Neural Fourier Filter Bank

Zhijie Wu   Yuhe Jin   Kwang Moo Yi
University of British Columbia
{zhijiewu, yuhejin, kmyi}@cs.ubc.ca

Figure 1. **Teaser** – We propose neural Fourier filter bank to perform spatial and frequency-wise decomposition jointly, inspired by wavelets. Our method provides significantly improved reconstruction quality given the same computation and storage budget, as represented by the PSNR curve and the error image overlay. Relying only on space partitioning without frequency resolution (InstantNGP) [37] or frequency encodings without space resolution (SIREN) [47] provides suboptimal performance and convergence. Simply considering both (ModSine) [34] enhances scalability when applied to larger scenes, but not in terms of quality and convergence.

**Abstract**

We present a novel method to provide efficient and highly detailed reconstructions. Inspired by wavelets, we learn a neural field that decompose the signal both spatially and frequency-wise. We follow the recent grid-based paradigm for spatial decomposition, but unlike existing work, encourage specific frequencies to be stored in each grid via Fourier features encodings. We then apply a multi-layer perceptron with sine activations, taking these Fourier encoded features in at appropriate layers so that higher-frequency components are accumulated on top of lower-frequency components sequentially, which we sum up to form the final output. We demonstrate that our method outperforms the state of the art regarding model compactness and convergence speed on multiple tasks: 2D image fitting, 3D shape reconstruction, and neural radiance fields. Our code is available at https://github.com/ubc-vision/NFFB.

1. Introduction

Neural fields [59] have recently been shown to be highly effective for various tasks ranging from 2D image compression [10, 64], image translation [4, 49], 3D reconstruction [41, 48], to neural rendering [1, 36, 37]. Since the introduction of early methods [36, 38, 48], efforts have been made to make neural fields more efficient and scalable. Among various extensions, we are interested in two particular directions: those that utilize spatial decomposition in the form of grids [7, 37, 51] that allow fast training and level of detail; and those that encode the inputs to neural fields with high-dimensional features via frequency transformation such as periodic sinusoidal representations [36, 47, 53] that fight the inherent bias of neural fields that is towards low-frequency data [53]. The former drastically reduced the training time allowing various new application areas [11, 52, 58, 60], while the latter has now become a standard operation when applying neural fields.

While these two developments have become popular, a caveat in existing works is that they do not consider the two together—all grids are treated similarly and interpreted together by a neural network. We argue that this is an important oversight that has a critical outcome. For a model to be efficient and accurate, different grid resolutions should focus on different frequency components that are properly localized. While existing grid methods that naturally local-
ize signals—can learn to perform this frequency decomposition, relying purely on learning may lead to sub-optimal results as shown in Fig. 1. This is also true when locality is not considered, as shown by the SIREN [47] example. Explicit consideration of both together is hence important.

This caveat remains true even for methods that utilize both grids and frequency encodings for the input coordinates [37] as grids and frequency are not linked, and it is up to the deep networks to find out the relationship between the two. Thus, there has also been work that focuses on jointly considering both space and frequency [18, 34], but these methods are not designed with multiple scales in mind thus single-scale and are designed to be non-scalable. In other words, they can be thought of as being similar to short-time Fourier transform in signal processing.

Therefore, in this work, we propose a novel neural field framework that decomposes the target signal in both space and frequency domains simultaneously, analogous to the traditional wavelet decomposition [46]; see Fig. 1. Specifically, a signal is decomposed jointly in space and frequency through low- and high-frequency filters as shown in Fig. 2. Here, our core idea is to realize these filters conceptually as a neural network. We implement the low-frequency path in the form of Multi-Layer Perceptrons (MLP), leveraging their frequency bias [53]. For the high-frequency components, we implement them as lookup operations on grids, as the grid features can explicitly enforce locality over a small spatial area and facilitate learning of these components. This decomposition is much resemblant of filter banks in signal processing, thus we name our method neural Fourier filter bank.

