Scene Representation Transformer: Geometry-Free Novel View Synthesis Through Set-Latent Scene Representations

Mehdi S. M. Sajjadi Henning Meyer Etienne Pot Urs Bergmann Klaus Greff
Noha Radwan Suhani Vora Mario Lučić Daniel Duckworth Alexey Dosovitskiy*
Jakob Uszkoreit* Thomas Funkhouser Andrea Tagliasacchi‡*

Google Research ‡ Simon Fraser University

Abstract

A classical problem in computer vision is to infer a 3D scene representation from few images that can be used to render novel views at interactive rates. Previous work focuses on reconstructing pre-defined 3D representations, e.g. *textured meshes, or implicit representations,* e.g. *radiance fields, and often requires input images with precise camera poses and long processing times for each novel scene.*

In this work, we propose the Scene Representation Transformer (SRT), a method which processes posed or unposed RGB images of a new area, infers a "set-latent scene representation", and synthesises novel views, all in a single feed-forward pass. To calculate the scene representation, we propose a generalization of the Vision Transformer to sets of images, enabling global information integration, and hence 3D reasoning. An efficient decoder transformer parameterizes the light field by attending into the scene representation to render novel views. Learning is supervised end-to-end by minimizing a novel-view reconstruction error.

We show that this method outperforms recent baselines in terms of PSNR and speed on synthetic datasets, including a new dataset created for the paper. Further, we demonstrate that SRT scales to support interactive visualization and semantic segmentation of real-world outdoor environments using Street View imagery.

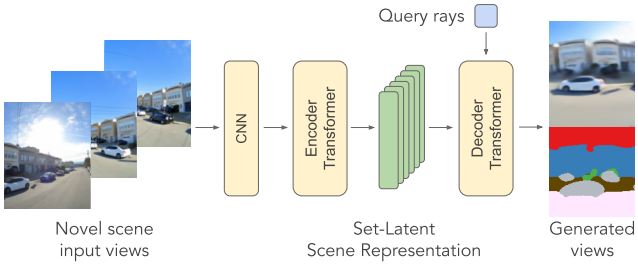

Figure 1. **Model overview** – SRT encodes a collection of images into the scene representation: a *set* of latent features. Novel views are rendered in real-time by attending into the latent representation with light field rays, see Fig. 2 for details.

1. Introduction

The goal of our work is interactive novel view synthesis: given *few* RGB images of a previously unseen scene, we synthesize novel views of the same scene at interactive rates and without expensive per-scene processing. Such a system can be useful for virtual exploration of urban spaces [26], as well as other mapping, visualization, and AR/VR applications [27]. The main challenge here is to infer a scene representation that encodes enough 3D information to render novel views with correct parallax and occlusions.

Traditional methods build explicit 3D representations, such as colored point clouds [1], meshes [31], voxels [36], octrees [42], and multi-plane images [52]. These representations enable interactive rendering, but they usually require expensive and fragile reconstruction processes.

More recent work has investigated representing scenes with purely implicit representations [39]. Notably, NeRF represents the scene as a 3D volume parameterized by an MLP [25], and has been demonstrated to scale to challenging real-world settings [23]. However, it typically requires hundreds of MLP evaluations for the volumetric rendering of each ray, relies on accurate camera poses, and requires an expensive training procedure to onboard new scenes, as no parameters are shared across different scenes.

*Work done while at Google.
Contributions: *MS*: original idea & conceptualization, main implementation, experiments, writing, organization, lead; *HM*: idea conceptualization, implementation, experiments, writing; *EP*: implementation, experiments, writing; *UB*: idea conceptualization, implementation, experiments, writing; *KG*: dataset support, advising; *NR*: project-independent code; *SV*: project-independent code; *ML*: advising, writing; *DD*: advising, project-independent code; *AD*: advising; *JU*: advising; *TF*: datasets, advising, writing; *AT*: advising, writing.
Correspondence: srt@msajjadi.com & tutmann@google.com
Project website: srt-paper.github.io

Follow-ups that address these shortcomings include re-projection based methods which still rely on accurate camera poses due to their explicit use of geometry, and latent models which are usually geometry-free and able to reason globally, an advantage for the sparse input view setting. However, those methods generally fail to scale to complex real-world datasets. As shown in Tab. 1, interactive novel view synthesis on complex real-world data remains a challenge.

We tackle this challenge by employing an encoder-decoder model built on transformers, learning a scalable implicit representation, and replacing explicit geometric operations with *learned* attention mechanisms. Different from a number of prior works that learn the latent scene representations through an auto-decoder optimization procedure [38, 16], we specifically rely on an encoder architecture to allow for *instant* inference on novel scenes. At the same time, we do not rely on explicit locally-conditioned geometry [49, 41] to instead allow the model to reason globally. This not only comes with the advantage of stronger generalization abilities (*e.g.*, for rendering novel cameras further away from the input views, see Fig. 4), but also enables the model to be more efficient, as global information is processed once per scene (*e.g.*, to reconstruct 3D geometry and to resolve occlusions) in our model, rather than once or hundreds of times per rendered pixel, as in previous methods [49, 41, 44].

We evaluate the proposed model on several datasets of increasing complexity against relevant prior art, and see that it strikes a unique balance between scalability to complex scenes (Sec. 4), robustness to noisy camera poses (or no poses at all, Fig. 6), and efficiency in interactive applications (Tab. 3). Further, we show a proof-of-concept for the downstream task of semantic segmentation using the learned representation on a challenging real-world dataset (Fig. 7).

2. Related Work

There is a long history of prior work on novel view synthesis from multiple RGB images based on neural representations (see Tab. 1 and [40]). Recent work has demonstrated that novel views can be synthesized using deep networks trained to compute radiance values for a given 3D position and direction. In particular, NeRF [25] trains an MLP to store radiance and density used in a volume rendering system and produces novel views with remarkably fine details. Several extensions have been proposed to mitigate some its shortcomings, including ones to accelerate rendering [12, 30, 48], handle color variations [23], optimize camera poses [21], and perform well with few images [15]. However, these methods generally still require an expensive optimization of an MLP for *each* novel scene, and thus are impractical for deployment at large scale.

