TriNeRFLet: A Wavelet Based Triplane NeRF Representation

Rajaei Khatib and Raja Giryes

Tel Aviv University
rajaeikhatib@mail.tau.ac.il and raja@tauex.tau.ac.il

Abstract. In recent years, the neural radiance field (NeRF) model has gained popularity due to its ability to recover complex 3D scenes. Following its success, many approaches proposed different NeRF representations in order to further improve both runtime and performance. One such example is Triplane, in which NeRF is represented using three 2D feature planes. This enables easily using existing 2D neural networks in this framework, e.g., to generate the three planes. Despite its advantage, the triplane representation lagged behind in 3D recovery quality compared to NeRF solutions. In this work, we propose the TriNeRFLet framework, where we learn the wavelet representation of the triplane and regularize it. This approach has multiple advantages: (i) it allows information sharing across scales and regularization of high frequencies; (ii) it facilitates performing learning in a multi-scale fashion; and (iii) it provides a 'natural' framework for performing NeRF super-resolution (SR), such that the low-resolution wavelet coefficients are computed from the provided low-resolution multi-view images and the high frequencies are acquired under the guidance of a pre-trained 2D diffusion model. We show the SR approach's advantage on both Blender and LLFF datasets.

Keywords: Neural Radiance Fields (NeRF) · Wavelet · Multiscale representation · 3D Super-Resolution · Diffusion Models

1 Introduction

3D scene reconstruction from multiple 2D views is a challenging task that has been widely studied, and many methods have been proposed to solve it. Neural radiance field (NeRF) [26] is prominent among these methods as it has demonstrated a high level of generalizability in generating novel views with high quality and consistent lighting. NeRF utilizes an implicit representation of the 3D scene in the form of a multi-layer perceptron (MLP). This enables it to capture complex 3D geometry and lighting.

At a high level, NeRF relies on the rendering equation that approximates each pixel in the image using points sampled along the ray that passes through it [26]. The MLP in NeRF takes as input the frequency encoding of the Euclidean coordinates and view direction of the point of interest, and outputs the radiance and density at this point. The MLP weights are learned via end-to-end

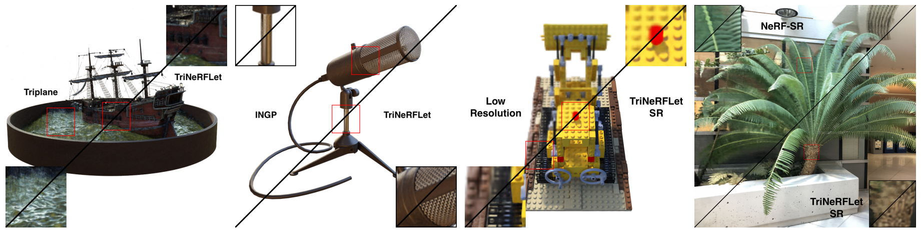


Fig. 1: Our approach improves the quality of NeRF reconstruction (zoom-in). From left to right: TriNeRFLet improvement over Triplane, TriNeRFLet compared to INGP, TriNeRFLet SR improvement, TriNeRFLet SR compared to NeRF-SR.

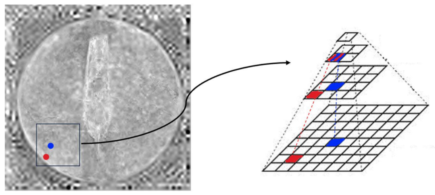
optimization, by comparing the values between the rendered pixel and its value in the corresponding 2D image. After the training process is over, the MLP can be used to render novel views.

Due to its success in representing 3D scenes, many methods proposed improvements to NeRF [5,6,27]. They sought to enhance NeRF 3D reconstruction capabilities, as well as other drawbacks such as high runtime and aliasing artifacts that it suffered from. One such approach uses three axis-aligned 2D feature planes, denoted as Triplane, to represent the NeRF [7]. In the rendering process, each point is sampled by projecting it onto each of the three planes and then concatenating the features that correspond to the three projections. This forms a single feature vector for the point that is then passed to a small MLP that outputs the density and color values of this point.

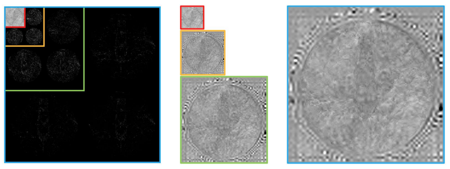
A significant advantage of the Triplane representation is that it can be used with many already existing 2D methods. In the original work [7], the authors used an existing 2D Generating Adversarial Network (GAN) architecture to generate its planes. Follow-up works employed the 2D property of the Triplane to perform NeRF super-resolution [4] and 3D generation [20,36].

While being useful due to its special 2D structure, the reconstruction quality achieved by Triplane lagged behind other efficient multiview reconstruction methods such as instant NGP (INGP) [27], which is an improvement of NeRF, and 3D Gaussian splatting [19]. Due to its structure, only Triplane entries that are included in the training views rays are learned. Thus, there might be entries at the planes that will still have their initial random values also after training finishes. These random features are used by novel views that rely on them and therefore they deteriorate the quality of their generation and create artifacts in these novel views.

To tackle these drawbacks, we present a novel Triplane representation that relies on a multiscale 2D wavelet structure. Instead of learning the feature planes directly, we learn the wavelet features of several resolutions, while regularizing the wavelet representation to be sparse. Thus, regions that are covered by the training views are learned in a similar way to the case of a regular Triplane, and regions that are not covered by the training views are updated by a lower resolution estimate according to how much they are affected by their neighboring regions. Figure 1(left) shows the improvement in reconstruction quality attained



(a) The multiscale wavelet representation transfers information between points on the triplane planes: Although the red dot on the plane was not covered by the training images views, it will get a coarser estimate from a lower resolution wavelet layer that corresponds also to the blue dot, which was updated in training.



(b) Example of the multiscale property of the wavelet representation. On the left, we present the wavelet representation. On the right, we show the feature plane of the wavelet representation on the left at different resolutions. Note that the wavelet representation of a higher resolution includes the wavelet representations of the lower-resolution planes (the colors correspond to the matching between each plane and its wavelet representation).

Fig. 2: Illustartion of the multiscale property of the wavelet representation.

by this representation. Figure 2a illustrates how regions on the plane may affect each other due to the wavelet representation.

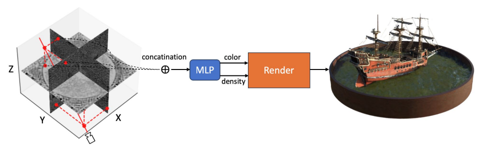
In our optimization, we use L_1 regularization on the wavelet coefficients. The goal is to sparsify the coefficients that are not trained by the given views such that only the coefficients that were trained will impact the reconstruction also from other views. Using L_1 regularization is inspired by the wavelet literature, which indicates that wavelet features can represent data accurately while being sparse [11].

