Ray-Patch: An Efficient Querying for Light Field Transformers

Tomás Berriel Martins   Javier Civera
I3A, University of Zaragoza, Spain
{tberriel,jcivera}@unizar.es [*]

Abstract

In this paper we propose the Ray-Patch querying, a novel model to efficiently query transformers to decode implicit representations into target views. Our Ray-Patch decoding reduces the computational footprint and increases inference speed up to one order of magnitude compared to previous models, without losing global attention, and hence maintaining specific task metrics. The key idea of our novel querying is to split the target image into a set of patches, then querying the transformer for each patch to extract a set of feature vectors, which are finally decoded into the target image using convolutional layers. Our experimental results, implementing Ray-Patch in 3 different architectures and evaluating it in 2 different tasks and datasets, demonstrate and quantify the effectiveness of our method, specifically a notable boost in rendering speed for the same task metrics.

1. Introduction

Autonomous agents rely typically on explicit representations of the environment for localization and navigation, such as point clouds [5, 46], or voxels [3, 34]. However, such approaches lack topological or semantic information, struggle to generalize to changes to novel viewpoints, and do not scale properly to tasks that require reasoning about 3D geometry and affordances. Autonomous agents require semantic, meaningful, and informative representations to properly understand and interact with their environment and perform complex tasks [4, 39].

Implicit representations are better suited to reasoning and are then relevant, as they capture in a continuous space the main high-level features of the scene. Many approaches focus on 3D geometry without topological restrictions using learned occupancy or signed distance functions [9, 15, 27, 28, 35, 36]. Nevertheless, the recent success of neural fields to encode the tridimensional geometry and

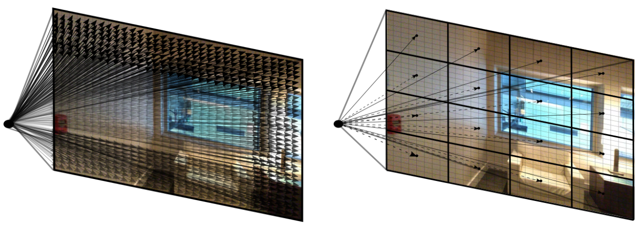

Figure 1. Light Field Networks sample a ray per pixel to render the target image (left). Our Ray-Patch (right) groups pixels in $k \times k$ patches and samples a ray per patch, reducing the querying cost by a factor of k^2 without loosing accuracy.

lighting of a scene has revolutionized the field [47]. Although Neural Radiance Fields (NeRFs) focused initially on learning colour and occupancy models in a 3D space [29], they have demonstrated promise in a wide array of tasks such as scene segmentation [6, 21, 57], depth estimation [18], SLAM [1, 44, 58], scene editing [2, 17, 20, 23, 33, 43], and many more [47].

The main limitations of neural rendering are 1) the exhaustive querying of the model that is required to recover each pixel of a specific viewpoint, and 2) the need to fit the NeRF model for each scene. Several approaches reduce the 3D querying cost using depth [13, 24, 37, 52], geometry [8, 48, 51, 55], or changing the discretization [25, 30, 54], and avoid per-scene optimization using latent vectors [8, 18, 23, 25, 48, 51, 55]. Among them, the extensions of Light Field Networks (LFNs) [42] with transformers (Light Field Transformers or LFTs) [18, 40, 41] have shown potential to solve both limitations, although they are constrained by the quadratic scaling of attention. Despite recent attempts to reduce it, these either modify the attention algorithm for a less expensive but less effective version [45, 53, 56]; or are based on extensive optimization of last generation hardware and software [11, 12]. Therefore, despite significant advances in both qualitative performance and efficiency, all these approaches are still far from being scalable to real scenarios with real-time performance.

[*]This work was supported by the Spanish Government (PID2021-127685NB-I00 and TED2021-131150B-I00) and the Aragón Government (DGA-T45 17R/FSE).

In this work we propose Ray-Patch, a novel decoding method that reduces the computation and memory load of LFTs up to one and two orders of magnitude respectively, while keeping the quality of the output. We developed Ray-Patch as a generic decoder that can be implemented on any LFT architecture in the literature. Instead of the typical per-pixel querying, we group all pixels in a square patch, as shown in Fig. 1, and compute a set of feature vectors, which are then grouped and decoded into the target viewpoint. Specifically, it combines a transformer decoder with convolutional neural networks to reduce the cost of the decoder processing. This results in a drastic reduction in the number of queries, which impacts quadratically in the cost, allowing to decode high-resolution images while keeping and sometimes even improving the training convergence.

2. Related Work

A NeRF [29] is an implicit representation of a scene that is learnt from a sparse set of multiple views of such scene, annotated with their corresponding camera poses. NeRFs encode a continuous volumetric model of a scene that can be used to render photorealistic novel views from arbitrary viewpoints. The rendering process involves projecting pixels into rays, sampling 3D positions along the rays, and querying a Multilayer Perceptron (MLP) network that predicts the colour and occupancy of the sampled 3D points. Despite its versatility and impressive results in various applications [47], NeRFs suffer from two major limitations: exhaustive 3D sampling is required to decode each pixel, and a new model must be trained for each new scene.

Multi-scene implicit representations. One of the most promising approaches to enable the generalization of neural fields across multiple scenes is conditioning the output of the MLP to a latent vector that is optimized for each scene at test time. NSVF [25] discretizes the 3D space into a sparse voxel octree associating each voxel with a feature vector that guides the sampling of 3D points. Control-NeRF [23] also utilizes voxel features, but employs a multi-resolution incremental training of the full feature volume. Nice-SLAM [58] leverages a multi-resolution feature grid to encode the scene while simultaneously performing camera tracking. InstantNGP [30] implements multi-resolution voxelization as a hash encoding, where the MLP is responsible for avoiding hash collisions, resulting in remarkable improvements in reconstruction quality and convergence.

Other approaches involve using an encoder architecture to compute latent vectors, and use these to condition the NeRF decoder. GRF [48] projects sampled 3D points into feature maps of the input views computed with a CNN encoder-decoder. These are first processed by shared MLP to condition on the sampled 3D point, and then aggregated using an attention module. The final feature vector is fed to a final MLP to estimate the 3D point colour and density. PixelNeRF [55] extends this approach by adding the feature vector as a residual in each layer of the MLP and using a simple average pooling instead of an attention module. IBRNet [51] also projects 3D points into nearby views to estimate the conditioning feature vectors, although it relies on a differentiable rendering algorithm rather than a neural representation to estimate the final colour and depth. MVS-NeRF [8] proposes the use of a CNN and homography re-projections to build a cost volume and compute the features. ENeRF [24] builds on MVS approaches to build the cost volume, guide the sampling, and condition the reconstruction. However, these methods that rely on homographies are limited to a small range of views in front of the reference camera, require accurate camera poses, and are not robust to occlusions.