In more detail, we utilize the multi-scale grid structure as in [20, 37, 51], but with a twist—we apply frequency encoding in the form of Fourier Features just before the grid features are used. By doing so, we convert the linear change in grid features that arise from bilinear/trilinear interpolation to appropriate frequencies that should be learned at each scale level. We then compose these grid features together through an MLP with sine activation functions, which takes these features as input at each layer, forming a pipeline that sequentially accumulates higher-frequency information as composition is performed as shown in Fig. 2. To facilitate training, we initialize each layer of the MLP with the target frequency band in mind. Finally, we sum up all intermediate outputs together to form the estimated field value.

We demonstrate the effectiveness of our method under three different tasks: 2D image fitting, 3D shape reconstruction, and Neural Radiance Fields (NeRF). We show that our method achieves a better trade-off between the model compactness versus reconstruction quality than the state of the arts. We further perform an extensive ablation study to verify where the gains are coming from.

To summarize, our contributions are as follows:

• we propose a novel framework that decomposes the modeled signal both spatially and frequency-wise;
• we show that our method achieves better trade-off between quality and memory on 2D image fitting, 3D shape reconstruction, and Neural Radiance Fields (NeRF);
• we provide an extensive ablation study shedding insight into the details of our method.

2. Related Work

Our work is in line with those that apply neural fields to model spatial-temporal signals [5, 6, 26, 35, 36, 38, 43]. In this section, we survey representative approaches on neural field modeling [2, 51, 37, 38, 51] and provide an overview of work on incorporating the wavelet transform into deep network designs [8, 13, 22].

Neural fields. A compressive survey can be found in [59]. Here we briefly discuss representative work. While existing methods have achieved impressive performance on modeling various signals that can be represented as fields [9, 16, 17, 35, 37, 38, 40, 51], neural fields can still fall short of representing the fine details [16], or incur high computational cost due to model complexity [23]. Prior works attempt to solve these problems by frequency transformations [36, 47, 53] and grid-based encodings [16, 37, 51].

For frequency transformations [37], Vaswani et al. [55] encode the input feature vectors into a high-dimension latent space through a sequence of periodic functions. Tanick et al. [53] carefully and randomly choose the frequency of the periodic functions and reveal how they affect the fidelity of results. Sitzmann et al. [47] propose to use periodic activation functions instead of encoding feature vectors. [12, 28] further push analysis in terms of the spectral domain with a multi-scale strategy, improving the capability in modeling band limited signals in a single model. To further understand the success of these methods, [3, 62]
analyze the implicit representations from the perspective of a structured dictionary and Fourier series, respectively.

For grid-based encodings [37, 51], the core idea is to encode the input to the neural field by interpolating a learnable basis consisting of grid-point features (space partitioning). A distinctive benefit of doing so is that one can trade memory for faster training—bigger networks can be used to represent complex scenes, as long as the entire grid used is within memory. To reduce this memory footprint, compact hash tables [37] and volumetric matrix decomposition [7] have been introduced. These recent methods, however, do not, at the very least explicitly, consider how grid resolutions and frequency interact.

Thus, some works try to combine both directions. For example, SAPE [18] progressively encodes the input coordinates by attending to time-spatial information jointly. Mehta et al. [34] decompose the inputs into patches, which are used to modulate the activation functions. They, however, utilize a single space resolution, limiting their modeling capability. Instead, we show that by using multiple scale levels, and a framework that takes into account the frequencies that are to be associated with these levels, one can achieve faster convergence with higher accuracy.

Wavelets in deep nets. The use of wavelet transforms has been well-studied in the deep learning literature. For example, they have been used for wavelet-based feature pooling operations [14, 30, 57], for the improvements on style transfer [13, 61], for denoising [29], for medical analysis [24], and for image generation [21, 32, 42, 56, 56]. Recently, Liang et al. [27] reproduce wavelets through linearly combining activation functions. Gauthier et al. [15] introduce wavelet scattering transform to create geometric invariants and deformation stability. Phung et al. [42] use Haar wavelets with diffusion models to accelerate convergence. In the 3D vision domain, De Queiroz et al. [8] propose a transformation that resembles an adaptive variation of Haar wavelets to facilitate 3D point cloud compression. Isik et al. [22] directly learn trainable coefficients of the hierarchical Haar wavelet transform, reporting impressive compression results. Concurrently, Rho et al. [44] propose using wavelet coefficients to improve model compactness. While our work shares a similar spirit as those that utilize wavelets, to the best of our knowledge, ours is the first work aimed at a general-purpose neural field architecture that jointly and explicitly models the spatial and frequency domains.