To further improve the training, we perform it in a multi-scale fashion, starting with a lower resolution version that updates only the coarse coefficients of the wavelet. Then, after some iterations, we increase the resolution and add additional wavelet layers to the representation.

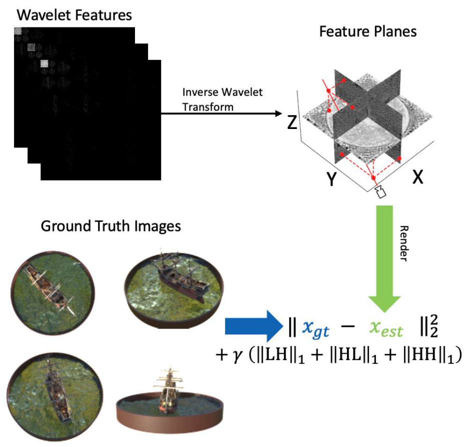
We apply the proposed TriNeRFLet framework to two tasks. First, for 3D reconstruction, where it closes the performance gap of Triplane against other multiview 3D reconstruction methods and makes it competitive with current state-of-the-art (SOTA) methods. Second, we combine it with a pre-trained 2D super-resolution (SR) diffusion models [32, 33] to perform NeRF SR. Figure 1(right) demonstrates the improvement this approach attains. We show in various experiments the advantage of our TriNeRFLet in novel view reconstruction compared to INGP and Triplane and the superiority of our SR solution compared to competing approaches that do not require dedicated training for multiview or 3D data.

2 Related Works and Background

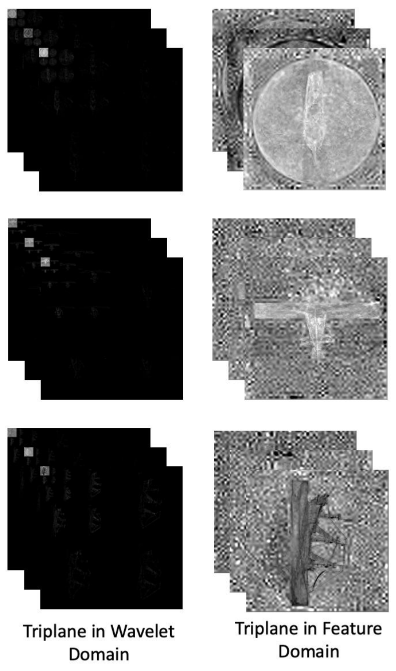
NeRF [26] is a framework for reconstructing a 3D scene from a set of multiview images. Its main component is an implicit neural representation of the 3D scene via a mapping neural network $F_\theta : (\boldsymbol{x}, \boldsymbol{d}) \to (\boldsymbol{c}, \sigma)$, where $\boldsymbol{x}$ is the Cartesian coordinate of the point of interest, $\boldsymbol{d}$ is the direction vector from it to the camera origin, $\boldsymbol{c}$ is the RGB color at this point and σ is the density.



(a) *Triplane rendering.* The feature vector of each point is sampled by gathering the projected features on the Triplane planes. This vector is passed to an MLP to obtain color and density at this point. The final value of a pixel is calculated by using the color and density of the points that reside on the ray that passes through it.


Wavelet Features
Inverse Wavelet Transform
Feature Planes
Z
Y
X
Render
Ground Truth Images
$\| x_{gt} - x_{est} \|_2^2 + \gamma\,(\|\mathrm{LH}\|_1 + \|\mathrm{HL}\|_1 + \|\mathrm{HH}\|_1)$

(b) *3D reconstruction training scheme.* First, wavelet features are transformed into the Triplane domain. Next, pixels are rendered using these features in order to fit them to their ground-truth values. The high frequencies channels of LH, HL and HH from all wavelet levels get regularized by L_1 loss.



(c) In the wavelet domain, the low-frequency coefficients $\mathrm{LL}_{N_{base}}$ are observed at the top left corner (the bright spot). The rest are the high frequencies channels $\mathrm{LH}_{N_{base}}$, $\mathrm{HL}_{N_{base}}$ and $\mathrm{HH}_{N_{base}}$. All the other LH, HL and HH channels can be observed accordingly. All these planes belong to the same 3D object.

Fig. 3: *The TriNeRFLet reconstruction framework* learns features in wavelet domain, which are transformed into feature domain to render 3D objects.

The network F is trained using the pixels of the multiview images using the rendering equation. For a given pixel, it relates its value to N points sampled along the ray that passes through this pixel, where the ray direction is derived from the image view. Denoting by $t_1 < t_2 < ... < t_N$ the depth of the points, $\delta_i = t_{i+1} - t_i$ the distance between adjacent samples, and $\sigma_i, \boldsymbol{c}_i$ the density and color of each point respectively, then the pixel value $\hat{C}$ can be estimated from these points via:

$$\hat{\boldsymbol{C}} = \sum_{i=1}^{N} T_i (1 - \exp(-\sigma_i \delta_i)) \boldsymbol{c}_i, \quad \text{where} \quad T_i = \exp\left(- \sum_{j=1}^{i-1} \sigma_j \delta_j \right). \tag{1}$$

This equation is a discrete approximation of the rendering equation. As we have the true pixel value $\boldsymbol{C}$ of the image, the network parameters θ are trained simply by minimizing the L2 loss $\left\| \boldsymbol{C} - \hat{\boldsymbol{C}} \right\|_2$.

In the classical NeRF scheme [26], the mapping function F_θ is divided into two main parts. The first part is a frequency encoding function that encodes $\boldsymbol{x}$ and $\boldsymbol{d}$ using Fourier features [37]. These vectors are then concatenated to form a feature vector that is fed to the second part, which is an MLP function that maps the feature vector into its corresponding color and density values. Some extensions to NeRF consider optimization using unconstrained photo collections [24], few view points [42], non-rigidly deforming scenes [28], and imperfect camera poses [21].

INGP [27] is an important development, which managed to decrease NeRF runtime significantly while getting even better reconstruction performance. It achieved that by using a multiscale hash grid structure alongside a small shallow MLP that represents the scene instead of the big deep one used in previous works. Furthermore, this method introduced new rendering techniques and low-level optimizations in order to get a faster runtime. The multiscale concept of INGP will also be used in our proposed TriNeRFLet framework.

3D Gaussian Splatting [19] is a state-of-the-art 3D reconstruction approach that takes a different strategy than NeRF. Instead of using an implicit neural function, it represents the scene as a group of 3D Gaussians, and learns their parameters. It introduces a new method for efficiently rendering these Gaussians, thus, managing to train rapidly and achieving 3D rendering in real-time. Note that one needs to distinguish between training and rendering time. While its training time is similar to INGP, its rendering is faster than NeRF solutions.