SRT [41] introduced a Transformer [49] to encode and decode the scene, performing self-attention between the features of different points of view. It generates a latent representation of the scene, which is decoded using a light-field cross-attention module. OSRT [40] improves SRT by disentangling the object of the scene and improving its control. DeFiNe [18] replaces the basic Transformer for a PerceiverIO [19] reducing the cost of self-attention and scaling to bigger resolutions. Despite the low rendering time achieved for new scenes and novel points of view, these methods computation cost scales poorly due to the use of attention. Consequently, they may not be suitable for large scenes with high-resolution images.

Rendering from implicit neural representations. To render a pixel, NeRF evaluates 3D coordinates sampled along a ray combining both uniform and stratified distribution. This random sampling results in a great number of evaluation wasted on empty space. DONeRF [32] trains an oracle network supervised on dense depth map on synthetic data. Although it achieves outstanding results, the setup is hard to generalize to real environments. NerfingMVS [52] trains a monocular depth estimation network, supervised with sparse depth from Structure from Motion (SfM), to guide the sampling and reduce empty queries both at training and test. Roessle *et al.* [37] instead uses a depth completion network to estimate depth and uncertainty from the sparse SfM prior at training time, improving performance while still requiring multiple sampling at test time. In contrast, ENeRF [24] uses an estimated cost volume to predict the depth and guide the sampling without any explicit depth supervision nor structure from motion. Other alternative like Neural RGB-D [1] and Mip-NeRF RGB-D [14], directly use RGB-D sensor as prior for the depth sampling. Despite reducing the number of samples, most of these approaches still perform multiple samples per pixel due to the error range of depth data and estimation. Instead,

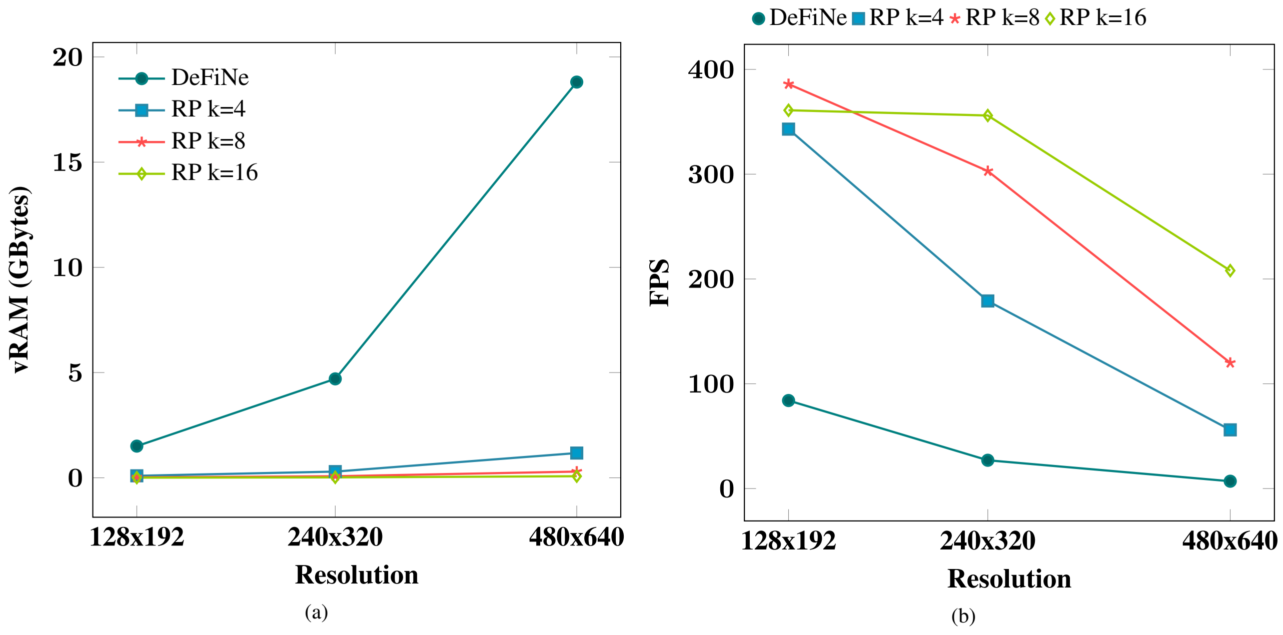

Figure 2. *(a)* **Peak GPU vRAM usage due to attention** for decoding a single image. vRAM usage scales linearly with the number of pixels (quadratically with resolution). Ray-Patch querying reduces $\times 10$ required resources on standard resolutions. Note that x-axis is in logarithmic scale. *(b)* **Single image rendering speed scaling.** The use of the Ray-Patch decoder increase rendering speed at high resolutions up to real-time for DeFiNe. To keep a fixed rendering speed, the patch size should increase at the same pace as the number of pixels.

Light Field Network [42] directly evaluate the unprojected pixels, parameterized as a ray, reducing the model query to the number of pixels. Although, this approach is able to perform real-time rendering of novel views without requiring heavy optimizations, it does not generalize yet to high resolution scenes.

3. Preliminaries: Light Field Transformers

While NeRFs learn a scene representation associated to a continuous space of 3D points, Light Field Networks (LFNs) [42] rely on 3D rays parametrized with Plücker coordinates to learn similar representations. This subtle difference reduces significantly the cost to decode a view of the scene, from several samples to a single sample per pixel. Despite this, LFNs are limited to simple settings, do not enforce geometric consistency and their ray parameterization is not robust to occlusions.

Light Field Transformers (LFTs) are an extension of LFNs which use a transformer architecture, a ray parametrization robust to occlusions, enforce geometric consistency through the training procedure, and encode and decode points of view of a scene without per-scene optimization [18, 40, 41].

3.1. Transformers

Transformers [49] are deep encoder-decoder neural models that incorporate attention mechanisms in their architecture. The encoder first performs self-attention on a set of tokens to extract common features. The decoder then uses cross-attention between the extracted features and a set of queries to generate an output per query. The attention block consists of a Multi-Head Attention (MHA) layer, followed by a Feed-Forward (FF) layer, with a skip-connection and layer normalization after each of them. In each head h, MHA operates in parallel a Scaled Dot-Product attention

$$\text{Attention}_h(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V, \qquad (1)$$

over a set of the three inputs: keys (K), values (V), and queries (Q). Each head linearly projects the inputs to reduced dimensions, d_k for Q and K and d_v for V, performs the attention operation, and then projects the output back to its original dimension. To perform self-attention $Q = K = V$ are the tokens to encode. Instead for cross-attention $K = V$ are the extracted features, while Q is the queries to decode.