3. Method

In this work, we aim for a multi-resolution grid-based framework that also ties in the frequency space to these grids, as is done with wavelets, and an architecture to effectively reconstruct the original signal. As shown in Fig. 2, we construct our pipeline, neural Fourier filter bank, composed of two parts: a Fourier-space analogous version of grid features (Sec. 3.1); and an MLP that composes the final signals from these grid values (Sec. 3.2). We discuss these in more detail in the following subsections.

3.1. The Fourier grid features

As discussed earlier in Sec. 1, we use a grid setup to facilitate the learning of high-frequency components via locality. Specifically, we aim for each grid level in the multi-grid setup to store different frequency bands of the field that we wish to store in the neural network. The core idea in how we achieve this is to combine the typical grid setup used by, e.g. [37], with Fourier features [53], which we then initialize appropriately to naturally encourage a given grid to focus on certain frequencies. This is analogous to how one can control the frequency details of a neural field by controlling the Fourier feature [53] encoding of the input coordinates, but here we are applying it to the grid features.

In more detail, the grid feature at the $i$-th level is defined as a continuous mapping from the input coordinate $x \in \mathbb{R}^n$ to $m$ dimension feature space:

$$\kappa_i : \mathbb{R}^n \rightarrow \mathbb{R}^m. \quad (1)$$

We set $n = 2, 3$ for 2D images and 3D shapes respectively. As shown in Fig. 3, $\kappa_i$ consists of two parts: a lookup table $\Phi_i$ which has $T_i$ feature vectors with dimensionality $F$; and a Fourier feature layer [53] $\Omega_i$.

Multi-scale grid. We apply a trainable hash table [37] to implement $\Phi_i$ for a better balance between performance and quality. For the $i$-th level, we store the feature vectors at the vertices of a grid, the resolution of which $N_i$ is chosen manually. To utilize this grid in a continuous coordinate setup, one typically performs linear interpolation [37, 51].

Hence, for a continuous coordinate $x$, to get the grid points, for each dimension we first scale $x$ by $N_i$ before rounding down and up, which we write with a slight abuse of notation (ignoring dimensions) as:

$$[x_i] = \lfloor x \cdot N_i \rfloor, [x_i] = \lfloor x \cdot N_i \rfloor. \quad (2)$$

Here, $[x_i]$ and $[x_i]$, for example occupies a voxel with $2^n$ integer vertices. As in [37], we then map each corner vertex to an entry in the matching lookup table, using a spatial hash function [37, 54] as:

$$h(\bar{x}) = \left\lfloor \prod_{i=1}^{n} \bar{x}_i \cdot \Pi_i \right\rfloor \mod T_1, \quad (3)$$

where $\bar{x}$ represents the position of a specific corner vertex, $\wedge$ denotes the bit-wise XOR operation and $\Pi_i$ are unique, large prime numbers. As in [37], we choose $\Pi_1 = 1, \Pi_2 = 2654435761$ and $\Pi_3 = 805459861$.

Finally, for $x$, we perform linear interpolation for its $2^n$ corner feature vectors based on their relative position to $x$. 

within its hypercube as \( w_i = x_i - [x_i] \). Specifically, we use bilinear interpolation for 2D image fitting and trilinear interpolation for 3D shape modeling. We denote the output features through the linear interpolation over the lookup table \( \Phi_i \), as \( \varphi(x; \Phi_i) \).

It is important to note that this linear interpolation operation makes these features behave similarly to how the input coordinates affect the neural field output [53]—introducing bias toward slowly changing components. Thus, in order for each grid level to focus on appropriate frequency bands it is necessary to explicitly take this into account.