Triplane [4, 7] is a NeRF variant, in which the frequency encoding part is replaced by three perpendicular 2D feature planes of dimension $N \times N$ with C channels. To get the feature vector of a point $\boldsymbol{x}$, the point is projected onto each of these planes, then the sampled feature values of the projected points are gathered together to form a feature vector. The final feature vector is obtained by concatenating this feature vector with the frequency encoding of the direction vector $\boldsymbol{d}$. These feature planes are learned alongside the MLP weights with a similar approach to what was explained earlier. Fig. 3a illustrates this process.

The advantage of Triplane compared to other alternatives is its use of 2D planes, which eases the use of standard 2D networks with it. This has been used in various applications. Bahat et al. [4] utilized it to train a NeRF super-resolution (SR) network from multiview training data. By exploiting the triplane structure, a 2D SR neural network was trained to perform SR on low resolution planes and in this way increase the overall NeRF resolution. Another example is [36], in which a diffusion model was trained to generate a Triplane. It utilizes the generalization power of diffusion models and brings them to 3D generation. Similarly, Triplane is predicted by a transformer-based neural network in [20] in order to have fast text to 3D generation.

Wavelets were widely used in signal and image processing applications [23]. With the advent of deep learning, they were employed in various architectures to improve performance and efficiency. The filter-bank approach of wavelet was utilized to represent each kernel in a neural network as a combination of filters from a given basis, e.g., a wavelet basis [30]. The concept of using multi-resolution was used to improve positional encoding of implicit representations [14, 22, 35].

Wavelets were used to improve efficiency and generation of StyleGAN [10], diffusion models [12, 16, 29] and normalizing flows [43].

Rho *et al.* [31] implemented a multiscale 3D grid using a grid decomposition method that transforms 2D planes alongside 1D vectors (unrelated to the 2D planes) into a 3D grid, where they used a 2D multiscale wavelet representation to represent the 2D planes. Despite the use of 2D multiscale representation, this method is different in essence from TriNeRFLet, which deploys the 2D wavelet multiscale representation on Triplane planes directly, thus being freed from 3D grid limitations.

Diffusion Models [9,15] have become a very popular tool in recent years for image manipulation and reconstruction tasks [17, 18, 32, 34, 40], where a denoising network is trained to learn the prior distribution of the data. Diffusing models can be used also beyond generation, where given a degraded image, some conditioning mechanism is combined with the learned prior to solve different tasks [2, 3, 8]. For example, in [1, 34, 40] diffusion models were utilized for the problems of deblurring and SR. In this work, we use the Stable Diffusion (SD) upscaler [32] to perform SR on 2D projections of the low-resolution NeRF in order to create a high-resolution version of the NeRF. While the SD upscaler is conditioned both on the low-resolution image and a given text prompt, we use it without text, as explained later in the paper.

The authors of [41] proposed to fine-tune a diffusion model on the views of a given scene to regularize unseen views. In a similar spirit, we use images generated by a diffusion model to fit the scene and thus improve NeRF SR.

3 TriNeRFLet Framework

To alleviate the Triplane drawbacks mentioned above, we present TriNeRFLet. Instead of using the 2D feature planes directly, we optimize them in the wavelet domain. At a high level, wavelet transforms a 2D plane to a multiscale representation that contains both low and high frequencies at different resolutions. The inverse wavelet transform, convert this representation back to the original 2D plane. Note that we apply the wavelet separably on each channel. We use the following notation for the wavelet representation. At resolution i of the wavelet representation, we denote by LL_i the low frequencies of this resolution, and by LH_i, HL_i and HH_i the high frequencies. Note that LL_i is used to create the coefficients of the next resolution of the wavelet representation. At a certain step, we stop and remain with LL.

In rendering, TriNeRFLet is similar to regular Triplane, in the sense that given a point it calculate its features by projecting it to the Triplane and then feed the features to a MLP that outputs the color c_i and density σ_i of the point that are then used by the rendering equation (see Figure 3a). Yet, instead of holding the Triplane in the feature domain, TriNeRFLet holds it in the Wavelet domain and optimizes the wavelet coefficients. Note that in the regular Triplane scheme, each plane entry impacts only one location in each plane. In TriNeRFLet, it impacts different resolutions in the wavelent representation of the plane (see

Figure 2a). Figure 3c shows some learned planes for the ship object and their corresponding Wavelet coefficients.

The learned parameters of TriNeRFLet, alongside the MLP weights, are the wavelet coefficients, and they are learned in an end-to-end fashion. The multiscale structure of the wavelet, allows the method to learn fine details whenever it needs, and to have a lower resolution estimate from lower resolution wavelet layers that are affected by nearby values when the area is not covered by the training pixels. The question remains, what should be done with the coefficients of the higher frequencies that are not updated.

Wavelet regularization. From wavelet theory [11], we expect the LH, HL and HH channels to be sparse. Thus, we regularize these channels to be sparse over all resolutions by adding an L_1 regularization for these coefficients to the training loss. This allows us to increase the resolution of the planes without having the problem of having features in it that remain random or redundant. In TriNeRFLet, also these features get updated.

To further comprehend the significance of the multiscale wavelet structure and the regularization we use for it, we compare the behavior to rendering novel views with a regular Triplane. In a regular Triplane, some features will still have their initial random values after training ends. This introduces artifacts when rendering novel views, as demonstrated in the ship example in Figure 1. Note that these artifacts are absent in TriNeRFLet because all the features in the planes are updated either by direct training or using a coarser estimate from lower resolution wavelet coefficients (see Figure 2a).

Multiscale training. A key advantage of TriNeRFLet is that the multiscale property of the wavelet representation enables increasing the resolution of the TriNeRFLet during training seamlessly. It is possible to continue to train the higher-resolution TriNeRFLet while using the wavelet coefficients of the lower-resolution TriNeRFLet. The wavelet of the lower-resolution plane is a subset of the wavelet of the higher resolution plane (see Figure 2b). We use this flexibility to improve the training by learning the wavelet planes in a coarse to fine manner. As we explain later, we leverage the multiscale structure of the wavelet both for NeRF reconstruction and SR.

The multiscale wavelet structure is not ideal in terms of training runtime due to the fact that the feature planes need to be reconstructed in each training step. Yet, its rendering runtime is quite fast since the feature planes are reconstructed just once and then the runtime is competitive with other well-known fast NeRF schemes like INGP [27].

4 TriNeRFLet Applications

4.1 TriNeRFLet Reconstruction

The TriNeRFLet framework is straightforwardly applied to 3D scene reconstruction task. Given the training pixels, wavelet features are optimized end-to-end on the pixel reconstruction loss in addition to the wavelet high frequencies regularization (see Figure 3b). The overall loss is:

$$\mathcal{L} = \sum_{i \in TrainingPixels} \left\| \boldsymbol{C}_i - \hat{\boldsymbol{C}}_i \right\|_2^2 + \gamma \sum_{l \in resolutions} \left(\|\mathrm{LH}_l\|_1 + \|\mathrm{HL}_l\|_1 + \|\mathrm{HH}_l\|_1 \right),$$

where γ is the L_1 regularization factor.