NeRF with Re-projection. Several NeRF extensions have been proposed that do not always require to be trained individually for each novel scene. These methods rely on explicit 3D-to-2D projections of samples along the rays to gather features extracted from CNNs, which are then aggregated by a neural network for each point [41, 44, 49], or into a voxel grid [5]. As a result, these networks can be pre-trained, and then applied to novel scenes either without any optimization, or with additional per-scene fine-tuning. However, reliance on explicit camera projections requires precise camera poses, and direct projections cannot take occlusions of objects between samples and input features into account.

	NeRF *et al.*					Re-projection				Latent			Other		Ours
	NeRF [25]	FastNeRF [9]	NeRF-W [23]	BARF [21]	DietNeRF [15]	GRF [41]	IBRNet [44]	PixelNeRF [49]	MVSNeRF[5]	CodeNeRF [16]	NeRF-VAE [20]	LFN [38]	NRW [24]	GFVS [32]	SRT
1) Real-world data	✓	✓	✓	✓	✓	✓	✓	✓	✓	×	×	×	✓	✓	✓
2) Temp. consistent	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	×	×	✓
3) Real-time	×	✓	×	×	×	×	×	×	×	×	×	✓	×	×	✓
4) Appearance enc.	×	×	✓	×	×	×	×	×	×	×	×	×	×	×	✓
5) Pose-free	×	×	×	✓	×	×	×	×	×	✓	×	×	×	×	✓
6) Few images	×	×	×	×	✓	×	×	✓	✓	✓	✓	✓	✓	✓	✓
7) Generalization	×	×	×	×	×	✓	✓	✓	✓	✓	✓	✓	×	✓	✓
8) Instant new scene	×	×	×	×	×	✓	✓	✓	✓	×	✓	×	×	✓	✓
9) Global latent	×	×	×	×	×	×	×	×	×	✓	✓	✓	×	×	✓

Table 1. **Comparison of features with prior methods.** The rows indicate whether each method: 1) has been demonstrated on real world data, 2) is designed to provide coherence between nearby views, 3) performs inference in real-time, 4) handles images with varying appearance in the same scene, 5) works without known camera poses at test time, 6) works with very sparse input images, 7) generalizes from a training set to novel scenes, 8) captures novel scenes in a feed-forward step, and 9) learns a global latent scene representation.

Light Fields. Other methods learn to represent a scene with a latent vector from which novel views can be synthesized [7]. More recently, LFN [38] use an auto-decoder to infer a latent code from input images, which then conditions an MLP for computing radiance values for rays of novel views. This method is several times faster than volumetric rendering due to the fact that the network is only queried *once* per pixel, rather than hundreds of times for volumetric rendering. However, it does not scale beyond simple controlled settings, requires an expensive optimization to produce a latent code for a novel scene, and therefore also relies upon accurate input camera poses. In contrast, we utilize an encoder to infer the scene representation, which executes nearly instantly in comparison (see Tab. 3), and can leverage, but does not require, camera poses (see Sec. 4.4).

Transformers. Originally proposed for seq-to-seq tasks in natural language processing, transformers [43] drop the recurrent memory of prior methods, and instead use a multi-head attention mechanism to gather information from the most relevant portions of the input. The vision community has deeply researched applications thereof in image

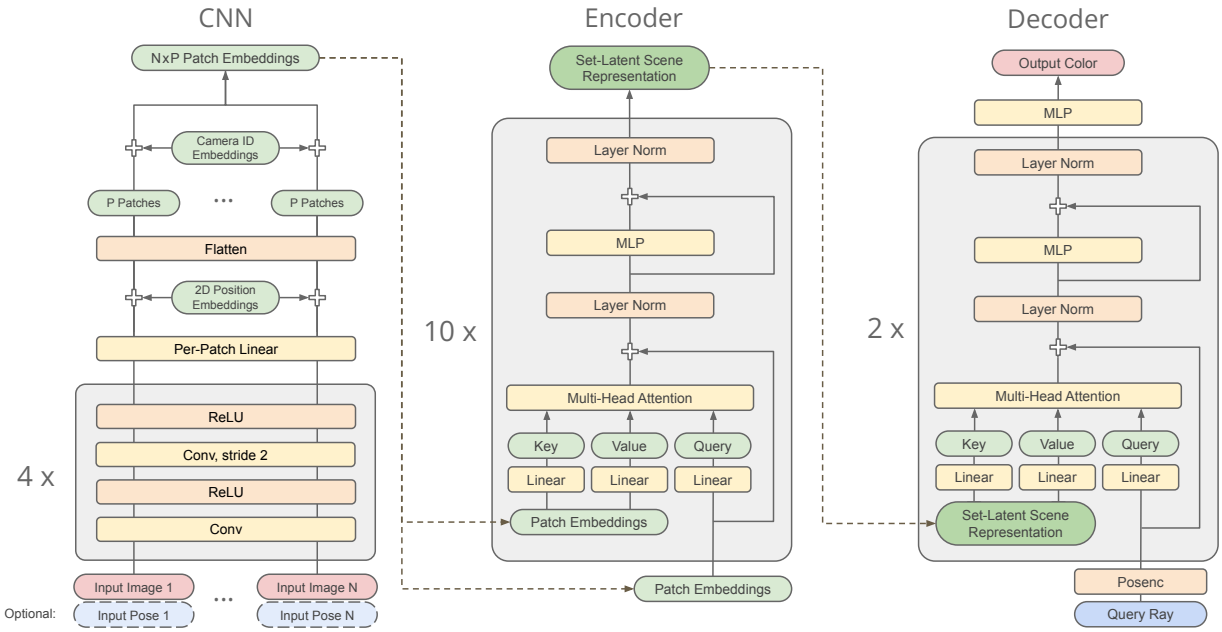

Figure 2. **Network architecture** – Given a set of posed (SRT) or unposed (UpSRT) RGB images, a CNN extracts patch features, which are processed by an encoder transformer, leading to the *set-latent scene representation*. Novel views are then rendered by attending into the scene representation for a given ray pose, from which an image can then be rendered. Details in Sec. 3 and Appendix A.2.

and video processing, where they have meanwhile achieved state of the art in several tasks [18], including inherently geometric applications such as depth estimation [29], point cloud processing [51], and generative geometry-free view synthesis [32]. The latter work uses transformers to learn a sequence of VQ-GAN embeddings [8] to sample realistic views given a single image. While results are of high-quality, this method is slow due to its auto-regressive nature, and cannot be used for video rendering, as each frame is sampled independently, creating strong flickering artifacts. ViT [6] has been proposed for very large-scale classification tasks: local features are extracted from patches of the input image, and fed to a transformer network. We will demonstrate that this architecture is well-suited for learning 3D scene representations by building on top of this idea, extending it to ingest patches from multiple images, and generalizing it to 3D by adding a transformer decoder queried with rays.

