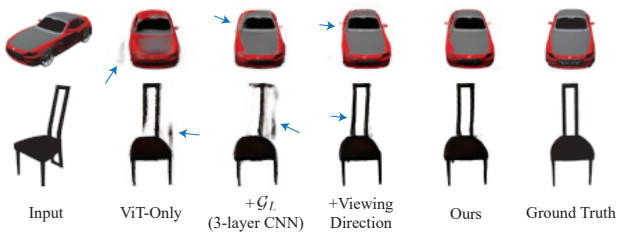


Figure 9. **Effects of different components.** The ViT-only model can render realistic images, but the local appearance and color may not look similar to the input view. By extracting local features with a 3-layer CNN, the rendered car shows more faithful colors to the input. With the viewing direction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

Method	Cars PSNR↑	Cars SSIM↑	Cars LPIPS↓	Chairs PSNR↑	Chairs SSIM↑	Chairs LPIPS↓
PixelNeRF	23.17	0.90	0.146	23.72	0.91	0.128
ViT only	21.95	0.89	0.130	23.45	0.92	0.099
+ $\mathcal{G}_L$ (3-layer CNN)	22.42	0.90	0.113	23.42	0.92	0.085
+ Viewing Direction	22.70	0.91	0.088	24.53	0.93	0.094
Ours	22.88	0.91	0.084	24.48	0.93	0.077

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 Fig. 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 Fig. 9). 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 Fig. 9.

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 [20] 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. Our method achieves the state-of-the-art performance against SRN [44], PixelNeRF [57], FE-NVS [15], FWD [4] and SRT [40] on all 13 categories in PSNR and SSIM. Our method achieves competitive performance in LPIPS compared to recent state-of-the-art approaches, FWD [4] and SRT [40]. 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 shows that our method reconstructs finer object structure and details. Even though FWD [4] achieves better LPIPS scores, their results show distorted renderings at larger displacements, possibly due to erroneous depth estimation at unseen areas. In Fig. 7, the vehicle on the third row shows severe artifacts when FWD tries to render viewpoints at the opposite side of the input. Please refer to supplementary materials for more examples. Since SRT [40] converts input images to a set of latent codes without a one-to-one mapping to the source image, their results often lose fine details, *e.g.*, the bench on

Method	Cars			Chairs		
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
PixelNeRF	23.17	0.90	0.146	23.72	0.91	0.128
Replace ViT with EfficientNet	23.28	0.91	0.106	24.09	0.92	0.105
Replace ViT with ConvNeXt	23.30	0.91	0.092	24.37	0.93	0.089
Ours	22.88	0.91	0.084	24.48	0.93	0.077

Table 5. **Comparison with different backbone choices.** We replace the vision transformer with EfficientNet [47] and ConvNeXt [25] to observe potential performance impact. Our method achieves favorable overall performance in LPIPS compared to other backbones.

the second to the last row in Fig. 7.

4.3. View Synthesis on Real Images

Our method generalizes to real images. We use our model trained on the ShapeNet car dataset to test on real car images from the Stanford cars dataset [22]. We use an image segmentation model [21] 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 start from the baseline method using only the ViT to extract global features. While ViT encodes the high-level global information, it fails to preserve the color and appearance from the input view due to the low-resolution latent embeddings, as shown in Fig. 9. The rendered results show inconsistent appearances to the input view on non-occluded regions, as shown in the second column in . 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. 9). However, we can see that the chair's back is still blurry (bottom of the third column in Fig. 9). 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. 9) and reveals more details such as the rear mirror of the car (top of the 4-th column in Fig. 9). Our final model adopts 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. 9). Table 4 also reports the quantitative results of these design decisions on both datasets.

4.5. Global Features from Different Backbones

To further verify that ViT outperforms convolutional backbones for image-conditioned NeRFs, we benchmark our method against two baselines that replace the proposed ViT backbone with EfficientNet [47] and ConvNeXt [25], i.e., modern CNN models with better performance than ResNet34 and comparable numbers of parameters to ViT. The results are presented in Table 5 which shows that our method achieves better LPIPS compared to these baselines on both the car and chair categories. This ablation study demonstrates that using ViT as the backbone achieves better performance for image-conditioned NeRFs due to the model architecture design instead of more parameters.

4.6. Limitations and Future Work

First, our method does not utilize geometry priors such as symmetry [54]. For example, in the car dataset, some details 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 components. Lastly, generative 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 ViT 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 lead to more exciting advances in the frontier for immersive 3D content.

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, Google, Amazon, a Sony Research Award, the Ronald L. Graham Chair, and the UC San Diego Center for Visual Computing.

References

[1] Relja Arandjelović and Andrew Zisserman. NeRF in detail: Learning to sample for view synthesis. *arXiv:2106.05264*, 2021.

[2] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, and Amanda Askell. Language models are few-shot learners. In *NeurIPS*, 2020.