As mentioned before, the flexible multiscale structure of TriNeRFLet enables the training to be in a coarse-to-fine manner. This mitigates the runtime overhead of the wavelet inverse transform. At the beginning of the training, the method usually learns high-level features, and then when it starts to converge it learns the low-level details, which is coherent with the coarse-to-fine approach.

4.2 TriNeRFLet Super-Resolution (SR)

We turn to present a novel NeRF SR approach that relies on the proposed multiscale structure. This method employs a 2D pre-trained stable diffusion (SD) upscaler [32] for SR "guidance". Our approach does not require any kind of low-high resolution 3D scene pairs for supervision as done in other works [4]. Utilizing the robustness of the SD upscaler, our approach has the potential to handle different types of scenes and a range of resolutions.

Given a set of low-resolution (LR) multiview images of a given scene, the goal is to generate high-resolution (HR) views of this scene. We first apply TriNeRFLet using the LR images, providing a LR 3D reconstruction. In this reconstruction, we learn only the low frequencies of the wavelet representation. Denote the size of the LR planes by N_{LR} and the number of wavelet levels by L_{LR}. For the HR planes, denote the size by N and the number of levels by L. Due to the multiscale nature wavelet representations, in the HR TriNeRFLet we just need to learn the high frequencies of the wavelet, i.e., the first L_{LR} levels of the HR wavelet planes are shared with the LR TriNeRFLet reconstruction.

Figure 4a illustrates our TriNeRFLetSR strategy. First, we start by reconstructng the LR TriNeRFLet using the provided low-resolution multiview images. After training for s_{LR} steps, the super-resolution process starts, and a key component of it is the stable diffusion upscaler refinement ($\mathrm{SD}_{\mathrm{refine}}$) step presented in Figure 4b. Given a low-resolution ground-truth image $\boldsymbol{x}_{LR}^{gt}$, its high-resolution version $\boldsymbol{x}_{HR}^{est}$ is rendered using all wavelet levels L. Initially, the result will not be of high-resolution. Thus, to improve it we use a diffusion step with time t randomly selected from the range $[T_{min}\ ,\ T_{max}]$. Then, similar to [13], a noise with variance that depends on t is added to $\boldsymbol{x}_{HR}^{est}$. Then, we plug the noisy version of $\boldsymbol{x}_{HR}^{est}$ into the SD upscaler to perform the diffusion process from step t until the end, while being conditioned on the LR image $\boldsymbol{x}_{LR}^{gt}$. This results in the enhanced version of $\boldsymbol{x}_{HR}^{est}$, denoted as $\boldsymbol{x}_{HR}^{enhanced}$. We denote this enhancement process by $\mathrm{SD}_{\mathrm{refine}}$.

We use the SD upscaler with a small modification compared to the original work [32]. In their work, the denoising step is

$$\boldsymbol{x}_{t-1} = \epsilon(\boldsymbol{x}_t, \boldsymbol{z}, \varnothing) + \alpha \left(\epsilon(\boldsymbol{x}_t, \boldsymbol{z}, \boldsymbol{y}) - \epsilon(\boldsymbol{x}_t, \boldsymbol{z}, \varnothing) \right),$$

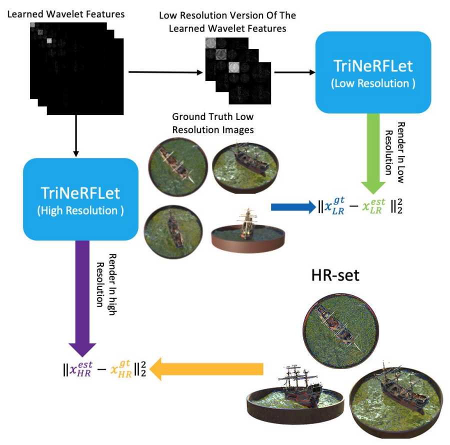
Algorithm 1 TriNeRFLet Super-Resolution (SR)

```
Input: Ground truth low resolution (LR) images $\{\boldsymbol{x}_{i}{}^{gt}_{LR}\}_{i=1}^{k}$, LR only steps $s_{LR}$, Total steps $s$, Wavelet LR levels $L_{LR}$, Refresh steps $s_{refresh}$, Diffusion step limits $T_{min}$ , $T_{max}$.
Output: High-Resolution (HR) TriNeRFLet
 1: HR-set ← {}                                              ▷ empty dictionary
 2: for $itr = 0 \cdots s-1$ do
 3:     Choose randomly a frame i - $\boldsymbol{x}_{i}{}^{gt}_{LR}$.
 4:     Render $\boldsymbol{x}_{i}{}^{est}_{LR}$ using only first $L_{LR}$ wavelet levels.
 5:     $loss \leftarrow \left\|\boldsymbol{x}_{i}{}^{est}_{LR} - \boldsymbol{x}_{i}{}^{gt}_{LR}\right\|_2^2$
 6:     if $itr \geq s_{LR}$ then
 7:         Render $\boldsymbol{x}_{i}{}^{est}_{HR}$ using all wavelet levels.
 8:
 9:         if $itr \% s_{refresh} == 0$ then
10:             HR-set ← {}                                  ▷ empty dictionary
11:         end if
12:
13:         if $i \in$ HR-set then
14:             $\boldsymbol{x}_{i}{}^{gt}_{HR} \leftarrow$ HR-set[$i$]
15:         else
16:             $t \sim Rand(T_{min}\ ,\ T_{max})$
17:             $\boldsymbol{x}_{i}{}^{gt}_{HR} \leftarrow \mathrm{SD}_{\mathrm{refine}}(\boldsymbol{x}_{i}{}^{est}_{HR}, \boldsymbol{x}_{i}{}^{gt}_{LR}, t)$
18:             HR-set[$i$] $\leftarrow \boldsymbol{x}_{i}{}^{gt}_{HR}$
19:         end if
20:         $loss \leftarrow loss + \left\|\boldsymbol{x}_{i}{}^{est}_{HR} - \boldsymbol{x}_{i}{}^{gt}_{HR}\right\|_1 + \lambda\ \mathrm{LPIPS}(\boldsymbol{x}_{i}{}^{est}_{HR}, \boldsymbol{x}_{i}{}^{gt}_{LR})$
21:     end if
22:     $loss \leftarrow loss + \gamma\ \sum_l (\|\mathrm{LH}_l\|_1 + \|\mathrm{HL}_l\|_1 + \|\mathrm{HH}_l\|_1)$
23:     $T_{max} \leftarrow scheduler(T_{max}, i)$                    ▷ decrease $T_{max}$
24: end for
```