3. Method

Our model receives as input an unordered *collection* of optionally posed images $\{\mathbf{I}_i \in \mathbb{R}^{H\times W\times 3}, \mathbf{c}_i \in \mathrm{SE}(3), K_i \in \mathbb{R}^{3\times 3}\}$ of the *same* scene, with extrinsic camera matrix $\mathbf{c}_i = [R_i|t_i]$ and intrinsic camera matrix K_i. The model encodes this collection into a *set* representation of cardinality Z via an encoder $\mathcal{E}$ with trainable parameters $\theta_\mathcal{E}$:

$$\{\mathbf{z}_z \in \mathbb{R}^d\} = \mathcal{E}_{\theta_\mathcal{E}}(\,\mathrm{CNN}_{\theta_{\mathrm{CNN}}}(\{\mathbf{I}_i, \mathbf{c}_i, K_i\})) \qquad (1)$$

In more details, our encoder first processes the images via a shared CNN backbone (Fig. 2, left), and then aggregates features into a set of flat patch embeddings. This set of embeddings is then processed by the encoder transformer that generates the codes $\{\mathbf{z}_z\}$ as output. Note that $\{\mathbf{z}_z\}$ fully encodes a specific 3D scene as observed by the corresponding set of images, hence it is our *set-latent scene representation*.

This representation is queried by the decoder transformer $\mathcal{D}$ with trainable parameters $\theta_\mathcal{D}$ to generate the pixel color for a view ray $\mathbf{r} = (\mathbf{o}, \mathbf{d})$ with origin $\mathbf{o}$ and direction $\mathbf{d}$:

$$C(\mathbf{r}) = \mathcal{D}_{\theta_\mathcal{D}}(\mathbf{r} \mid \{\mathbf{z}_z\}) \qquad (2)$$

By convention, we assume $\mathbf{I}_0$ to be the *canonical* image, hence queries $\mathbf{r}$ are expressed in the coordinates of $\mathbf{c}_0$, and, whenever camera information is not available, we assume the canonical camera pose to be identity. If pose information is known, images within a scene are randomly shuffled, hence the reference frame is randomly chosen both at train and test time. Given a collection of images $\{\mathbf{I}_{s,i}\}$ from different scenes indexed by s, we train all parameters end-to-end via a reconstruction loss on novel-view synthesis:

$$\underset{\theta=[\theta_{\mathrm{CNN}},\theta_\mathcal{E},\theta_\mathcal{D}]}{\arg\min} \sum_s \mathbb{E}_{\mathbf{r}\sim\mathbf{I}^{\mathrm{gt}}_{s,i}} \|C(\mathbf{r}) - \mathbf{I}^{\mathrm{gt}}_{s,i}(\mathbf{r})\|_2^2 \qquad (3)$$

As both inputs and outputs of the model during training are 2D images, there is no explicit 3D supervision. We now describe the internals of our architecture in details, as illustrated in Fig. 2.

Pose information. We start by concatenating along the channel dimension the positional encoding γ [43, 25] of camera origin with $L_\mathbf{o}$ octaves and camera direction with

	NMR			Multi-ShapeNet		
	↑PSNR	↑SSIM	↓LPIPS	↑PSNR	↑SSIM	↓LPIPS
LFN [38]	24.95	0.870	-	14.77	0.328	0.582
PN [49]	26.80	0.910	0.108	21.97	0.689	**0.332**
SRT (Ours)	**27.87**	**0.912**	**0.066**	**23.41**	**0.697**	0.369

Table 2. **Quantitative results** – Evaluation of new scene novel-view synthesis on simple and challenging datasets. NMR baselines provided by the authors (LPIPS for LFN not available). SRT is best across datasets and metrics, with the exception of LPIPS on MSN.

$L_{\mathbf{d}}$ octaves, as seen from the currently selected canonical camera $\mathbf{c}_0$ to produce:

$$\gamma(\mathbf{r}) = \gamma(\mathbf{c}_0^{-1} \cdot \mathbf{o}, L_{\mathbf{o}}) \oplus \gamma(\mathbf{c}_0^{-1} \cdot \mathbf{d}, L_{\mathbf{d}}) \quad (4)$$

$$\mathbf{F}_i^{\gamma}(\mathbf{r}) = \mathbf{I}_i(\mathbf{r}) \oplus \gamma(\mathbf{r}) \quad (5)$$

Overall, the output of this stage will have dimensionality $|\mathbf{F}_i^{\gamma}| = (H,\, W,\, C)$ where $C{=}(3 + 6L_o + 6L_d)$ or $C{=}3$, depending on whether camera pose information is provided or not. Following [25], we perform sine and cosine encoding independently for each of the 3D dimensions (*i.e.*, $6L$).

CNN – Fig. 2 (left). A shared convolutional neural network extracts patch features while scaling the spatial dimensions of the input images by a factor of $K{=}16$, hence mapping the shape from $(H,\, W,\, C)$ to $(h = {}^{H}/_{K},\, w = {}^{W}/_{K},\, \mathrm{C})$:

$$\mathbf{F}_i^{\mathrm{CNN}} = \mathrm{CNN}_{\theta_{\mathrm{CNN}}}(\mathbf{F}_i^{\gamma}) + \mathcal{B}(\theta) \quad (6)$$