Computational complexity. Linear projections have a complexity of $\mathcal{O}(nd_0d_p)$ with n the length of the sequence and d_0 and d_p the dimensions before and after the projection. Instead, the scaled-dot product has $\mathcal{O}(n_qn_{kv}d_k)$ complexity, being n_q and n_{kv} the number of queries and keys/values respectively. For self-attention, $n_q = n_{kv}$ and then the complexity is $\mathcal{O}(n_q^2d_k)$.

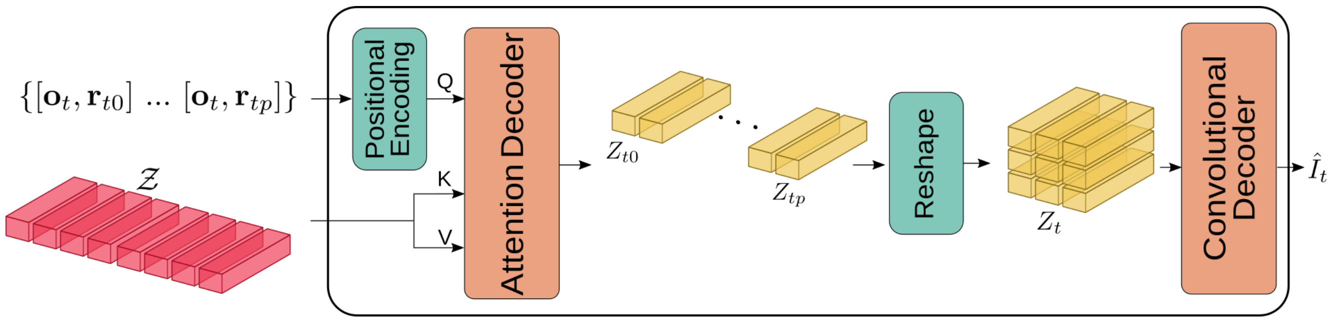

Figure 3. **Ray-Patch querying.** Given a latent representation of a scene $\mathcal{Z}$, in order to render an image I_t of shape $h \times w$, a query is performed for each patch p. Each patch is parametrized with a ray $\mathbf{r}_{tp}$ that passes through it and the the camera position $\mathbf{o}_t$. The queries are encoded with multiple Fourier frequencies, and fed to the attention decoder to compute a feature vector per query. The feature vectors of the image are re-shaped as a rectangle and forward-passed through the convolutional decoder to obtain the target image render $\hat{I}_t$

3.2. Scene Representation Transformer

The Scene Representation Transformer (SRT) [41] is an encoder-decoder LFT, which parametrizes rays with its 3D coordinates and their origin position. Given a set of N input views $\{I_n\}$[1], and their relative camera poses $\{P_n\}$ with camera Instrinsic parameters $\{K_n\}$, the encoder $\mathcal{E}$ generates a set-latent scene representation (SLSR)

$$\mathcal{Z} = \mathcal{E}\left(\{I_n, P_n\}\right), \tag{2}$$

To decode a view of the scene, the light-field based decoder is queried. Each query refers to the ray direction and camera center for a given pixel, and recovers its RGB values. To decode a full view, as many queries as pixels are needed.

The encoder is made of two parts. First, a convolutional network extracts features from the scene images. Then a set self-attention blocks computes common features between the multiple views of the scene to generate a SLSR. The decoder is a two-blocks cross-attention module. It performs attention between the ray queries and the SLSR to generate the RGB pixel values. SRT has been extended by OSRT [40] to disentangle its latent representation by integrating it with Slot-Attention [26] and designing the Slot Mixer Decoder. Using Slot Mixer attention weights, OSRT is able to generate unsupervised segmentation masks.

Attention cost. With a convolutional encoder which halves the resolution (divides by four the number of queries) three times, $n_q = n_{kv} = N\frac{h \times w}{64}$ for the encoder self-attention block. Therefore the complexity is

$$\mathcal{O}\left(\left(\frac{Nhw}{64}\right)^2 d_k\right). \tag{3}$$

Instead, for the decoder cross-attention block to decode an image, $n_q = h \times w$ and $n_{kv} = N\frac{h \times w}{64}$, therefore the complexity is

$$\mathcal{O}\left(\frac{N\,(hw)^2}{64} d_k\right). \tag{4}$$

Doubling the resolution will increase by a factor of 4 the total number of pixels $h \times w$, and by 16 the computational complexity. As a consequence, both SRT and OSRT are limited due to the quartic scaling of the attention cost with respect to the resolution of the images, and to the quadratic cost with respect to the number of input images N.

3.3. Depth Field Network

The Depth Field Network (DeFiNe) [18] can be considered as an extension of SRT. As its main novelties, the convolutional encoder is a pretrained ResNet-18, the cross-attention decoder is reduced from two blocks to one, and a set of geometric data augmentations are proposed for stereo and video depth training. The main contribution is the use of a PerceiverIO [19] instead of the self-attention encoder to use a SLSR with a fixed size n_l.

Attention cost. With $n_{kv} = n_l$ in both the encoder and decoder, the quadratic scaling with respect to the resolution of the images is reduced to

$$\mathcal{O}\left(\frac{Nhw}{64} n_l d_k\right) \tag{5}$$

for the encoder attention process, and to

$$\mathcal{O}\left(hwn_l d_k\right), \tag{6}$$

for decoding an image. These improvements reduce the cost considerably for $hw >> n_l$, although there is still a quadratic dependence with the number of pixels (quartic with resolution) that limit the model's use.

[1]We abuse notation here for simplicity, $\{\circ_n\} \equiv \{\circ_1, \ldots, \circ_N\}$

4. Our Method: The Ray-Patch Decoding

We propose the Ray-Patch querying to attenuate the quartic complexity of Light Field Transformers with respect to image resolution. Instead of using a ray to query the cross-attention decoder and generate a pixel value, we use a ray to compute a feature vector of a square patch of pixels. Then a transposed convolutional decoder unifies the different patches' feature vectors and recovers the full image. Our approach reduces the number of queries to $\frac{hw}{k^2}$ and the cross-attention cost by the same factor.