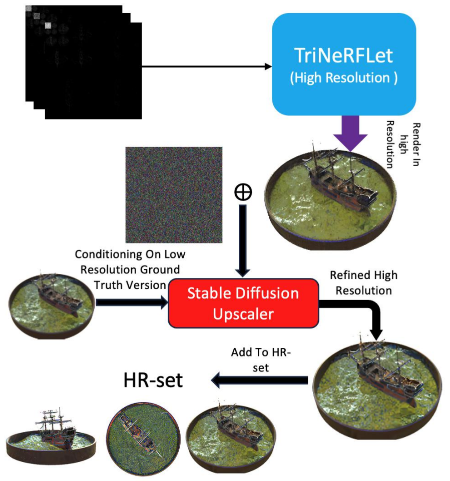
where ϵ, α, $\boldsymbol{z}$ and $\boldsymbol{y}$ are the U-net denoiser, a guidance factor, the low-resolution image and the guidance text respectively. Instead of doing the guidance in the text direction, we do it in the low-resolution image direction and without text, i.e., $\boldsymbol{x}_{t-1} = \epsilon(\boldsymbol{x}_t, \varnothing, \varnothing) + \alpha\left(\epsilon(\boldsymbol{x}_t, \boldsymbol{z}, \varnothing) - \epsilon(\boldsymbol{x}_t, \varnothing, \varnothing)\right)$.

Having the refined HR images, we want to use them to update the TriNeRFLet. We add these images to the set denoted HR-set. This set is initialized to be empty and gets updated sequentially. It is refreshed (i.e. emptied) every $s_{refresh}$ steps as illustrated in Algorithm 1. The images in HR-set are used as "ground-truth" images for training the HR TriNeRFLet to update all its L levels.

To tie all threads together, the method starts by fitting TriNeRFLet with low-resolution features to the low-resolution ground-truth images. Then $\mathrm{SD}_{\mathrm{refine}}$ is used to improve the resolution of the TriNeRFLet generated views and the images are stored in HR-set. Next, a TriNeRFLet with high-resolution features (using all wavelet levels) is fitted to HR-set and simultaneously the low-resolution images are used to train the low levels of the wavelet representation of the same TriNeRFLet. As in regular TriNeRFLet training, a L_1 regularization of


Learned Wavelet Features
Low Resolution Version Of The Learned Wavelet Features
TriNeRFLet (Low Resolution)
Ground Truth Low Resolution Images
Render In Low Resolution
$\|x_{LR}^{gt} - x_{LR}^{est}\|_2^2$
TriNeRFLet (High Resolution)
Render In high Resolution
HR-set
$\|x_{HR}^{est} - x_{HR}^{gt}\|_2^2$

(a) *TriNeRFLet based super-resolution.* Low-resolution TriNeRFLet renders LR images using the LR version of the wavelet features, and fits them to the given low-resolution images. High-resolution TriNeRFLet renders HR images using all wavelet levels, and fits them to HR images from HR-set.


TriNeRFLet (High Resolution)
Render In high Resolution
⊕
Conditioning On Low Resolution Ground Truth Version
Stable Diffusion Upscaler
Refined High Resolution
Add To HR-set
HR-set

(b) *SD_{refine} process.* First, a HR image is rendered. To improve its quality, noise is added to it and then it is plugged into Stable Diffusion upscaler with its LR version for conditioning. The result, which is a refined HR image is added to the HR-set.

Fig. 4: TriNeRFLet Super-Resolution (SR) components.

the wavelet coefficients is added to the loss. In addition, we use also a LPIPS loss between the rendered high resolution and the ground truth low resolution. As mentioned earlier, HR-set gets refreshed every $s_{refresh}$ training steps. Furthermore, the diffusion step upper limit T_{max} is scheduled to linearly decrease during training in order to force $\text{SD}_{\text{refine}}$ to introduce fewer changes as training converges. This is described in Algorithm 1.

The TriNeRFLet multiscale wavelet structure plays an important role in the SR scheme. First, its multiscale structure establishes a connection between high-resolution and low-resolution renderings by forcing them to share information (using the same first L_{LR} wavelet levels). Second, wavelet regularization, especially in the layers that only the HR TriNeRFLet uses, constrains the optimization process and forces the HR TriNeRFLet to learn only useful wavelet coefficients that contribute to HR details. Thus, the combination of multiscale wavelet and diffusion models allows the method to learn the high-resolution details it exactly needs without redundancy.

The diffusion SR model that we use supports SR from 128 to 512. Yet, we want to support other resolutions, such as 100 to 400 or 200 to 800. For lower resolutions, e.g., going from 100 to 400, we pad $\boldsymbol{x}_{LR}^{gt}$ and $\boldsymbol{x}_{HR}^{est}$ with zeros in order to achieve the desired resolution before plugging them into the SD upscaler. When it finishes, the result is cropped to the original resolution. For higher resolutions, e.g., going from 200 to 800, we randomly crop 128 and 512 resolution crops from $\boldsymbol{x}_{LR}^{gt}$ and $\boldsymbol{x}_{HR}^{est}$, respectively, that correspond to the same relative location. We plug the images into the upscaler, and the result is actually an enhanced version of the crop from $\boldsymbol{x}_{HR}^{est}$. Then, we only fit the HR render to

Method	Mic	Chair	Ship	Materials	Lego	Drums	Ficus	Hotdog	Avg.	Train Time ↓	Render FPS ↑
NeRF	32.91	33.00	28.65	29.62	32.54	25.01	30.13	36.18	31.005	≥ 1 day	≤ 0.2
INGP (paper)	36.22	35.00	31.10	29.18	36.39	26.02	33.51	37.40	33.176	5 mins	1
INGP (rerun)	36.22	34.81	31.1	29.68	36.03	25.15	32.41	37	32.8	5 mins	1
INGP (rerun)*	36.3	35.18	31.22	29.8	36.06	24.96	33.22	37.1	32.98	6hours	1
3D Gaussian Splatting	35.36	<u>35.83</u>	30.80	<u>30.00</u>	35.78	<u>26.15</u>	**34.87**	<u>37.72</u>	33.32	10 mins	135
Triplane	33.85	32.83	29.58	28.15	34.70	24.86	30.35	35.80	31.26	2 hours	1.5
Triplane*	34.4	33.5	29.97	28.1	35.4	24.9	30.68	36.16	31.64	6 hours	1.5
Ours: TriNeRFLet Small	35.18	33.76	30.33	29.14	35.32	25.66	33.34	36.24	32.37	15 mins	1.5
Ours: TriNeRFLet Base Light	35.77	35.00	31.10	29.35	36.44	25.98	33.96	36.93	33.07	1.5 hours	1.5
Ours: TriNeRFLet Base	<u>36.72</u>	35.54	<u>31.77</u>	29.72	<u>36.66</u>	26	33.7	37.31	<u>33.43</u>	6 hours	1.5
Ours: TriNeRFLet Large	**37.22**	**36**	**32.7**	**30.34**	**37.32**	**26.2**	<u>34.47</u>	**37.89**	**34.017**	∼ 1 day	1.5

Table 1: Blender Dataset PSNR (↑) results with train time and render FPS for each method. **Bold** is best, <u>underline</u> is second. For fair comparison, we also show INGP (rerun)* and Triplane*, which are trained for 6 hours as our TriNeRFLet base.

this crop instead of the whole image. When the HR-set is refreshed, a different random crop will be chosen, thus covering more areas as the process continues.