We add a single globally learned 2D position embedding $\mathcal{B}(\theta) \in \mathbb{R}^{h \times w \times \mathrm{C}}$ to all patches of all images, to allow the model to retain the position of each patch in the corresponding image. The spatial dimensions are then flattened to obtain $\hat{\mathbf{F}}_i^{\mathrm{CNN}} \in \mathbb{R}^{hw, \mathrm{C}}$. Further, one of two learned camera id embeddings $\mathcal{B}_{i=0}(\theta),\, \mathcal{B}_{i\neq 0}(\theta) \in \mathbb{R}^{\mathrm{C}}$ are added to all patches of the canonical camera, and all remaining cameras, respectively. This is to allow the model to distinguish the reference camera from all others:

$$\mathbf{F}_{i,n} = \hat{\mathbf{F}}_{i,n}^{\mathrm{CNN}} + \begin{cases} \mathcal{B}_{i=0}(\theta), & \text{if } i = 0, \\ \mathcal{B}_{i\neq 0}(\theta), & \text{if } i \neq 0, \end{cases} \quad \forall n \in \{1, .., hw\} \quad (7)$$

All features together form (unsorted) sets of *patch embeddings* of cardinality $|\{\mathbf{f}_f\}| = Ihw$, each of size $\mathbb{R}^C$:

$$\{\mathbf{f}_f\} = \bigcup_{i=1}^{I} \bigcup_{n=1}^{hw} \mathbf{F}_{i,n} \quad (8)$$

Encoder Transformer – Fig. 2 (center). We pass the patch embeddings through a standard transformer [43] that alternates between attention to all tokens, and small MLP blocks:

$$\{\mathbf{z}_z\} = \mathcal{E}_{\theta_{\mathcal{E}}}(\{\mathbf{f}_f\}) \quad (9)$$

This network integrates information of the scene at a global scale by attending across patches and input images to infer a 3D scene representation. Note that crucially, the set-latent scene representation scales in size with the amount of input information rather than being fixed-size [16, 20, 38].

	LFN [38]	PN [49]	SRT
Model size (# parameters)	105 M	28 M	74 M
a) Scene encoding time	∼ 100 s	**0.005 s**	0.010 s
b) Image rendering speed	**192 fps**	1.3 fps	121 fps
c) New-scene video rendering	∼ 100 s	75.5 s	**0.182 s**

Table 3. **Computational performance** – Model size, and a) time required to encode a scene, b) frame rate for individually rendering 100 frames after encoding, c) application: total time to encode a novel scene and render a video of 100 frames. SRT is orders of magnitudes faster for real-world use. See Appendix A.2 for details.

Decoder Transformer – Fig. 2 (right). The decoder $\mathcal{D}$ is also a transformer. The main difference with respect to the encoder is that the ray corresponding to the pixel to be rendered is used as the *query* for the multi-head attention mechanism, while key and value are computed from the scene representation $\{\mathbf{z}_z\}$ in all layers. In other words, the decoder learns to *attend* to the most relevant subset of features in the scene representation to calculate the output color. Note that the query ray position is encoded as in Eq. (4). The output of the decoder is finally passed to a small 2-layer MLP to compute the pixel color.

3.1. Training and Inference

During training, all components are simply trained end-to-end using an L2 reconstruction loss on novel views, see Eq. (3). We provide further training details in the Appendix A.2. For inference, input images are encoded into a scene representation once, which can then be used to render an arbitrary number of novel views (*e.g.*, for a video). Note that this methodology is designed to maximize inference efficiency, as the *3D reasoning* is executed only *once* in the encoder transformer, independent of the number of novel views to be rendered. For downstream tasks, the encoder for the scene representation can be pre-trained and either frozen, or fine-tuned along with a new decoder for a novel task (e.g. semantic segmentation - Sec. 4.5).

4. Experimental Results

We run a series of experiments to evaluate how well SRT performs novel view synthesis in comparison to previous works and to study how each component of the network architecture contributes to the results. In all experiments, all methods are pre-trained on a dataset of images from one set of scenes and then tested on images from novel scenes. Unless stated otherwise, we train SRT with the same model architecture, hyper parameters, and training protocols for all experiments across all datasets. Synthesized images are evaluated with PSNR, SSIM [46], and LPIPS [50] (VGG).

4.1. Datasets

Neural 3D Mesh Renderer Dataset (NMR) [17]. NMR has been used in several previous studies. It consists of ShapeNet [4] objects rendered at 24 fixed views. The dataset is very simple, and likely does not provide good evidence for the applicability in real-world settings.

MultiShapeNet (MSN). We therefore propose this significantly more challenging dataset rendered using photorealistic ray tracing [11]. Each scene has 16-31 ShapeNet objects dropped at random locations within an invisible bounding box. We further sample from 382 complex HDR backgrounds and environment maps. Viewpoints are selected by uniformly sampling within a half-sphere shell.

This dataset is significantly more difficult than NMR for several reasons: 1) images are rendered photorealistically, 2) scenes contain many different types of objects in complex, tightly-packed configurations, 3) the nontrivial background maps hamper the model's abilities to segment the objects (just like in the real world), and 4) the randomly sampled viewpoints prevent models from overfitting, unlike NMR where the same cameras and canonically oriented objects are used. The dataset is available at srt-paper.github.io.

Street View. This is a new dataset created from real-world Street View data [10]. The training dataset contains 5.5 M scenes from San Francisco, each with 10 viewpoints. This dataset is very challenging, since the input viewpoints are typically several meters from the reference viewpoints, the cameras are often arranged in straight lines (along the car trajectory, meaning there is not a lot of diversity in viewing angles), the scenes contain challenging geometry (*e.g.*, trees and thin poles) and dynamic elements (*e.g.*, moving vehicles and pedestrians), the cameras have fisheye distortion and rolling shutter (making reprojection-based methods such as [49, 41, 44] inapplicable), and finally, different images often contain different exposure and white-balance settings.

4.2. Comparisons to Baselines

In the first set of experiments, we compare the results of SRT to previous methods for novel view synthesis. For these experiments, we restrict our tests to the NMR and MSN datasets, since they provide an opportunity for *direct comparisons* to recently published methods in a synthetic setting with known ground truth. We run SRT with the same parameters on both datasets.

Baselines. We provide comparisons to two baselines that share aspects of our approach. The first baseline is PixelNeRF [49], which also pretrains a network to produce features from images. The second baseline is Light Field Networks (LFN) [38], which like our method uses a light field formulation rather than volumetric rendering. These methods have demonstrated favorable quantitative and qualitative results in comparison to other previous work [41, 37, 28, 22].