**Converting grid features to Fourier features.** Then, to associate the spatial area with the specific frequency level, we apply Fourier feature encoding to \( v_i = \varphi(x; \Phi_i) \) before we utilize them:

\[
\gamma_i(v_i) = [\sin(2\pi \cdot B_{i,1} \cdot v_i^1), \ldots, \sin(2\pi \cdot B_{i,m} \cdot v_i^m)]^T,
\]

where \( \{B_{i,1}, B_{i,2}, \ldots, B_{i,m}\} \) means trainable frequency transform coefficients on \( i \)-th level. We then utilize \( \gamma_i(v_i) \) in our network that converts these into desired field values.

Importantly, we directly associate the frequency band on the \( i \)-th level with desired grid size by explicitly initializing \( \{B_{i,1}, B_{i,2}, \ldots, B_{i,m}\} \) with adaptive Gaussian distribution variance similarly to Gaussian mapping [53, Sec. 6.1]. We choose to initialize with different variances, as it is difficult to set a specific frequency range for a given grid \textit{a priori}. Instead of trying to set a proper range that is hard to accomplish, we initialize finer grids with larger variance and naturally bias finer grids towards higher frequency components since the multiplier for \( v \) will then be larger—they will be biased to converge to larger frequencies [18].

### 3.2. Composing the field value

To compose the field values from our Fourier grid features, we start from two important observations:

- The stored Fourier grid features at different layers, after going through a deep network layer for interpretation, are not orthogonal to each other. This calls for the need for learned layers when aggregating features from different levels so that this non-orthogonality is mitigated.
- The Fourier grid features should be at a similar ‘depth’ so that they are updated simultaneously. This makes residual setups preferable.

We thus utilize an MLP, which takes in the Fourier grid features at various layers. As shown in Fig. 3, each layer takes in features from the previous layer, as well as the Fourier grid features, then either passes it to the next layer or to an output feature that is then summed up to form a final output.

Mathematically, denoting the MLP as a series of fully-connected layers \( \mathcal{L} = \{L_1, L_2, \ldots\} \), we write

\[
f_i = \sin(\alpha_i \cdot W_i g_{i-1} + b_i), \quad g_i = f_i + \gamma_i(v_i),
\]

where \( W_i \) and \( b_i \) are trainable weight and bias in the \( i \)-th layer \( L_i \), and \( \alpha_i \) is the scaling factor for this layer that control the frequency range that this layer focuses on, which is equivalent to the \( \psi_0 \) hyperparameter in SIREN [47]. Note here that \( f_i \) corresponds to the output of the lower-frequency component, and the Fourier grid features \( \gamma_i(v_i) \) are the higher-frequency ones in Fig. 2. For the first layer, as there is no earlier level, we use the input position \( x \). Thus,

\[
f_1 = \sin(\alpha_1 \cdot W_1 x + b_1), \quad g_1 = f_1 + \gamma_1(v_1).
\]
representing large-scale 2D images. For all models, we train

4.1. 2D Image Fitting

More experiment discussions can be found in the appendix.

g. Ablation study is shown in Sec. 4.4. This results in consistently inferior performance compared to our method of composition.

3.3. Implementation details

Depending on the target applications, some implementation details vary—the loss function, the number of training iterations, and the network capacity are task dependent and we elaborate on them later in their respective subsections. Other than the task-specific components we keep the same training setup for all experiments. We implement our method in PyTorch [39]. We use the Adam optimizer [25] with default parameters $\beta_1 = 0.9$ and $\beta_2 = 0.99$. We use a learning rate of $10^{-4}$, and decay the learning rate to half every 5,000 iterations. We set the dimension of grid features as $F = 2$. We train our method on a single NVidia RTX 3090 GPU. Here, for brevity, we note only the critical setup for each experiment. For more details on the architectures and the hyperparameter settings, please see the supplementary material.

4. Experimental Results

We evaluate our method on three different tasks: 2D image fitting (Sec. 4.1), 3D shape reconstruction using signed distance functions (Sec. 4.2), and novel view synthesis using NeRF (Sec. 4.3). Ablation study is shown in Sec. 4.4. More experiment discussions can be found in the appendix.