5 Experiments

We turn to present the results of our method. More visual results and ablations appear in the supplementary material. Code is available in the github repo.

5.1 3D Scene Reconstruction

For evaluating TriNeRFLet in the 3D recovery task, we define four versions of TriNeRFLet - small, base light, base and large. The parameters of TriNeRFLet are wavelet LL component resolution - N_{LL}, base resolution - N_{base}, wavelet layers - L, final resolution - N_{final}, number of channels - C, wavelet regularization γ, MLP dim (neurons) - W, MLP hidden layers - $D_{density}$ and D_{color} and training steps. These parameters are provided for each model in the Supplementary Materials. In the small, base light and base versions, TriNeRFLet uses the exact same MLPs as in INGP. TriNerfLet is trained using the Adam optimizer with a learning rate of 0.01 and an exponential decay scheduler. The wavelet low frequencies in LL (of size $N_{LL} \times N_{LL}$) are randomly initialized, and the other frequencies in LH, HL and HH at all levels are initialized with zeros. We use the Biorthogonal 6.8 (Bior6.8) wavelet type. Ablation for other wavelet types is found in the sup. mat. The other training details are similar to INGP. Our implementation uses [38], which is a Pytorch-based implementation of INGP.

Since the wavelet reconstruction component affects the runtime of TriNeRFLet, we start the training with lower resolutions of the planes, training coarse to fine. The initial plane resolution is set to N_{base} and it is increased throughout the training till it reaches the size N_{final} that has L wavelet levels. As described above, due to the wavelet multiscale structure, when we move to the next resolution in training we simply add another level (of higher frequencies) to the wavelet representation and add it to the training, where the lower levels are kept as is (but continue to train). As mentioned in Sec. 3, the relatively high training runtime is one of TriNeRFLet framework disadvantages, and this is due to the inverse wavelet transform overhead during optimization. Thus, doing the optimization in this coarse to fine manner mitigates this drawback and accelerates the training (inverse wavelet on smaller planes is faster).

To evaluate the performance of our method, we use the Blender dataset [26], which contains 8 synthetic 3D scenes with complex geometry and lighting. We compare TriNeRFLet with the vanilla NeRF [26], regular Triplane (with base config), INGP [27] and 3D Gaussian Splatting [19]. Results are reported in Table 1. For INGP we report 3 versions: INGP (paper) denotes the officially published results by [27], and INGP (rerun) and INGP (rerun)* denote the results that we obtained by running INGP (using [27] software) for 5 mins and 6 hours respectively. Also, Triplane and Triplane* denote Triplane results when trained for 5 mins and 6 hours respectively. We ran all TriNeRFLet versions on a single A6000 GPU. The reported results indeed demonstrate the improvement in performance that the multiscale wavelet brings, as it outperforms regular Triplane by a significant margin, and improves over the current SOTA, even in case they were trained for a longer time.

Regarding rendering time, the TriNeRFLet (all versions) frames per second (FPS) are approximately 1.5, while INGP and 3D Gaussian Splatting are approximately 1 and 135 respectively. These FPS are tested on a single A6000 GPU. Note that TriNeRFLet rendering FPS is compatible and even slightly better than INGP, which is known to be fast within the NeRF approaches. 3D Gaussian Splatting, which is not a NeRF method, is faster in its rendering time since it uses a different internal structure than NeRF. When compared only to NeRF-based solutions, our method is very fast with competitive reconstruction performance. While our rendering time is faster, the training time of TriNeRFLet is higher than INGP. This is due to the wavelet reconstruction, which needs to be done at each training step but only once during rendering time.

5.2 3D Scene Super-Resolution

Blender. We turn to present our results for TriNeRFLet SR. The objective here is to generate novel views with $\times 4$ resolution than the input images. To this end, we tested two different settings on the Blender dataset [26], 100 to 400 and 200 to 800. For the first one, we use $N_{LR} = 256$ and $N = 1024$, while in the second setting $N_{LR} = 512$ and $N = 2048$. Further technical details appear in the Supplementary Materials.

We compare our method with NeRF*, NeRF-Bi, NeRF-Liif, NeRF-Swin, NeRF-SR (SS) (these methods do not require 3D supervision and are described in [39]) and NVSR (results of $200 \rightarrow 800$ were provided to us by the authors) [4].

	TriNeRFLetSR	TriNeRFLetSR w/o multiscale	TriNeRFLetSR w/o LPIPS	Our SR scheme with INGP
PSNR↑	**29.49**	28.49	29.27	28.25
LPIPS↓	**0.051**	0.057	0.06	0.06
SSIM↑	**0.930**	0.919	0.903	0.915

Table 2: Ablation: Impact of wavelet multiscale and LPIPS on SR. To show the significance of having the multi-scale structure of the wavelet, we apply our SR scheme without using the multiscale rendering or with INGP instead of TriNeRFLet rendering. We also show the significance of adding the LPIPS loss.

NeRF* is regular NeRF trained on LR views and then rendered in HR resolution, NeRF-Bi, NeRF-Liif and NeRF-Swin are trained on LR views, then rendered in LR resolution and upscaled using different 2D SR methods [39].

Results are reported in Table 3. Note that NVSR requires 3D supervision (paired LR and HR multiview images) and therefore it is considered only as a reference. When compared only to methods that use 2D supervised methods, we perform the best in most of the cases. Also, our method's results are very close to NVSR, and even slightly outperforms it in some cases, which demonstrates the superiority of our method despite the fact that it was not trained using 3D data. Qualitative examples of our approach appear in Figure 5 and sup. mat.

Impact of multiscale on SR. To demonstrate the impact of TriNeRFLet multiscale on the SR scheme, we modify it to neutralize the multiscale element. This is done by replacing the low-resolution rendering that uses low-resolution wavelet features (first L_{LR} layers) with a downscaled HR rendering that uses all wavelet layers (i.e. without multiscale render). In the same manner, INGP is also tested as a representation. Results in Table 2 demonstrate the important contribution of the TriNeRFLet multiscale structure to the SR scheme, utilizing its full potential, and that without it the SR scheme performs worse.

Impact of LPIPS loss. To demonstrate the LPIPS loss impact on SR, we trained the scheme without the LPIPS loss. Results are reported in Table 2, and as noticed, the LPIPS component indeed improves the SR scheme performance.