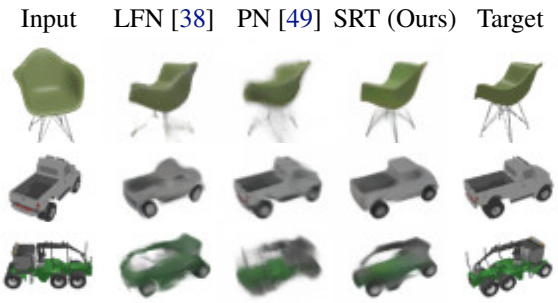

Figure 3. **Qualitative NMR results** – While PixelNeRF has high-quality for target views *close* to the input (middle), results quickly degrade when further away (bottom). The quality of SRT renders is much more consistent and outperforms both baselines. More results are available at srt-paper.github.io.

Quantitative Results – Tabs. 2 and 3. SRT provides the best image quality on NMR, outperforming LFN and PixelNeRF. In terms of computational performance (Tab. 3), both PixelNeRF and SRT are fast to encode new scenes, while LFN requires slow optimization of the scene embedding. Once a scene is encoded, both LFN and SRT render new frames at interactive rates, while PixelNeRF is significantly slower due to the volumetric rendering. SRT is the only method fast at both scene encoding and novel-view generation, making it suitable for the practical application of rendering videos of novel scenes at interactive rates. See Appendix A.2 further details. We note that SRT is comparably small for a transformer: its 74 M parameters consist of 23 M for CNN, 47 M for Encoder, and 4 M for Decoder. For comparison, ViT [6] has 86 - 632 M parameters.

On the MSN dataset, SRT outperforms LFN and PixelNeRF in terms of PSNR and SSIM. In terms of LPIPS, PixelNeRF outperforms SRT, which we attribute to the fact that SRT tends to blur regions of uncertainty. In a separate experiment, we tried increasing the latent dimension of LFN (from 256 to 1024) to account for more complex scenes, but this did not increase PSNR on novel scenes.

Qualitative Results – Figs. 3 and 4. The results show that our method outperforms both baselines on the NMR dataset. While all methods are able to synthesize high quality images for nearby views (Fig. 3, center), PixelNeRF renders degrade particularly quickly as the camera moves away from the input views (Fig. 3, bottom). LFN render quality degrades similarly, and it is also blurry for more complex objects. In contrast, SRT provides similar performance for a wide range of views. The MSN dataset (Fig. 4) with its complex scenes clearly demonstrates the limits of LFN's *single-latent* scene representation. While PixelNeRF results in detailed reconstructions for target views close to the inputs (Fig. 4 middle), SRT is the only method that allows for good reconstructions of far-away target views (Fig. 4 bottom).

Model Inspection – Fig. 5. Further insight can be gleaned

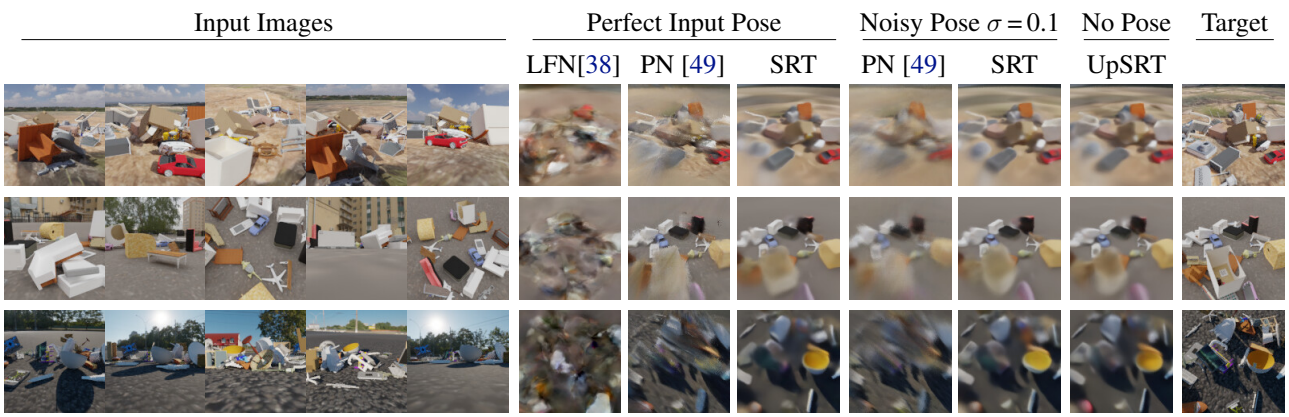

Figure 4. **Qualitative results on MultiShapeNet –** LFN does not scale to this demanding dataset due to its global latent conditioning. With perfect input camera poses, PixelNeRF resolves details in the center of the scene more sharply for target views nearby the inputs (middle). This quickly changes for views further away (bottom), where PixelNeRF produces projection artifacts even with perfect pose, while SRT's results are more coherent. PixelNeRF further has trouble compensating for noisy cameras, where SRT only experiences a mild drop in quality. Finally, UpSRT is the only model that can be run without input camera poses at all (see Sec. 4.4).

by investigating the attention weights of the encoder and decoder transformers. We can see from Fig. 5 (top) that the patch with the small chair (marked with a green box) mostly attends into the same chair in the other input images, even when the chair is facing different directions. We further noticed that all patches additionally attend into a subset of patches positioned along the edges of the images (bottom corners of the first input image). We believe that the encoder learns to store global information in those specific patches.

A glance at the decoder supports this conjecture. When rendering a ray of the chair from a novel direction, the first attention layer only attends into the bottom edge of all input views, while the second layer attends into similar patches as the encoder, and into the bottom corners of the first input image. This learned 2-stage inference in the decoder appears to be crucial, as we noticed in prior experiments that a 1-layer decoder is significantly worse, while more than two layers do not lead to significant gains. The model appears to have learned a *hybrid global/local* conditioning pattern through back propagation, without explicit geometric projections.

4.3. Ablation Studies

We now perform a series of ablation studies on the MSN dataset, where we remove or substitute the main components of SRT and measure the change in performance to evaluate individual contributions of different design decisions.

No Encoder Transformer. Retraining the system without an encoder transformer (*i.e.*, the decoder attends directly into the output of the CNN) provides a significant drop from 23.41 to 21.64 PSNR. This suggests that the encoder transformer adds crucial capacity for inference, and that the choice to move compute from the decoder (which should be as small and fast as possible) to the encoder is a feasible design decision. Note that its computational overhead is negligible in most practical settings, as it is only a single feed-forward step, and only run once per novel scene, independent of the number of rendered frames.