Parametrization. To decode a target view $I_t \in \mathbb{R}^{h\times w\times c}$ of the scene, the view is split into $\frac{hw}{k^2}$ square patches of size $[k,k]$, being the split image now defined as $\{I_{tp} \in \mathbb{R}^{\frac{h}{k}\times\frac{w}{k}\times 3}\}$. Each patch p is parametrized by the location of the camera $\mathbf{o}_t$, and the ray $\mathbf{r_{tp}}$ that passes both by the camera position and the center of the patch. Given the camera intrinsic K_t and extrinsic parameters ${}^{W}T^{C_t} = [R_t|\mathbf{o}_t] \in SE(3)$, the ray $\mathbf{r_{tp}}$ is computed as the unprojection of the center of patch p in the 2D camera plane. For each patch center in homogeneous coordinates $\mathbf{x}_{tp} = (u_{tp}, v_{tp}, 1)^T$, it is first unprojected in the the camera reference frame C_t,

$$\mathbf{r}_{tp}^{C_t} = K_n^{-1}\cdot\mathbf{x}_{tp} = [x_{tp}/z_{tp}, y_{tp}/z_{tp}, 1, 1]^T, \quad (7)$$

and after that it is translated to the world reference W,

$$\mathbf{r}_{tp}^{W} = {}^{W}T^{C_t}\cdot\mathbf{r}_{tp}^{C_t}. \quad (8)$$

Using Fourier positional encoding [29], the parametrization of each patch is mapped to a higher frequency, to generate a set of queries for the decoder.

$$\{\mathcal{Q}_{tp}\} = \{\gamma(\mathbf{o}_t)\oplus\gamma(\mathbf{r}_{tp})\} \quad (9)$$

Decoder. The decoder $\mathcal{D}$ is a composition

$$\mathcal{D} = (\mathcal{D}_{\text{CNN}}\circ\mathcal{D}_A) \quad (10)$$

of an attention decoder $\mathcal{D}_A$, followed by a convolutional decoder block $\mathcal{D}_{\text{CNN}}$. The attention decoder performs cross-attention between the queries $\{\mathcal{Q}_{tp}\}$ and the SLSR $\mathcal{Z}$, to compute a set of feature vectors

$$\{Z_{tp}\} = \mathcal{D}_A(\{\mathcal{Q}_{tp}\},\mathcal{Z}) \quad (11)$$

with dimension f. These vectors ensemble a feature map $Z_t \in \mathbb{R}^{\frac{h}{k}\times\frac{w}{k}\times f}$, which is decoded by the convolutional decoder into the target image

$$\hat{I}_t = D_{\text{CNN}}(Z_t), \quad (12)$$

as shown in Fig. 3. We use a vanilla convolutional decoder $\mathcal{D}_{CNN}$ based on GIRAFFE's decoder [33]. It is a combination of upsampling blocks with convolutions and preliminary outputs. The number of channels of the output, c, will vary depending on the desired task, *e.g.* $c = 3$ for RGB colour image, or $c = 1$ for depth estimation.

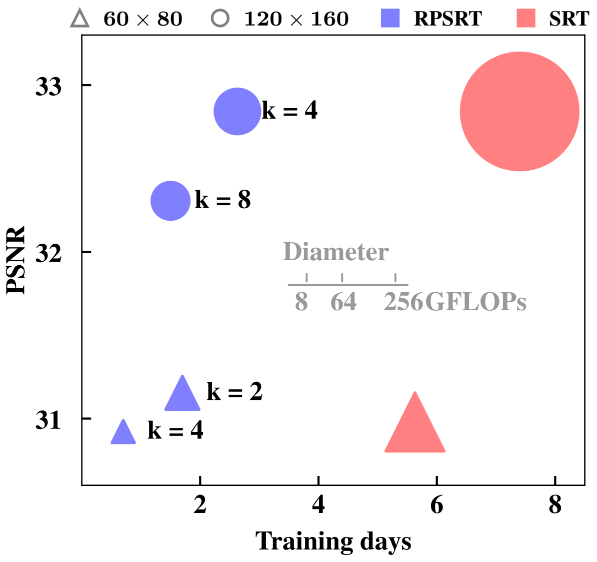

Figure 4. **Training time on one V100 comparison.** For both 60×80 (△) and 120×160 (○) resolution, Ray-Patch (*blue*) configurations, $k = \{2,4\}$ and $k = \{4,8\}$ respectively, achieve similar or better rendering performance than SRT (*red*), with $60-70\%$ cost reduction.

Integration. The simplicity of the Ray-Patch querying allows to easily integrate it in LFTs like SRT, OSRT, or DeFiNe. Changing the number of channels of the output of their decoders to f, they can be used as $\mathcal{D}_A$ to decode the final image as

$$\hat{I}_t = D_{cnn}(D_A(\{\mathcal{Q}_{tp}\},\mathcal{Z})). \quad (13)$$

The optimization process does not change. The model's parameters θ are optimized on a collection of images from different scenes minimizing the Mean Squared Error (MSE) of the generated novel-views for RGB images

$$\mathcal{L}_{rgb} = \frac{1}{hw}\sum_{ij}\left(\hat{I}_t - I_t\right)^2, \quad (14)$$

and minimizing the absolute log difference for depth maps

$$\mathcal{L}_d = \frac{1}{n_{tp}}\sum_{tp}|\log\hat{\mathbf{D}}_{tp} - \log\mathbf{D}_{tp}|, \quad (15)$$

Depth optimization is performed only over the subset $\{tp\}\subset\{ij\}$ of target pixels with depth info, giving freedom to the model to generalize to unseen parts.

Attention cost. The proposed Ray-Patch querying reduces the complexity of the decoders to

$$\mathcal{O}\left(\frac{N(hw)^2}{64k^2}d_k\right), \quad (16)$$

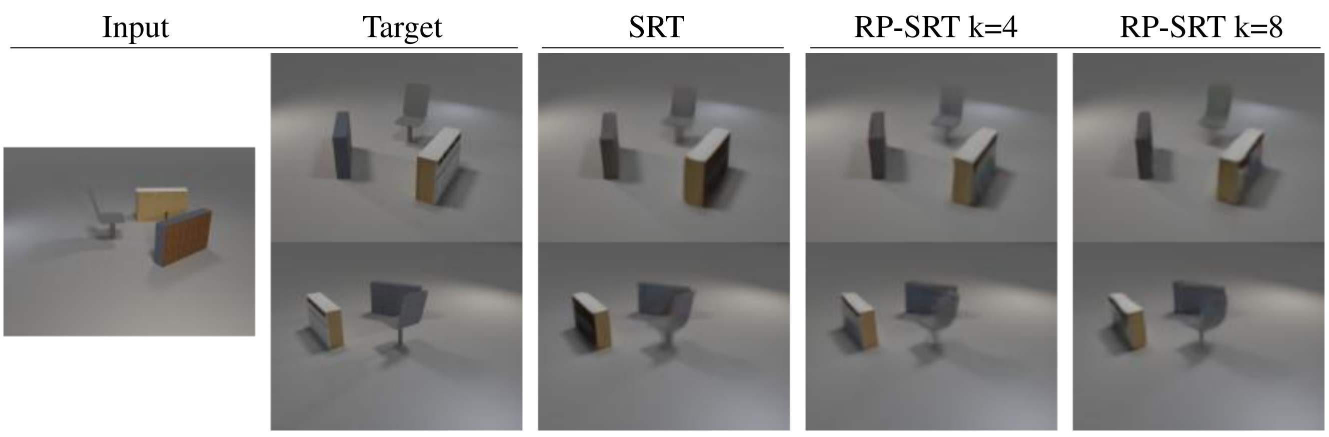

Figure 5. **Novel view synthesis results on MSN-Easy.** Given an input image, the models are queried to decode target images at 120º (first row) and 240º (second row). Both SRT and Ray-Patch models (RP-SRT $k = \{4, 8\}$) encode a coherent representation, with slight differences on colour and edges. For RP-SRT it can be seen how the bigger the patch size the more diffuse the image looks.