[3] Chris Buehler, Michael Bosse, Leonard McMillan, Steven Gortler, and Michael Cohen. Unstructured lumigraph rendering. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, 2001.

[4] Ang Cao, Chris Rockwell, and Justin Johnson. Fwd: Real-time novel view synthesis with forward warping and depth. *CVPR*, 2022.

[5] Angel X. Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. ShapeNet: An Information-Rich 3D Model Repository. *arXiv:1512.03012*, 2015.

[6] Shenchang Eric Chen and Lance Williams. View interpolation for image synthesis. In *Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques*, 1993.

[7] Zhiqin Chen and Hao Zhang. Learning implicit fields for generative shape modeling. 2019.

[8] Paul E Debevec, Camillo J Taylor, and Jitendra Malik. 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] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv:1810.04805*, 2018.

[10] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2021.

[11] Emilien Dupont, Bautista Miguel Angel, Alex Colburn, Aditya Sankar, Carlos Guestrin, Josh Susskind, and Qi Shan. Equivariant neural rendering. In *ICML*, 2020.

[12] John Flynn, Michael Broxton, Paul Debevec, Matthew DuVall, Graham Fyffe, Ryan Overbeck, Noah Snavely, and Richard Tucker. DeepView: View synthesis with learned gradient descent. In *CVPR*, 2019.

[13] John Flynn, Ivan Neulander, James Philbin, and Noah Snavely. DeepStereo: Learning to predict new views from the world's imagery. In *ICCV*, 2016.

[14] Steven J Gortler, Radek Grzeszczuk, Richard Szeliski, and Michael F Cohen. The lumigraph. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, 1996.

[15] Pengsheng Guo, Miguel Angel Bautista, Alex Colburn, Liang Yang, Daniel Ulbricht, Joshua M. Susskind, and Qi Shan. Fast and explicit neural view synthesis. In *WACV*, 2022.

[16] Han Hu, Zheng Zhang, Zhenda Xie, and Stephen Lin. Local relation networks for image recognition. In *ICCV*, 2019.

[17] Wonbong Jang and Lourdes Agapito. CodeNeRF: Disentangled neural radiance fields for object categories. In *ICCV*, 2021.

[18] Mohammad Mahdi Johari, Yann Lepoittevin, and François Fleuret. GeoNeRF: Generalizing nerf with geometry priors. *arXiv:2111.13539*, 2021.

[19] Nima Khademi Kalantari, Ting-Chun Wang, and Ravi Ramamoorthi. Learning-based view synthesis for light field cameras. *ACM TOG*, 35(6):1–10, 2016.

[20] Hiroharu Kato, Yoshitaka Ushiku, and Tatsuya Harada. Neural 3D mesh renderer. In *CVPR*, 2018.

[21] Alexander Kirillov, Yuxin Wu, Kaiming He, and Ross Girshick. PointRend: Image segmentation as rendering. In *CVPR*, 2020.

[22] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3D object representations for fine-grained categorization. In *4th International IEEE Workshop on 3D Representation and Recognition (3dRR-13)*, Sydney, Australia, 2013.

[23] Marc Levoy and Pat Hanrahan. Light field rendering. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, 1996.

[24] Kai-En Lin, Lei Xiao, Feng Liu, Guowei Yang, and Ravi Ramamoorthi. Deep 3D mask volume for view synthesis of dynamic scenes. In *ICCV*, 2021.

[25] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[26] Stephen Lombardi, Tomas Simon, Jason Saragih, Gabriel Schwartz, Andreas Lehrmann, and Yaser Sheikh. Neural volumes: Learning dynamic renderable volumes from images. *ACM TOG*, 38(4), 2019.

[27] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3D reconstruction in function space. In *CVPR*, 2019.

[28] Ben Mildenhall, Pratul P. Srinivasan, Rodrigo Ortiz-Cayon, Nima Khademi Kalantari, Ravi Ramamoorthi, Ren Ng, and Abhishek Kar. Local light field fusion: Practical view synthesis with prescriptive sampling guidelines. *ACM TOG*, 2019.

[29] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. NeRF: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020.

[30] Michael Niemeyer, Lars Mescheder, Michael Oechsle, and Andreas Geiger. Differentiable volumetric rendering: Learning implicit 3D representations without 3D supervision. In *CVPR*, 2020.

[31] Simon Niklaus, Long Mai, Jimei Yang, and Feng Liu. 3D ken burns effect from a single image. *ACM TOG*, 38(6):1–15, 2019.

[32] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *CVPR*, 2019.

[33] Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Lukasz Kaiser, Noam Shazeer, Alexander Ku, and Dustin Tran. Image transformer. In *ICML*, 2018.

[34] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *NeurIPS*. 2019.