Flat Latent Scene Representation. One of the main novelties of SRT is the *set-latent* scene representation. We investigate the more commonly used flat latent by feeding the mean patch embedding of the encoder transformer into a large 8-layer MLP and dropping the decoder transformer. This architecture leads to a significant drop from 23.41 down to 20.88 in PSNR, showing the strength of a large set scene representation along with an attention-decoder.

Volumetric Rendering. As an alternative to the light field formulation, we also investigate a volumetric parametrization. To this end, we simply query the decoder not with rays, but rather with 3D points, followed by the volumetric rendering [25]. For simplicity, we do not inject viewing directions, and only use a single *coarse* network. We call this variation V-SRT. While this variation leads to an explicit 3D volume with easy-to-visualize depth maps, and results are visually comparable to SRT, it is to be mentioned that the V-SRT's decoding is slower by a theoretical factor of $192\times$, the number of samples per ray, making inference time of this variation more similar to existing volumetric methods [49]. It is notable that our model architecture can be trained in both setups without further changes to architecture or hyper-parameters. See details in Appendix A.3.

4.4. Robustness Study

While perfect pose information is available in synthetic setups, real-world applications often depend on estimated camera poses (*e.g.*, [35]), which is slow and often contains errors [21]. We therefore study SRT and baselines in a

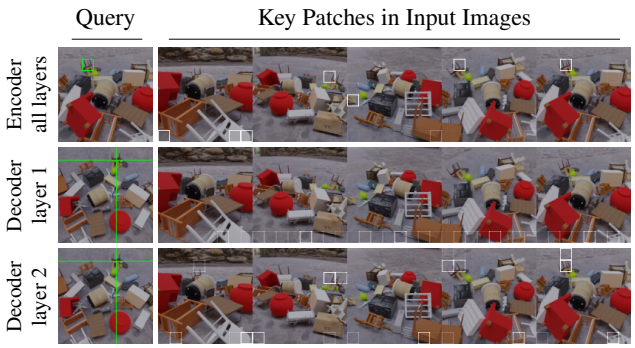

Figure 5. **Attention visualization** – Input patches that the green input patch attends into in the encoder, and the first & second decoder layers attend into when rendering the marked query ray at the intersection of the green lines. The model learns to attend into the same 3D positions, and to store global information into specific tokens (along the bottom edge). The decoder first attends into the global patches, then into relevant 3D positions of the scene.

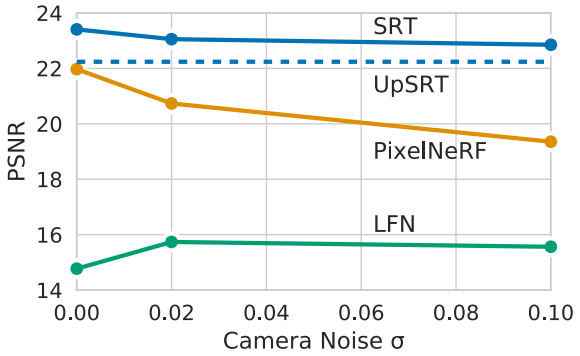

Figure 6. **Robustness** – PixelNeRF [49] quality drops quickly even for minor amounts of noise despite being trained for this setting. LFN [38] fails to scale to MSN even with perfect cameras – some noise regularizes the model and actually leads to a small increase in quality. SRT handles camera noise gracefully, outperforming competing methods even without any camera parameters (UpSRT).

noisy camera pose regime. To this end, we follow [21] and synthetically perturb all input camera poses $\mathbf{c}_i$ for $i \neq 0$ (all but the reference camera, see Sec. 3) with additive noise $\delta\mathbf{c}_i = \mathcal{N}(0, \sigma)$ to various degrees. For each value of σ, we retrain all models from scratch to allow them to adjust to the noise. To allow PixelNeRF to distinguish the reference input camera $\mathbf{c}_0$ from the others, we add learned camera identity embeddings to the output of the CNN, see Sec. 3.

PixelNeRF heavily relues on accurate camera poses to perform projections from 3D volumes into 2D images during rendering. As expected, Fig. 6 shows that the performance of this method degrades sharply even with small amounts of noise. LFN does not scale to the MSN dataset (see Fig. 4), and we note that minor amounts of noise actually increase PSNR due to a regularization effect. As LFN depends on test-time optimization for the scene latent, it is not robust to noisy camera positions.

In contrast, SRT handles noisy poses much more gracefully. Taking it to the extreme, **our method even works with fully un-posed imagery, still outperforming both baselines**. We call this model variation without test-time poses UpSRT. Further inspection shows that UpSRT is not only trivially using the first input image (for which the pose is known by definition, see Sec. 3) but the model is in fact learning to use all input cameras, even the un-posed ones. This is evidenced by inspection of attention patterns, and by the fact that UpSRT significantly outperforms SRT with a single input view only, see Appendix A.3.

4.5. Applications

In this section, we investigate the use of SRT in applications. For these experiments, we use the Street View dataset, which contains images of real-world outdoor environments. To compensate for variations in appearance and exposure present in this dataset, we augment the network with an appearance encoder, see Appendix A.2 for details.

View Interpolation. The goal in this application is to interpolate images captured in Street View panoramas to provide smooth video transitions between them. This application requires not only accurate view synthesis, but also temporal coherence between nearby views. We show a representative sample of interpolated views in Fig. 7. Although the renders are blurrier than the input views, our results demonstrate that SRT scales to complex real-world scenes with nontrivial camera pose distributions. It also shows that the model learns enough 3D scene information to render novel views far away from the inputs (last column). We provide videos and further results showing temporal coherence in render videos in Appendix A.3.

Semantic Segmentation. Our goal in this application is to predict dense semantics for novel views of outdoor scenes. Rather than synthesizing the color images for novel views and then employing a 2D image segmentation network on them, we show that SRT's scene representation trained on RGB can be leveraged more directly in new domains. Once SRT has been trained for the RGB reconstruction task, we freeze the encoder, and train a new decoder transformer to synthesize semantic segmentation images from the frozen scene representation directly. The semantic decoder has the same network architecture as the color decoder, except for the final output layer that is changed from 3 RGB channels to 46 semantic classes. We train the semantic decoder using the standard multi-class cross-entropy loss in a semi-supervised setup, see Appendix A.2.