for models with the basic Transformer, like SRT and OSRT; and to

$$\mathcal{O}\left(\frac{hw}{k^2} n_l d_k\right), \tag{17}$$

for PerceiverIO based models, like DeFiNe. Although there is still a quadratic dependency on the resolution, the attenuation introduced by the Ray-Patch querying can reduce the number of queries in up to two orders of magnitudes for high resolutions.

5. Experimental Results

We evaluate Ray-Patch using two setups of different complexity. Firstly, we integrate Ray-Patch into both SRT and OSRT for novel view synthesis on the MultiShapeNet-Easy (MSN-Easy) dataset. Given input images, the model encodes a representation of the scene, and its goal is decoding the other two viewpoints. In this dataset we asses the impact of different patch sizes at different resolutions on SRT implementation, and its integration on OSRT. After that, we evaluate its ability to generalize to more challenging scenes and textures in a stereo depth task.

Secondly, we also implemented Ray-Patch into DeFiNe and evaluated on ScanNet. Given two images, the model encodes a representation, and the goal is recovering RGB and depth from the same point of view. Following Sajjadi *et al.* [40, 41], rendered views are benchmarked with PSNR, SSIM, and LPIPS; and segmentation masks with FG-ARI. Following Guizilini *et al.* [18], depths are benchmarked with Absolute Relative Error (Abs.Rel), Square Relative Error (Sq.Rel) and Root Mean Square Error (RMSE). Computational aspects are evaluated measuring peak RAM usage, image rendering speed as in Sajjadi *et al.* [41], training time, and Float Point Operations (FLOPs) needed to encode and render an image. We assume the use of float-32 data, and report time metrics from a GPU NVIDIA Tesla V100. Further design and implementation details are provided on the supplementary material.

5.1. Datasets

MultiShapeNet-Easy [43] has 70K training scenes and 10K test scenes with resolution 240×320. Due to the high cost of training both SRT and OSRT, we work at 60×80 and 120×160. In each scene there are between 2 and 4 objects of 3 different classes: chair, table, or cabinet. The object shapes are sampled from the ShapeNetV2 dataset [7]. Each scene has 3 views sampled at 120° steps on a circle around the center of the scene, with extrinsics and intrinsics camera annotations. For each training step, one image is used as input and the other two are used as target to be reconstructed.

ScanNet [10] is a collection of real indoor scenes with RGB-D and camera pose information. It has 1.2K different scenes with a total of 90K views. We follow DeFiNe's [18] stereo setup: RGB input images are downscaled to a resolution of 128×192; and a custom stereo split is used [22], resulting in 94212 training and 7517 test samples.

5.2. Computational performance

While our Ray-Patch querying still has quadratic scaling with n_q, the reduction we achieve in the number of queries results in a notable boost in rendering speed, as can be seen in Tab. 1 and Fig. 2b. Furthermore, when increasing the resolution the patch can also be increased, keeping an appropriate rendering speed at higher resolutions. Comparing rendering speeds for different patches and resolutions in Fig. 2b, it can be observed how the improvement tends to saturate for big patch sizes. As a consequence of reducing the number of queries, its impact on the scaled-dot product complexity will be out-weighted by n_{kv}. For $n_q << n_{kv}$, n_{kv} will set a minimum cost and increasing the patch size

	MSN-Easy							
	60 × 80			120 × 160			120 × 160	
	SRT	RP-SRT		SRT	RP-SRT		OSRT	RP-OSRT
		k = 2	k = 4		k = 4	k = 8		k = 8
↑ PSNR	30.98	**31.16**	30.92	**32.842**	32.818	32.306	30.95	**31.03**
↑ SSIM	0.903	**0.906**	0.901	0.934	**0.935**	0.929	**0.916**	0.915
↓ LPIPS	0.173	**0.163**	0.175	**0.250**	0.254	0.274	**0.287**	0.303
↑ FG-ARI	-	-	-	-	-	-	**0.958**	0.914
↓ Training time	5.6 days	1.7 day	**0.7 days**	7.4 days	1.7 days	**1 day**	25 days	**3.7 days**
↓ Giga FLOPs	48.2	15.8	**7.3**	192.1	28.5	**19.7**	278.6	**24.7**
↑ Rendering speed	117 fps	288 fps	**341 fps**	30 fps	275 fps	**305 fps**	21 fps	**278 fps**

Table 1. **Quantitative results on MSN-Easy.** Evaluation of new scene novel view synthesis and computational performance on a simple dataset. While SRT's performance is surpassed only by the configuration with patch size $k = 2$, Ray-Patch increases $\times 3$ and $\times 10$ the rendering speed with minimum impact.

over this limit will not be reflected on the rendering speed. It is also worth of attention that the biggest patch does not have the lower rendering time. When $n_q << n_{kv}$, increasing the patch size also adds more convolutions and interpolations to the convolutional decoder, hence increasing the deconvolutional overhead without reducing the cost of the attention decoder. Finally, the decrease in n_q implies a smaller memory peak in the softmax of the decoder attention, see Fig. 2a. This matrix is $n_q \times n_{kv}$. As an illustrative example, for DeFiNe, decoding a single 960×1280 image, with $n_{kv} = 2048$, requires 75 GBytes of GPU memory, almost two full A100 GPUs. Instead, for the Ray-Patch querying with $k = 16$, it is reduced to only 0.3 GBytes. This notable reduction allows to increase parallelization, improving even more the rendering speed for scene reconstruction tasks.

5.3. Novel view synthesis

On MSN-Easy, for SRT we evaluate two different patch sizes for each resolution: $k = \{2, 4\}$ for 60×80; and $k = \{4, 8\}$ for 120×160. Instead for OSRT we only evaluate at 120×160 with a patch size $k = 8$.