4.1. 2D Image Fitting

We first validate the effectiveness of our method in representing large-scale 2D images. For all models, we train them with the mean squared error. Hence, our loss function for this task is

$$L_{img} = \|y - y_{gt}\|^2_2,$$

where $y$ is the neural field estimate and $y_{gt}$ is the ground-truth pixel color.

Data: To keep our experiments compatible with existing work, we follow ACORN [33] and evaluate each method on two very high-resolution images. The first image is a photo of ‘Einstein’1, already shown in Fig. 1. This image has a resolution of $3250 \times 4333$ pixels, with varying amounts of details in different regions of the image, making it an interesting image to test how each model is capable of representing various levels of detail—background is blurry and smooth, while the eye and the clothes exhibit high-frequency details. Another image is a photo of the nightscape of ‘Tokyo’ [33] with a resolution of $6144 \times 2324$, where near and far objects provide a large amount of detail at various frequencies.

Baselines. We compare our method against four different baselines designed for this task: InstantNGP [37], which utilizes grid based space partitions for the input; SIREN [47], which resembles modeling the Fourier space; and two methods (SAPE [18] and ModSine [34]) that consider both the frequency and the space decomposition but not as in our method. For all methods, we use the official implementation by the authors but change their model capacity (number of parameters, and grid/hash table size) and task-specific parameters. Specifically, for SIREN, we set the frequency parameter $\omega_0 = 30.0$ and initialize the network with 5 hidden layers with size $512 \times 512$. For SAPE, we preserve their original network size. For InstantNGP, we adjust its maximum hashable size as $T = 2^{17}$ and the grid level to $L = 8$ for the ‘Einstein’ image and set $T = 2^{19}$ and $L = 16$ for ‘Tokyo’ to better cater to complex details. To allow all models to fully converge, we report results after 50,000 iterations of training.

Results. We provide qualitative results for the ‘Tokyo’ image in Fig. 4, and report the quantitative metrics in Tab. 1. As shown, our method provides the best trade-off between model size and reconstruction quality, both in terms of Peak Signal-to-Noise Ratio (PSNR), Structural Similarity Index Metric (SSIM) [19], and Learned Perceptual Image Patch Similarity (LPIPS) [63]. Among these, note that the gap in performance is larger with SSIM and LPIPS, which better represents the local structure differences. This is also visible in Fig. 4, where our method provides results that are nearly indistinguishable from the ground truth.

We note that the importance of considering both frequency and space is well exemplified in Fig. 4. As shown, while InstantNGP provides good details for nearby regions (second row), as further away regions are investigated (third

---

Table 1. 2D Fitting – We report the reconstruction comparisons in terms of Peak Signal-to-Noise Ratio (PSNR), Structural Similarity Index Metric (SSIM) (SSIM) [19] and Learned Perceptual Image Patch Similarity (LPIPS) [63]. Our method provides the best trade-off between model size and reconstruction quality.

<table>
<thead>
<tr>
<th>Method</th>
<th>Tokyo</th>
<th>Albert</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Size (MB)</td>
<td>PSNR</td>
</tr>
<tr>
<td>InstantNGP [37]</td>
<td>36.0</td>
<td>33.38</td>
</tr>
<tr>
<td>SIREN [47]</td>
<td>5.2</td>
<td>28.52</td>
</tr>
<tr>
<td>SAPE [18]</td>
<td>3.2</td>
<td>21.64</td>
</tr>
<tr>
<td>ModSine [34]</td>
<td>3.5</td>
<td>23.23</td>
</tr>
<tr>
<td>Ours</td>
<td>10.0</td>
<td>33.62</td>
</tr>
</tbody>
</table>

and last row), artifacts are more visible. This demonstrates that even when multiscale grid is used, without consideration of the frequencies associated with these scales, results degrade. Other baselines, SIREN, ModSine, and SAPE, are all single-scale and show results as if they are focusing on a single frequency band. Ours on the other hand does not suffer from these artifacts.