LLFF. We turn to demonstrate the TriNeRFLet-based SR scheme on the LLFF dataset [25], which contains 8 real-world captured scenes. As in the Blender SR case, we compare our method with the same methods but for input resolution 378×504 and $\times 4$ upscaling. Following [39], we trained TriNeRFLet using the low-resolution version of all images and then tested the SR results also on all the images. Table 3 reports the results. Our method is almost as good as [39] in PSNR and outperforms it in the LPIPS and SSIM metrics by a margin. Our LPIPS and SSIM performance indicates that our method manages to create a scene with better visual details and that is perceptually more similar to the ground-truth high-resolution scene.

6 Conclusions

This paper introduced a new NeRF structure that relies on a multiscale wavelet representation. This structure improved the performance of Triplane, which

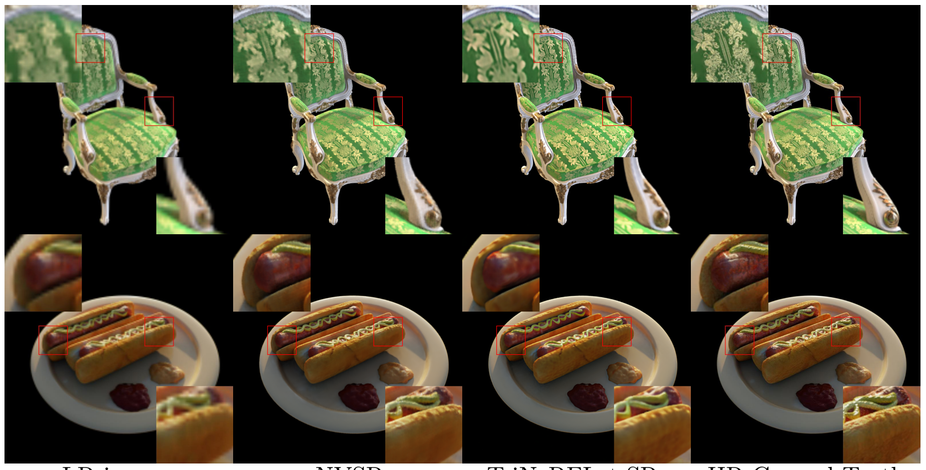


Fig. 5: *TriNeRFLet SR qualitative results.* Note the visual improvement that is achieved by our approach to NeRF SR compared to LR reconstruction. Moreover, the qualitative results of our method compete with NVSR, which unlike our strategy is a 3D supervised method that we consider as a reference. In the sup. mat. we provide addtional visual examples with more comparisons to other methods.

	Blender			Blender			LLFF		
	$100 \rightarrow 400$			$200 \rightarrow 800$			$378 \times 504 \rightarrow 1512 \times 2016$		
Method	PSNR↑	LPIPS↓	SSIM↑	PSNR↑	LPIPS↓	SSIM↑	PSNR↑	LPIPS↓	SSIM↑
NeRF*	25.56	0.170	0.881	27.47	0.128	0.900	24.47	0.388	0.701
NeRF-Bi	24.74	0.244	0.868	26.67	0.175	0.900	23.90	0.481	0.676
NeRF-Liif	25.36	0.125	0.885	27.34	0.096	0.912	24.76	0.292	0.723
NeRF-Swin	24.85	0.108	0.881	26.78	0.086	0.906	23.26	0.247	0.685
NeRF-SR (SS)	28.07	0.071	**0.921**	28.46	0.076	0.921	**25.13**	0.244	0.730
TriNeRFLetSR	**28.55**	**0.061**	0.913	29.49	**0.051**	0.930	25.00	**0.203**	**0.771**
NVSR (3D supervised)		**		**29.53**	0.054	**0.931**		**	

Table 3: *Super Resolution* $\times 4$ *Results.* NVSR is trained with multiview (3D) supervision and therefore we consider it as reference. **Bold** is best, underline is second. **For $100 \rightarrow 400$ NVSR reports results of only 4 shapes in the Blender dataset. In this case, we outperform it with PSNR of 29.18dB for our method compared to 28.5dB of NVSR. For LLFF, NVSR did not report any results on this resolution or LLFF average.

lagged behind NeRF state-of-the-art methods, achieving competitive results. Having the multiscale structure enabled us to implement a new diffusion-guided SR for NeRF. We believe that the concept of learning the features in the multiscale wavelet domain instead of the original one has the potential to improve other vision-related applications.

Despite the advantages of our approach, it also has some limitations, mainly due to the runtime overhead it adds to training time. This can be partially mitigated by using its light-weight versions and the coarse to fine training, which reduce training time. We defer further run-time improvements to future work.

Acknowledgments This work was supported in part by KLA grant.

References

1. Abu-Hussein, S., Tirer, T., Giryes, R.: Adir: Adaptive diffusion for image reconstruction. arXiv preprint arXiv:2212.03221 (2022)
2. Avrahami, O., Fried, O., Lischinski, D.: Blended latent diffusion. arXiv preprint arXiv:2206.02779 (2022)
3. Avrahami, O., Lischinski, D., Fried, O.: Blended diffusion for text-driven editing of natural images. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 18208–18218 (2022)
4. Bahat, Y., Zhang, Y., Sommerhoff, H., Kolb, A., Heide, F.: Neural volume super-resolution. arXiv preprint arXiv:2212.04666 (2022)
5. Barron, J.T., Mildenhall, B., Tancik, M., Hedman, P., Martin-Brualla, R., Srinivasan, P.P.: Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 5855–5864 (2021)
6. Barron, J.T., Mildenhall, B., Verbin, D., Srinivasan, P.P., Hedman, P.: Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 5470–5479 (2022)
7. Chan, E.R., Lin, C.Z., Chan, M.A., Nagano, K., Pan, B., De Mello, S., Gallo, O., Guibas, L.J., Tremblay, J., Khamis, S., et al.: Efficient geometry-aware 3d generative adversarial networks. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 16123–16133 (2022)
8. Chung, H., Lee, E.S., Ye, J.C.: Mr image denoising and super-resolution using regularized reverse diffusion. IEEE Transactions on Medical Imaging **42**(4), 922–934 (2023)
9. Dhariwal, P., Nichol, A.: Diffusion models beat gans on image synthesis. Advances in neural information processing systems **34**, 8780–8794 (2021)
10. Gal, R., Hochberg, D.C., Bermano, A., Cohen-Or, D.: Swagan: A style-based wavelet-driven generative model. ACM Trans. Graph. **40**(4) (2021)
11. Guo, T., Zhang, T., Lim, E., Lopez-Benitez, M., Ma, F., Yu, L.: A review of wavelet analysis and its applications: Challenges and opportunities. IEEE Access **10**, 58869–58903 (2022)
12. Guth, F., Coste, S., De Bortoli, V., Mallat, S.: Wavelet score-based generative modeling. In: Advances in Neural Information Processing Systems. vol. 35, pp. 478–491 (2022)
13. Haque, A., Tancik, M., Efros, A., Holynski, A., Kanazawa, A.: Instruct-nerf2nerf: Editing 3d scenes with instructions. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (2023)
14. Hertz, A., Perel, O., Giryes, R., Sorkine-Hornung, O., Cohen-Or, D.: Sape: Spatially-adaptive progressive encoding for neural optimization. Advances in Neural Information Processing Systems **34**, 8820–8832 (2021)
15. Ho, J., Jain, A., Abbeel, P.: Denoising diffusion probabilistic models. Advances in neural information processing systems **33**, 6840–6851 (2020)
16. Kadkhodaie, Z., Guth, F., Mallat, S., Simoncelli, E.P.: Learning multi-scale local conditional probability models of images. In: The Eleventh International Conference on Learning Representations (2023), `https://openreview.net/forum?id=VZX2I_VVJKH`