As reported in Tab. 1 and Fig. 5, the experiment metrics for RP-SRT shows that the size of the patch impacts on the model, with smaller patches having better rendering quality on both resolutions. For smaller patches, the first decoder focus attention on less pixels than for a bigger patch, each feature vector is up-sampled less, and more information is recovered from the same amount of data. Therefore, excessively increasing the patch reduces the quality of reconstructed views, as shown by RP-SRT with $k = 8$ for 120×160, which slightly underperforms the baselines's PSNR (32.3 vs 32.8). Nevertheless, Ray-Patch querying is still able to match rendering quality of both SRT and OSRT at 120×160, with $k = 4$ and $k = 8$ respectively; and outperform at 60×80, with $k = 2$. Furthermore, for similar performance our approach improves rendering speed $\times 10$ for the highest resolution (275 vs 30 fps, and 278 vs 21 fps), and reduces training time almost $\times 4$ (see Fig. 4 and Tab. 1). This is thanks to scaling the attenuation factor k together with resolution, compensating for the increasing number of queries. Regarding RP-OSRT's unsupervised segmentation, we up-sample the $\frac{120}{8} \times \frac{160}{8}$ attention weights of the Slot Mixer Decoder to generate a 120×160 segmentation map, achieving only slightly worse metrics than OSRT (0.914 vs 0.958 FG-ARI). Finally, note in Tab. 1 that even if increasing resolution improves rendering quality (higher PSNR and SSIM) for all models, the perceptual similarity metric gets worse (higher LPIPS). This implies that when working at low resolution, LPIPS is not able to appropriately evaluate the model representation as perceptual inconsistencies from the 3D representation are hard to distinguish due to the poor quality. Therefore the usefulness of Ray-Patch querying increases. Reducing the computational cost of LFTs not only speeds-up training and inference, it also opens the possibility to work with more expensive loss functions rather than simple L1 or L2 losses, *e.g.* using perceptual losses or adversarial discriminators, following current state of the art in image generation [16, 38].

5.4. Stereo depth

Based on the results of the previous section, the integration with DeFiNe to render at 480×640 has be done with $k = 16$. This value is chosen to have n_q close to n_{kv}, improving the computation efficiency without compressing too much the information. Our results shows that Ray-Patch improves the convergence of the model. This configuration not only reduces the computational cost, but also improves the view reconstruction and depth stereo estimation in all metrics reported in Tab. 2. Despite taking as input a 128×192 image, reconstructions are closer to the 480×640 target output recovering a similar quality while DeFiNe's look diffused and blurred (see Fig. 6). It can also be observed how the estimated depth is smoother, with less abrupt changes, while still preserving clear depth discontinuities. Regarding the computation, the evaluated configu-

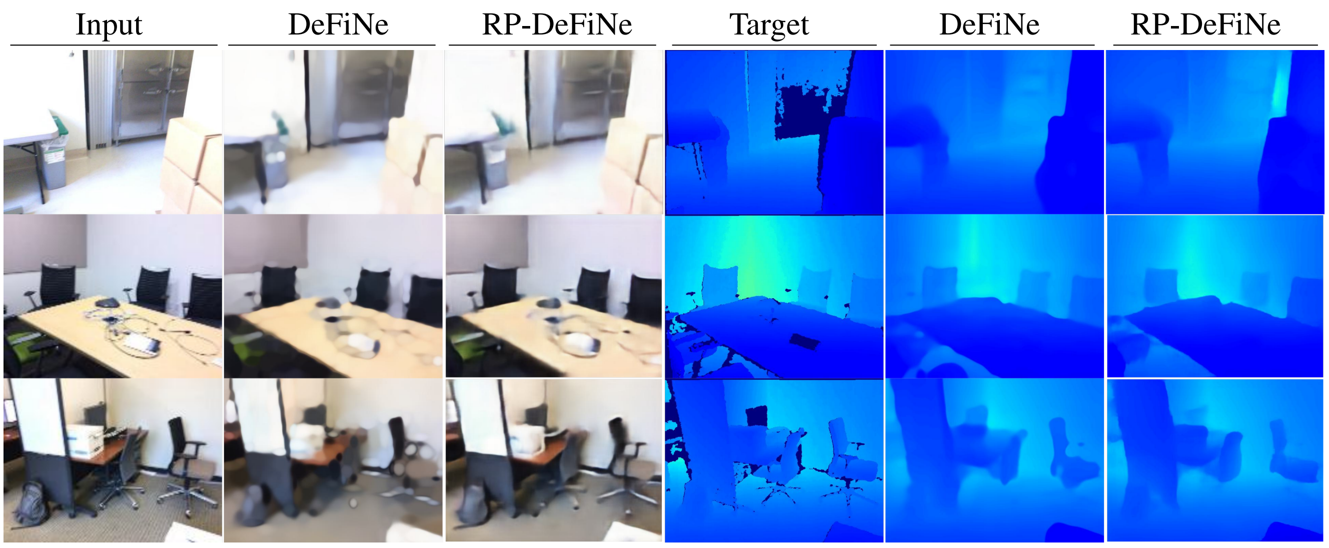

Figure 6. **Stereo depth results on ScanNet.** Given two input images, models decode both input images and an estimation of corresponding depth maps. Note how our $k = 16$ Ray-Patch querying (RP-DeFiNe) generates sharper edges in both RGB and depth images.

	DeFiNe	RP-DeFiNe
		k=16
↑ PSNR	23.46	**24.54**
↑ SSIM	0.783	**0.801**
↓ LPIPS	0.495	**0.453**
↓RMSE	0.275	**0.263**
↓Abs.Rel	0.108	**0.103**
↓Sq.Rel	0.053	**0.050**
↓ Giga FLOPs	801	**81**
↑ Rendering speed	7 fps	**208 fps**

Table 2. **Quantitative results on ScanNet.** Evaluation of stereo depth and RGB rendering on a realistic dataset. The integration of a Ray-Patch decoder with patch size $k = 16$ increases rendering speed by 2 orders of magnitude, while also outperforming DeFiNe's rendering and depth metrics.

ration reduces FLOPs $\times 10$, and increases rendering prediction of novel depth maps from 7 frames per second to 208.

6. Limitations

Our proposed decoder reduces the complexity problem of decoding images with Transformers. Despite that, we cannot decode single pixels and performance may depend on choosing an appropriate patch size. As a simple heuristic to choose the patch, we propose to keep $n_q \sim n_{kv}$, as it has been shown that 1) rendering speed saturates for bigger patches, and 2) too much compression reduces decoding performance. Nevertheless, hyper-parameter tuning may be needed to find the best patch size for each model. Regarding unsupervised segmentation, we observed that RP-OSRT has fallen into a tessellation failure mode, already observed by Sajjadi [40] *et al.* This failure is dependent on architectural choices, and further experimentation would be required to address it. Also note that we have only evaluated square patches. Nevertheless, our method could also be used with rectangular patches to obtain an intermediate number of queries. Finally, notice that the Ray-Patch does not attempt to solve the base attention's quadratic cost scaling. Rather, its focus on reducing the number of queries makes it compatible with other less expensive alternatives to vanilla attention [11, 12, 45, 53, 56].