4.2. 3D Shape Reconstruction

We further evaluate our method on the task of representing 3D shapes as signed distance fields (SDF). For this task, we use the square of the Mean Absolute Percentage Error (MAPE) [37] as training objective, to facilitate detail modeling. We thus train models by minimizing the loss:

$$L_{sdf} = \| y - y_{gt} \|_2^2 / \left( \epsilon + \| y_{gt} \|_2^2 \right),$$  

where $\epsilon$ denotes a small constant to avoid numerical problems. $y$ is the neural field estimate, and $y_{gt}$ is the ground-truth SDF value.

Data. For this task, we choose two standard textured 3D shapes for evaluation: ‘Bearded Man’ (with 691K vertices and 1.38M faces); and ‘Asian Dragon’ (3.6M vertices and 7.2M faces). Both shapes exhibit coarse and fine geometric details. When training with these shapes, we sample 3D
Figure 5. **3D Fitting** – Qualitative comparisons for the ‘Bearded Man’ shape. Our method is the most compact among the compared methods, and is capable of reconstructing both coarse and fine details without obvious artifacts.

<table>
<thead>
<tr>
<th>Size (MB)</th>
<th>Asian Dragon F-score</th>
<th>IoU</th>
<th>Cham dist</th>
<th>Bearded Man F-score</th>
<th>IoU</th>
<th>Cham dist</th>
</tr>
</thead>
<tbody>
<tr>
<td>InstantNGP [37]</td>
<td>46.5</td>
<td>0.8714</td>
<td>1.0</td>
<td>0.00191</td>
<td>0.999</td>
<td>0.9970</td>
</tr>
<tr>
<td>SIREN [47]</td>
<td>2.0</td>
<td>0.8593</td>
<td>0.998</td>
<td>0.00234</td>
<td>0.997</td>
<td>0.9951</td>
</tr>
<tr>
<td>BACON [55]</td>
<td>2.0</td>
<td>0.3200</td>
<td>0.995</td>
<td>0.00242</td>
<td>0.716</td>
<td>0.9932</td>
</tr>
<tr>
<td>SAPE [18]</td>
<td>3.2</td>
<td>0.3210</td>
<td>0.959</td>
<td>0.00584</td>
<td>0.284</td>
<td>0.9837</td>
</tr>
<tr>
<td>ModSine [51]</td>
<td>12.0</td>
<td>0.6892</td>
<td>0.995</td>
<td>0.00238</td>
<td>0.873</td>
<td>0.9952</td>
</tr>
<tr>
<td>Ours</td>
<td>1.4</td>
<td>0.8717</td>
<td>1.0</td>
<td>0.00189</td>
<td>0.999</td>
<td>0.9985</td>
</tr>
</tbody>
</table>

Table 2. **3D Fitting** – We report the Intersection over Union (IoU), F-Score and Chamfer distance (CD) after performing marching cubes to extract surfaces. Our method performs best, with the exception of F-score on ‘Asian Dragon’, which is due to BACON preferring blobby output, as demonstrated by the higher Chamfer distance and worse IoU.

Points \( x \in \mathbb{R}^3 \) with a 20/30/50 split—20% of the points are sampled uniformly within the volume, 30% of the points are sampled near the shape surface, and the rest sampled directly on the surface.

**Baselines.** We compare against the same baselines as in Sec. 4.1, and additionally BACON, which also utilizes frequency decomposition for efficient neural field modeling. For BACON and SIREN, we use networks with 8 hidden layers and 256 hidden features, and again \( \omega_l = 30.0 \) for SIREN. For ModSine, we set the grid resolution as \( 64 \times 64 \times 64 \) and apply 8 hidden layers and 256 hidden features for both the modulation network and the synthesis network. For SAPE and InstantNGP, use the author-tuned defaults for this task. All models are trained for 100K iterations for full training.

**Results.** We present our qualitative results in Fig. 5 and report quantitative scores in Tab. 2. To extract detailed surfaces from each implicit representation we apply marching cubes with a resolution of 1024\(^3\). As shown, our method provides the best performance, while having the smallest model size. Note that in Tab. 2 our results are worse in terms for F-score for the Asian Dragon, while the other metrics report performance comparable to InstantNGP with 30× smaller model size. The lower F-score but higher Chamfer distance is due to our model having lower recall than BACON, which provides more blobby results, as demonstrate by the IoU and Chamfer distance metrics. We also note that for the ‘Bearded Man’, our method outperforms all other methods.