Example semantic segmentation results are shown in Fig. 7. These results suggest that the scene representation learned through a color reconstruction loss contains enough information about the scene to allow semantic reasoning.

Figure 7. **Qualitative results on Street View** – SRT performs reasonably on highly challenging real-world data with small and large changes in camera perspective. Furthermore, the scene representation contains enough information for 3D semantic scene inference. A comparison with NeRF-based optimization methods is provided in Appendix A.3.

Though this is just one example of many possible downstream tasks, it is a significant finding that SRT has learned a scene representation useful for a non-trivial application.

5. Limitations

This project investigates whether transformers can learn scalable scene representations from only images without explicit geometry processing. We here identify a few limitations, which we believe to be tackled by follow up works.

First, our results show some blurriness in images on the complex datasets, even for views close to the input cameras. This is expected, as the model needs to learn pixel-accurate light ray transformations, which leads to uncertainty about exact positions, known to result in blurriness under an L2 loss [33].

Second, SRT is a geometry-free learning-based method, and hence will not work as well on very small datasets compared to methods with explicit geometric inductive biases. In practice, we find that our model converges at rates similar to others, but even more training usually results in better performance. In particular, when trained on the standard datasets reported in this paper, SRT outperforms LFN and PixelNeRF. Further investigation is required to better understand the best training protocols.

Finally, in comparisons to prior art, our model is best when input views are sparse or when camera poses are noisy or missing. When novel views are reliably close to input views, and perfect poses are known, explicit geometric methods such as [49] often provide better results, albeit with much longer inference times. While SRT has been specifically designed for the sparse input scenario, future work could investigate how to better leverage nearby input views at inference time when they are available.

6. Conclusion

We propose Scene Representation Transformer, a model for learning scalable neural scene representations using only self-supervision from color images. The novel encoder-decoder transformer architecture learns to render novel views without explicit geometric reasoning, and optionally without 3D image poses, yet surpassing the quality of prior art on standard benchmarks and a novel, more challenging dataset that we propose. At the same time, SRT is orders of magnitudes faster than prior methods for the realistic settings such as novel-scene video generation. This unique combination of flexibility, generality, and efficiency is well-suited for real world applications with very large datasets, and we believe that this work will inspire the community towards similar more implicit, yet scalable methods.

7. Acknowledgments

We thank Adam Kosiorek, Alex Yu, Alexander A. Kolesnikov, Aravindh Mahendran, Luke Barrington, Konstantinos Rematas, Sebastian Ebert, Srinadh Bhojanapalli Vickie Ye, and Vincent Sitzmann for their help and fruitful discussions.

References

[1] Sameer Agarwal, Yasutaka Furukawa, Noah Snavely, Ian Simon, Brian Curless, Steven M Seitz, and Richard Szeliski. Building rome in a day. *Communications of the ACM*, 2011.

[2] PixelNeRF authors. Official code. https://github.com/sxyu/pixel-nerf, 2021.

[3] Jonathan Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul Srinivasan. Mip-NeRF: A Multiscale Representation for Anti-Aliasing Neural Radiance Fields. In *ICCV*, 2021.

[4] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015.

[5] Anpei Chen and Zexiang Xu. MVSNeRF: Fast Generalizable Radiance Field Reconstruction From Multi-View Stereo. In *ICCV*, 2021.

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

[7] SM Ali Eslami, Danilo Jimenez Rezende, Frederic Besse, Fabio Viola, Ari S Morcos, Marta Garnelo, Avraham Ruderman, Andrei A Rusu, Ivo Danihelka, Karol Gregor, et al. Neural scene representation and rendering. *Science*, 2018.

[8] Patrick Esser, Robin Rombach, and Bjorn Ommer. Taming transformers for high-resolution image synthesis. In *CVPR*, 2021.

[9] Stephan Garbin and Marek Kowalski. FastNeRF: High-Fidelity Neural Rendering at 200FPS. In *ICCV*, 2021.

[10] Google. Street view, 2007. URL www.google.com/streetview/.

[11] Klaus Greff, Francois Belletti, Lucas Beyer, Carl Doersch, Yilun Du, Daniel Duckworth, David J Fleet, Dan Gnanapragasam, Florian Golemo, Charles Herrmann, Thomas Kipf, Abhijit Kundu, Dmitry Lagun, Issam Laradji, Hsueh-Ti (Derek) Liu, Henning Meyer, Yishu Miao, Derek Nowrouzezahrai, Cengiz Oztireli, Etienne Pot, Noha Radwan, Daniel Rebain, Sara Sabour, Mehdi S. M. Sajjadi, Matan Sela, Vincent Sitzmann, Austin Stone, Deqing Sun, Suhani Vora, Ziyu Wang, Tianhao Wu, Kwang Moo Yi, Fangcheng Zhong, and Andrea Tagliasacchi. Kubric: A Scalable Dataset Generator. In *CVPR*, 2022.

[12] Peter Hedman, Pratul Srinivasan, Ben Mildenhall, Jonathan Barron, and Paul Debevec. Baking Neural Radiance Fields for Real-Time View Synthesis. In *ICCV*, 2021.

[13] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.

[14] Hosted by Niemeyer et al. *NMR Dataset*, 2021. URL https://s3.eu-central-1.amazonaws.com/avg-projects/differentiable_volumetric_rendering/data/NMR_Dataset.zip.

[15] Ajay Jain, Matthew Tancik, and Pieter Abbeel. Putting NeRF on a Diet: Semantically Consistent Few-Shot View Synthesis. In *ICCV*, 2021.

[16] Wongbong Jang and Lourdes Agapito. CodeNeRF: Disentangled Neural Radiance Fields for Object Categories. In *ICCV*, 2021.

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

[18] Salman H. Khan, Muzammal Naseer, Munawar Hayat, Syed Waqas Zamir, Fahad Shahbaz Khan, and Mubarak Shah. Transformers in vision: A survey. In *CoRR*, 2021.