17. Kawar, B., Elad, M., Ermon, S., Song, J.: Denoising diffusion restoration models. In: Advances in Neural Information Processing Systems (2022)
18. Kawar, B., Zada, S., Lang, O., Tov, O., Chang, H., Dekel, T., Mosseri, I., Irani, M.: Imagic: Text-based real image editing with diffusion models. arXiv preprint arXiv:2210.09276 (2022)
19. Kerbl, B., Kopanas, G., Leimkühler, T., Drettakis, G.: 3d gaussian splatting for real-time radiance field rendering. ACM Transactions on Graphics (ToG) **42**(4), 1–14 (2023)
20. Li, M., Zhou, P., Liu, J.W., Keppo, J., Lin, M., Yan, S., Xu, X.: Instant3d: Instant text-to-3d generation (2023)
21. Lin, C.H., Ma, W.C., Torralba, A., Lucey, S.: Barf: Bundle-adjusting neural radiance fields. arXiv preprint arXiv:2104.06405 (2021)
22. Lindell, D.B., Van Veen, D., Park, J.J., Wetzstein, G.: Bacon: Band-limited coordinate networks for multiscale scene representation. In: CVPR (2022)
23. Mallat, S.: A Wavelet Tour of Signal Processing, Third Edition: The Sparse Way. Academic Press, Inc. (2008)
24. Martin-Brualla, R., Radwan, N., Sajjadi, M.S.M., Barron, J.T., Dosovitskiy, A., Duckworth, D.: NeRF in the Wild: Neural Radiance Fields for Unconstrained Photo Collections. In: CVPR (2021)
25. Mildenhall, B., Srinivasan, P.P., Ortiz-Cayon, R., Kalantari, N.K., Ramamoorthi, R., Ng, R., Kar, A.: Local light field fusion: Practical view synthesis with prescriptive sampling guidelines. ACM Transactions on Graphics (TOG) **38**(4), 1–14 (2019)
26. Mildenhall, B., Srinivasan, P.P., Tancik, M., Barron, J.T., Ramamoorthi, R., Ng, R.: Nerf: Representing scenes as neural radiance fields for view synthesis. Communications of the ACM **65**(1), 99–106 (2021)
27. Müller, T., Evans, A., Schied, C., Keller, A.: Instant neural graphics primitives with a multiresolution hash encoding. ACM Transactions on Graphics (ToG) **41**(4), 1–15 (2022)
28. Park, K., Sinha, U., Barron, J.T., Bouaziz, S., Goldman, D.B., Seitz, S.M., Martin-Brualla, R.: Deformable neural radiance fields. arXiv preprint arXiv:2011.12948 (2020)
29. Phung, H., Dao, Q., Tran, A.: Wavelet diffusion models are fast and scalable image generators. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 10199–10208 (2023)
30. Qiu, Q., Cheng, X., Sapiro, G., et al.: Dcfnet: Deep neural network with decomposed convolutional filters. In: International Conference on Machine Learning. pp. 4198–4207 (2018)
31. Rho, D., Lee, B., Nam, S., Lee, J.C., Ko, J.H., Park, E.: Masked wavelet representation for compact neural radiance fields. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 20680–20690 (2023)
32. Rombach, R., Blattmann, A., Lorenz, D., Esser, P., Ommer, B.: High-resolution image synthesis with latent diffusion models. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 10684–10695 (2022)
33. Saharia, C., Chan, W., Saxena, S., Li, L., Whang, J., Denton, E.L., Ghasemipour, K., Gontijo Lopes, R., Karagol Ayan, B., Salimans, T., et al.: Photorealistic text-to-image diffusion models with deep language understanding. Advances in Neural Information Processing Systems **35**, 36479–36494 (2022)
34. Saharia, C., Ho, J., Chan, W., Salimans, T., Fleet, D.J., Norouzi, M.: Image super-resolution via iterative refinement. IEEE Transactions on Pattern Analysis and Machine Intelligence (2022)

35. Saragadam, V., LeJeune, D., Tan, J., Balakrishnan, G., Veeraraghavan, A., Baraniuk, R.G.: Wire: Wavelet implicit neural representations. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 18507–18516 (2023)
36. Shue, J.R., Chan, E.R., Po, R., Ankner, Z., Wu, J., Wetzstein, G.: 3d neural field generation using triplane diffusion. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 20875–20886 (2023)
37. Tancik, M., Srinivasan, P.P., Mildenhall, B., Fridovich-Keil, S., Raghavan, N., Singhal, U., Ramamoorthi, R., Barron, J.T., Ng, R.: Fourier features let networks learn high frequency functions in low dimensional domains. NeurIPS (2020)
38. Tang, J.: Torch-ngp: a pytorch implementation of instant-ngp (2022), https://github.com/ashawkey/torch-ngp
39. Wang, C., Wu, X., Guo, Y.C., Zhang, S.H., Tai, Y.W., Hu, S.M.: Nerf-sr: High quality neural radiance fields using supersampling. In: Proceedings of the 30th ACM International Conference on Multimedia. pp. 6445–6454 (2022)
40. Whang, J., Delbracio, M., Talebi, H., Saharia, C., Dimakis, A.G., Milanfar, P.: Deblurring via stochastic refinement. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 16293–16303 (2022)
41. Wu, R., Mildenhall, B., Henzler, P., Park, K., Gao, R., Watson, D., Srinivasan, P.P., Verbin, D., Barron, J.T., Poole, B., et al.: Reconfusion: 3d reconstruction with diffusion priors. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 21551–21561 (2024)
42. Yu, A., Ye, V., Tancik, M., Kanazawa, A.: pixelNeRF: Neural radiance fields from one or few images. In: CVPR (2021)
43. Yu, J.J., Derpanis, K.G., Brubaker, M.A.: Wavelet Flow: Fast training of high resolution normalizing flows. In: NeurIPS (2020)