This difference in quantitative metrics is also visible in Fig. 5. As shown, our method provides high-quality reconstruction for both zoomed-in regions, whereas other compared methods show lower-quality reconstructions for at least one of them. For example, SIREN provides good reconstruction for the beard region (second row), but not for the region around the ears (top row), where sinusoidal artifacts are visible. InstantNGP also delivers high-quality reconstruction for the ‘Bearded Man’, but with much higher memory requirement.

### 4.3. Novel View Synthesis

As our last task, we apply our method to modeling Neural Radiance Fields (NeRF) [36]. Because we are interested in comparing the neural field architectures, not the NeRF method itself, we focus on the simple setup using the synthetic Blender dataset.

We train all architectures with a pure NeRF setup [36],
where volumetric rendering is used to obtain pixel colors, which are then compared to ground-truth values for training. Specifically, a pixel color is predicted as

$$\hat{C}(r) = \sum_{i=1}^{n} T_i (1 - \exp(-\sigma_i \delta_i)) c_i,$$

$$T_i = \exp\left(-\sum_{j=1}^{i-1} \sigma_j \delta_j\right),$$

(9)

where $c_i$ and $\sigma_i$ denote the color and density estimated at the $i$-th queried location along the ray $r$ and $\delta_i$ is the distance between adjacent samples along a given ray. Then, the mean-squared loss for training is:

$$L_{rec} = \sum_{r \in \mathcal{R}} \left\| \hat{C}(r) - C_{gt}(r) \right\|_2^2,$$

(10)

where $\mathcal{R}$ is the whole ray set and $C_{gt}$ is the ground truth.

**Adaptation.** For this task, we found that the complexity of the task, estimating both the color and the density, requires appending our pipeline with an additional MLP that decodes deep features into either the color or the density. Thus, instead of directly outputting these values from our framework, we output a deep feature, which is then converted into color and density. Specifically, as in NeRF [36], we apply two $64 \times 64$ linear layers to predict density value and a low-dimension deep feature, which is further fed into three $64 \times 64$ linear layers for RGB estimation.

**Baselines.** We compare against five baselines: NeRF [36], Plenoxels [45], DVGO [50], and instantNGP [37], which are grid-based methods.

**Results.** We report our results in Fig. 6 and Tab. 3. Our method provides similar performance as other methods, but with a much smaller model size.

**4.4. Ablation Study**

To justify the design choices of our method we explore three variants of our method: our method where only Grid features are used as ‘Only Grid’; our method with Grid and the Fourier features encoding as ‘Grid+FF’; and finally when only using the MLP architecture for composition without the grid as ‘Only MLP’. For a fair evaluation of the effects of the MLP part, we adjust the ‘Only MLP’ model to possess similar number of trainable parameters as the full model. We report our results for the ‘Tokyo’ image in Fig. 7. As shown, all variants perform significantly worse. Interestingly, simply applying Fourier Features to the grid does not help, demonstrating the proposed MLP architecture is also necessary to achieve its potential.

**5. Conclusions**

We have proposed the neural Fourier filter bank, inspired by wavelets, that provide high-quality reconstruction with more compact models. We have shown that taking into account both the space and frequency is critical when decomposing the original signal as neural field grids. Our method provides the best trade-off between quality and model compactness for 2D image reconstruction, 3D shape representation, and novel-view synthesis via NeRF.
References


[33] Julien N.P. Martel, David B. Lindell, Connor Z. Lin, Eric R. Chan, Marco Monteiro, and Gordon Wetzstein. ACORN:
Adaptive Coordinate Networks for Neural Scene Representation. SIGGRAPH, 2021. 5


[37] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant Neural Gradients with a Multiresolution Hash Encoding. SIGGRAPH, 2022. 1, 2, 3, 5, 6, 7, 8, 12


[42] Hao Phung, Quan Dao, and Anh Tran. Wavelet diffusion models are fast and scalable image generators. arXiv preprint, 2022. 3