[35] Prajit Ramachandran, Niki Parmar, Ashish Vaswani, Irwan Bello, Anselm Levskaya, and Jon Shlens. Stand-alone self-attention in vision models. In *NeurIPS*, 2019.

[36] René Ranftl, Alexey Bochkovskiy, and Vladlen Koltun. Vision transformers for dense prediction. In *ICCV*, 2021.

[37] Jeremy Reizenstein, Roman Shapovalov, Philipp Henzler, Luca Sbordone, Patrick Labatut, and David Novotny. Common objects in 3D: Large-scale learning and evaluation of real-life 3D category reconstruction. In *ICCV*, 2021.

[38] Konstantinos Rematas, Ricardo Martin-Brualla, and Vittorio Ferrari. Sharf: Shape-conditioned radiance fields from a single view. *arXiv:2102.08860*, 2021.

[39] Shunsuke Saito, , Zeng Huang, Ryota Natsume, Shigeo Morishima, Angjoo Kanazawa, and Hao Li. Pifu: Pixel-aligned implicit function for high-resolution clothed human digitization. *arXiv:1905.05172*, 2019.

[40] Mehdi S. M. Sajjadi, Henning Meyer, Etienne Pot, Urs Bergmann, Klaus Greff, Noha Radwan, Suhani Vora, Mario Lucic, Daniel Duckworth, Alexey Dosovitskiy, Jakob Uszkoreit, Thomas Funkhouser, and Andrea Tagliasacchi. Scene Representation Transformer: Geometry-Free Novel View Synthesis Through Set-Latent Scene Representations. *CVPR*, 2022.

[41] Jonathan Shade, Steven Gortler, Li-wei He, and Richard Szeliski. Layered depth images. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, 1998.

[42] Meng-Li Shih, Shih-Yang Su, Johannes Kopf, and Jia-Bin Huang. 3D photography using context-aware layered depth inpainting. In *CVPR*, 2020.

[43] Vincent Sitzmann, Justus Thies, Felix Heide, Matthias Nießner, Gordon Wetzstein, and Michael Zollhofer. DeepVoxels: Learning persistent 3D feature embeddings. In *CVPR*, 2019.

[44] Vincent Sitzmann, Michael Zollhöfer, and Gordon Wetzstein. Scene representation networks: Continuous 3D-structure-aware neural scene representations. In *NeurIPS*, 2019.

[45] Robin Strudel, Ricardo Garcia, Ivan Laptev, and Cordelia Schmid. Segmenter: Transformer for semantic segmentation. In *ICCV*, 2021.

[46] Rick Szeliski and Polina Golland. Stereo matching with transparency and matting. volume 32, pages 45–61, July 1999.

[47] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*, pages 6105–6114. PMLR, 2019.

[48] Alex Trevithick and Bo Yang. GRF: Learning a general radiance field for 3D scene representation and rendering. In *arXiv:2010.04595*, 2020.

[49] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *NeurIPS*, 2017.

[50] Dan Wang, Xinrui Cui, Xun Chen, Zhengxia Zou, Tianyang Shi, Septimiu Salcudean, Z Jane Wang, and Rabab Ward. Multi-view 3D reconstruction with transformers. In *ICCV*, 2021.

[51] Qianqian Wang, Zhicheng Wang, Kyle Genova, Pratul Srinivasan, Howard Zhou, Jonathan T. Barron, Ricardo Martin-Brualla, Noah Snavely, and Thomas Funkhouser. IBRNet: Learning multi-view image-based rendering. In *CVPR*, 2021.

[52] Wenhai Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, Tong Lu, Ping Luo, and Ling Shao. Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. In *ICCV*, 2021.

[53] Ross Wightman. Pytorch image models. `https://github.com/rwightman/pytorch-image-models`, 2019.

[54] Shangzhe Wu, Christian Rupprecht, and Andrea Vedaldi. Unsupervised learning of probably symmetric deformable 3D objects from images in the wild. In *CVPR*, 2020.

[55] Qiangeng Xu, Weiyue Wang, Duygu Ceylan, Radomir Mech, and Ulrich Neumann. Disn: Deep implicit surface network for high-quality single-view 3d reconstruction. In *NeurIPS*, 2019.

[56] Zexiang Xu, Sai Bi, Kalyan Sunkavalli, Sunil Hadap, Hao Su, and Ravi Ramamoorthi. Deep view synthesis from sparse photometric images. *ACM TOG*, 38(4), July 2019.

[57] Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. pixelNeRF: Neural radiance fields from one or few images. In *CVPR*, 2021.

[58] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018.

[59] Hengshuang Zhao, Jiaya Jia, and Vladlen Koltun. Exploring self-attention for image recognition. In *CVPR*, 2020.

[60] Tinghui Zhou, Richard Tucker, John Flynn, Graham Fyffe, and Noah Snavely. Stereo magnification: Learning view synthesis using multiplane images. In *ACM SIGGRAPH*, 2018.