[19] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

[20] Adam Kosiorek, Heiko Strathmann, Daniel Zoran, Pol Moreno, Rosalia Schneider, Sona Mokrá, and Danilo Rezende. NeRF-VAE: A Geometry Aware 3D Scene Generative Model. In *ICML*, 2021.

[21] Chen-Hsuan Lin, Wei-Chiu Ma, Antonio Torralba, and Simon Lucey. BARF: Bundle-Adjusting Neural Radiance Fields. In *ICCV*, 2021.

[22] Shichen Liu, Tianye Li, Weikai Chen, and Hao Li. Soft rasterizer: A differentiable renderer for image-based 3d reasoning. In *ICCV*, 2019.

[23] Ricardo Martin-Brualla, Noha Radwan, Mehdi S. M. Sajjadi, Jonathan Barron, Alexey Dosovitskiy, and Daniel Duckworth. NeRF in the Wild: Neural Radiance Fields for Unconstrained Photo Collections. In *CVPR*, 2021.

[24] Moustafa Meshry, Dan B Goldman, Sameh Khamis, Hugues Hoppe, Rohit Pandey, Noah Snavely, and Ricardo Martin-Brualla. Neural rerendering in the wild. In *CVPR*, 2019.

[25] Ben Mildenhall, Pratul Srinivasan, Matthew Tancik, Jonathan Barron, Ravi Ramamoorthi, and Ren Ng. NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. In *ECCV*, 2020.

[26] Piotr Mirowski, Matt Grimes, Mateusz Malinowski, Karl Moritz Hermann, Keith Anderson, Denis Teplyashin, Karen Simonyan, Andrew Zisserman, Raia Hadsell, et al. Learning to navigate in cities without a map. *NeurIPS*, 2018.

[27] Przemyslaw Musialski, Peter Wonka, Daniel G Aliaga, Michael Wimmer, Luc Van Gool, and Werner Purgathofer. A survey of urban reconstruction. In *Comput. Graph. Forum*, 2013.

[28] Michael Niemeyer, Lars Mescheder, Michael Oechsle, and Andreas Geiger. Differentiable Volumetric Rendering: Learning Implicit 3D Representations without 3D Supervision. In *CVPR*, 2020.

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

[30] Christian Reiser, Songyou Peng, Yiyi Liao, and Andreas Geiger. KiloNeRF: Speeding Up Neural Radiance Fields With Thousands of Tiny MLPs. In *ICCV*, 2021.

[31] Andrea Romanoni, Daniele Fiorenti, and Matteo Matteucci. Mesh-based 3d textured urban mapping. In *IROS*, 2017.

[32] Robin Rombach, Patrick Esser, and Björn Ommer. Geometry-free view synthesis: Transformers and no 3d priors. In *ICCV*, 2021.

[33] Mehdi S. M. Sajjadi, Bernhard Scholkopf, and Michael Hirsch. Enhancenet: Single image super-resolution through automated texture synthesis. In *ICCV*, 2017.

[34] Mehdi SM Sajjadi, Rolf Köhler, Bernhard Schölkopf, and Michael Hirsch. Depth estimation through a generative model of light field synthesis. In *German Conference on Pattern Recognition*, 2016.

[35] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[36] Inwook Shim, Yungeun Choe, and Myung Jin Chung. 3d mapping in urban environment using geometric featured voxel. In *URAI*, 2011.

[37] Vincent Sitzmann, Michael Zollhöfer, and Gordon Wetzstein. Scene Representation Networks: Continuous 3D-Structure-Aware Neural Scene Representations. In *NeurIPS*, 2019.

[38] Vincent Sitzmann, Semon Rezchikov, William T Freeman, Joshua B Tenenbaum, and Fredo Durand. Light field networks: Neural scene representations with single-evaluation rendering. In *NeurIPS*, 2021.

[39] Ayush Tewari, Vincent Sitzmann, Stephen Lombardi, Kalyan Sulkavalli, Ricardo Martin-Brualla, Tomas Simon, Matthias Nießner, Gordon Wetzstein, Christian Theobalt, Dan Goldman, and Michael Zollhöfer. State of the Art on Neural Rendering. In *Eurographics*, 2020.

[40] Ayush Tewari, Justus Thies, Ben Mildenhall, Pratul Srinivasan, Edgar Tretschk, Yifan Wang, Christoph Lassner, Vincent Sitzmann, Ricardo Martin-Brualla, Stephen Lombardi, et al. Advances in neural rendering. *arXiv preprint arXiv:2111.05849*, 2021.

[41] Alex Trevithick and Bo Yang. GRF: Learning a General Radiance Field for 3D Scene Representation and Rendering. In *ICCV*, 2021.

[42] Linh Truong-Hong and Debra F Laefer. Octree-based, automatic building facade generation from lidar data. In *Computer-Aided Design*, 2014.

[43] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017.

[44] Qianqian Wang, Zhicheng Wang, Kyle Genova, Pratul Srinivasan, Howard Zhou, Jonathan Barron, Ricardo Martin-Brualla, Noah Snavely, and Thomas Funkhouser. IBRNet: Learning Multi-View Image-Based Rendering. In *CVPR*, 2021.

[45] Ting-Chun Wang, Alexei A Efros, and Ravi Ramamoorthi. Occlusion-aware depth estimation using light-field cameras. In *ICCV*, 2015.

[46] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. In *IEEE TIP*, 2004.

[47] Tao Yan, Fan Zhang, Yiming Mao, Hongbin Yu, Xiaohua Qian, and Rynson WH Lau. Depth estimation from a light field image pair with a generative model. *IEEE Access*, 2019.

[48] Alex Yu, Ruilong Li, Matthew Tancik, Hao Li, Ren Ng, and Angjoo Kanazawa. PlenOctrees for Real-Time Rendering of Neural Radiance Fields. In *ICCV*, 2021.

[49] Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. pixelNeRF: Neural Radiance Fields from One or Few Images. In *CVPR*, 2021.

[50] 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.

[51] Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip HS Torr, and Vladlen Koltun. Point transformer. In *ICCV*, 2021.

[52] Tinghui Zhou, Richard Tucker, John Flynn, Graham Fyffe, and Noah Snavely. Stereo Magnification: Learning View Synthesis using Multiplane Images. In *Siggraph*, 2018.