7. Conclusion

In this paper we propose Ray-Patch querying, which reduces significantly the cost associated to Light Field Transformer's decoder. Our Ray-Patch does not only reduce significantly the training time and improve convergence, but could also be generalized to different tasks using transformers to decode scenes. We validate experimentally our approach and its benefits by integrating it into three recent LFT models for two different tasks and in two different datasets. The models with our Ray-Patch querying match or even outperform the baseline models in photometric and depth metrics, while at the same time reducing the computation and memory load in one and two orders of magnitude respectively. In addition, this is achieved with a minimum modification to the implementation of given baselines. Reducing the computational footprint of LFTs is essential for to continue its development and for deployment in constrained platforms such as mobile devices or robots, in the same line than works such as [31, 50] did for other architectures and tasks.

References

[1] Dejan Azinović, Ricardo Martin-Brualla, Dan B Goldman, Matthias Nießner, and Justus Thies. Neural rgb-d surface reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6290–6301, 2022. 1, 2

[2] Miguel Angel Bautista, Pengsheng Guo, Samira Abnar, Walter Talbott, Alexander Toshev, Zhuoyuan Chen, Laurent Dinh, Shuangfei Zhai, Hanlin Goh, Daniel Ulbricht, et al. Gaudi: A neural architect for immersive 3d scene generation. *arXiv preprint arXiv:2207.13751*, 2022. 1

[3] Michel Breyer, Jen Jen Chung, Lionel Ott, Roland Siegwart, and Juan Nieto. Volumetric grasping network: Real-time 6 dof grasp detection in clutter. In *Conference on Robot Learning*, pages 1602–1611. PMLR, 2021. 1

[4] Cesar Cadena, Luca Carlone, Henry Carrillo, Yasir Latif, Davide Scaramuzza, José Neira, Ian Reid, and John J Leonard. Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. *IEEE Transactions on robotics*, 32(6):1309–1332, 2016. 1

[5] Carlos Campos, Richard Elvira, Juan J Gómez Rodríguez, José MM Montiel, and Juan D Tardós. Orb-slam3: An accurate open-source library for visual, visual–inertial, and multimap slam. *IEEE Transactions on Robotics*, 37(6):1874–1890, 2021. 1

[6] Anh-Quan Cao and Raoul de Charette. Monoscene: Monocular 3d semantic scene completion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3991–4001, 2022. 1

[7] 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. 6

[8] Anpei Chen, Zexiang Xu, Fuqiang Zhao, Xiaoshuai Zhang, Fanbo Xiang, Jingyi Yu, and Hao Su. Mvsnerf: Fast generalizable radiance field reconstruction from multi-view stereo. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 14124–14133, 2021. 1, 2

[9] Zhiqin Chen and Hao Zhang. Learning implicit fields for generative shape modeling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5939–5948, 2019. 1

[10] Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5828–5839, 2017. 6

[11] Tri Dao. Flashattention-2: Faster attention with better parallelism and work partitioning, 2023. 1, 8

[12] Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in Neural Information Processing Systems*, 35:16344–16359, 2022. 1, 8

[13] Kangle Deng, Andrew Liu, Jun-Yan Zhu, and Deva Ramanan. Depth-supervised nerf: Fewer views and faster training for free. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12882–12891, 2022. 1

[14] Arnab Dey, Yassine Ahmine, and Andrew I Comport. Mipnerf rgb-d: Depth assisted fast neural radiance fields. *arXiv preprint arXiv:2205.09351*, 2022. 2

[15] 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*, 360 (6394):1204–1210, 2018. 1

[16] Patrick Esser, Robin Rombach, and Bjorn Ommer. Taming transformers for high-resolution image synthesis. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12873–12883, 2021. 7

[17] Jiatao Gu, Lingjie Liu, Peng Wang, and Christian Theobalt. Stylenerf: A style-based 3d-aware generator for high-resolution image synthesis. *arXiv preprint arXiv:2110.08985*, 2021. 1

[18] Vitor Guizilini, Igor Vasiljevic, Jiading Fang, Rares Ambrus, Greg Shakhnarovich, Matthew Walter, and Adrien Gaidon. Depth field networks for generalizable multi-view scene representation. In *European Conference on Computer Vision (ECCV)*, 2022. 1, 2, 3, 4, 6

[19] Andrew Jaegle, Sebastian Borgeaud, Jean-Baptiste Alayrac, Carl Doersch, Catalin Ionescu, David Ding, Skanda Koppula, Daniel Zoran, Andrew Brock, Evan Shelhamer, et al. Perceiver io: A general architecture for structured inputs & outputs. *arXiv preprint arXiv:2107.14795*, 2021. 2, 4

[20] Wonbong Jang and Lourdes Agapito. Codenerf: Disentangled neural radiance fields for object categories. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12949–12958, 2021. 1

[21] Abhijit Kundu, Kyle Genova, Xiaoqi Yin, Alireza Fathi, Caroline Pantofaru, Leonidas J Guibas, Andrea Tagliasacchi, Frank Dellaert, and Thomas Funkhouser. Panoptic neural fields: A semantic object-aware neural scene representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12871–12881, 2022. 1

[22] Uday Kusupati, Shuo Cheng, Rui Chen, and Hao Su. Normal assisted stereo depth estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2189–2199, 2020. 6

[23] Verica Lazova, Vladimir Guzov, Kyle Olszewski, Sergey Tulyakov, and Gerard Pons-Moll. Control-nerf: Editable feature volumes for scene rendering and manipulation. *arXiv preprint arXiv:2204.10850*, 2022. 1, 2

[24] Haotong Lin, Sida Peng, Zhen Xu, Yunzhi Yan, Qing Shuai, Hujun Bao, and Xiaowei Zhou. Efficient neural radiance fields for interactive free-viewpoint video. In *SIGGRAPH Asia 2022 Conference Papers*, pages 1–9, 2022. 1, 2

[25] Lingjie Liu, Jiatao Gu, Kyaw Zaw Lin, Tat-Seng Chua, and Christian Theobalt. Neural sparse voxel fields. *Advances in Neural Information Processing Systems*, 33:15651–15663, 2020. 1, 2

[26] Francesco Locatello, Dirk Weissenborn, Thomas Unterthiner, Aravindh Mahendran, Georg Heigold, Jakob

Uszkoreit, Alexey Dosovitskiy, and Thomas Kipf. Object-centric learning with slot attention. *Advances in Neural Information Processing Systems*, 33:11525–11538, 2020. 4

[27] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4460–4470, 2019. 1

[28] Mateusz Michalkiewicz, Jhony K Pontes, Dominic Jack, Mahsa Baktashmotlagh, and Anders Eriksson. Implicit surface representations as layers in neural networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4743–4752, 2019. 1

[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 *European conference on computer vision*, pages 405–421. Springer, 2020. 1, 2, 5

[30] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Trans. Graph.*, 41(4):102:1–102:15, 2022. 1, 2

[31] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Transactions on Graphics (ToG)*, 41(4):1–15, 2022. 8

[32] Thomas Neff, Pascal Stadlbauer, Mathias Parger, Andreas Kurz, Joerg H. Mueller, Chakravarty R. Alla Chaitanya, Anton S. Kaplanyan, and Markus Steinberger. DONeRF: Towards Real-Time Rendering of Compact Neural Radiance Fields using Depth Oracle Networks. *Computer Graphics Forum*, 2021. 2

[33] Michael Niemeyer and Andreas Geiger. Giraffe: Representing scenes as compositional generative neural feature fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11453–11464, 2021. 1, 5

[34] Helen Oleynikova, Zachary Taylor, Marius Fehr, Roland Siegwart, and Juan Nieto. Voxblox: Incremental 3d euclidean signed distance fields for on-board mav planning. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1366–1373. IEEE, 2017. 1

[35] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 165–174, 2019. 1

[36] Songyou Peng, Michael Niemeyer, Lars Mescheder, Marc Pollefeys, and Andreas Geiger. Convolutional occupancy networks. In *European Conference on Computer Vision*, pages 523–540. Springer, 2020. 1

[37] Barbara Roessle, Jonathan T Barron, Ben Mildenhall, Pratul P Srinivasan, and Matthias Nießner. Dense depth priors for neural radiance fields from sparse input views. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12892–12901, 2022. 1, 2

[38] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022. 7

[39] David M Rosen, Kevin J Doherty, Antonio Terán Espinoza, and John J Leonard. Advances in inference and representation for simultaneous localization and mapping. *Annual Review of Control, Robotics, and Autonomous Systems*, 4: 215–242, 2021. 1

[40] Mehdi SM Sajjadi, Daniel Duckworth, Aravindh Mahendran, Sjoerd van Steenkiste, Filip Pavetić, Mario Lučić, Leonidas J Guibas, Klaus Greff, and Thomas Kipf. Object scene representation transformer. *arXiv preprint arXiv:2206.06922*, 2022. 1, 2, 3, 4, 6, 8

[41] Mehdi SM Sajjadi, Henning Meyer, Etienne Pot, Urs Bergmann, Klaus Greff, Noha Radwan, Suhani Vora, Mario Lučić, Daniel Duckworth, Alexey Dosovitskiy, et al. Scene representation transformer: Geometry-free novel view synthesis through set-latent scene representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6229–6238, 2022. 1, 2, 3, 4, 6

[42] Vincent Sitzmann, Semon Rezchikov, Bill Freeman, Josh Tenenbaum, and Fredo Durand. Light field networks: Neural scene representations with single-evaluation rendering. *Advances in Neural Information Processing Systems*, 34: 19313–19325, 2021. 1, 3

[43] Karl Stelzner, Kristian Kersting, and Adam R Kosiorek. Decomposing 3d scenes into objects via unsupervised volume segmentation. *arXiv preprint arXiv:2104.01148*, 2021. 1, 6

[44] Edgar Sucar, Shikun Liu, Joseph Ortiz, and Andrew J Davison. imap: Implicit mapping and positioning in real-time. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6229–6238, 2021. 1

[45] Yi Tay, Mostafa Dehghani, Samira Abnar, Yikang Shen, Dara Bahri, Philip Pham, Jinfeng Rao, Liu Yang, Sebastian Ruder, and Donald Metzler. Long range arena: A benchmark for efficient transformers. In *International Conference on Learning Representations*, 2020. 1, 8

[46] Zachary Teed and Jia Deng. Droid-slam: Deep visual slam for monocular, stereo, and rgb-d cameras. *Advances in neural information processing systems*, 34:16558–16569, 2021. 1

[47] Ayush Tewari, Justus Thies, Ben Mildenhall, Pratul Srinivasan, Edgar Tretschk, W Yifan, Christoph Lassner, Vincent Sitzmann, Ricardo Martin-Brualla, Stephen Lombardi, et al. Advances in neural rendering. In *Computer Graphics Forum*, pages 703–735. Wiley Online Library, 2022. 1, 2

[48] Alex Trevithick and Bo Yang. Grf: Learning a general radiance field for 3d representation and rendering. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 15182–15192, 2021. 1, 2

[49] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017. 2, 3

[50] Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao. Yolov7: Trainable bag-of-freebies sets

new state-of-the-art for real-time object detectors. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2023. 8

[51] Qianqian Wang, Zhicheng Wang, Kyle Genova, Pratul P Srinivasan, Howard Zhou, Jonathan T Barron, Ricardo Martin-Brualla, Noah Snavely, and Thomas Funkhouser. Ibrnet: Learning multi-view image-based rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4690–4699, 2021. 1, 2

[52] Yi Wei, Shaohui Liu, Yongming Rao, Wang Zhao, Jiwen Lu, and Jie Zhou. Nerfingmvs: Guided optimization of neural radiance fields for indoor multi-view stereo. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5610–5619, 2021. 1, 2

[53] Yunyang Xiong, Zhanpeng Zeng, Rudrasis Chakraborty, Mingxing Tan, Glenn Fung, Yin Li, and Vikas Singh. Nyströmformer: A nyström-based algorithm for approximating self-attention. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 14138–14148, 2021. 1, 8

[54] Bangbang Yang, Yinda Zhang, Yinghao Xu, Yijin Li, Han Zhou, Hujun Bao, Guofeng Zhang, and Zhaopeng Cui. Learning object-compositional neural radiance field for editable scene rendering. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 13779–13788, 2021. 1

[55] Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. pixelnerf: Neural radiance fields from one or few images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4578–4587, 2021. 1, 2

[56] Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, et al. Big bird: Transformers for longer sequences. *Advances in neural information processing systems*, 33:17283–17297, 2020. 1, 8

[57] Shuaifeng Zhi, Tristan Laidlow, Stefan Leutenegger, and Andrew J Davison. In-place scene labelling and understanding with implicit scene representation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15838–15847, 2021. 1

[58] Zihan Zhu, Songyou Peng, Viktor Larsson, Weiwei Xu, Hujun Bao, Zhaopeng Cui, Martin R Oswald, and Marc Pollefeys. Nice-slam: Neural implicit scalable encoding for slam. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12786–12796, 2022. 1, 2