Polynomial Neural Fields for Subband Decomposition and Manipulation

Guandao Yang[*] Cornell University
Sagie Benaim[*] University of Copenhagen
Varun Jampani Google Research
Kyle Genova Google Research
Jonathan T. Barron Google Research
Thomas Funkhouser Google Research
Bharath Hariharan Cornell University
Serge Belongie University of Copenhagen

Abstract

Neural fields have emerged as a new paradigm for representing signals, thanks to their ability to do it compactly while being easy to optimize. In most applications, however, neural fields are treated like a black box, which precludes many signal manipulation tasks. In this paper, we propose a new class of neural fields called basis-encoded polynomial neural fields (PNFs). The key advantage of a PNF is that it can represent a signal as a composition of a number of manipulable and interpretable components without losing the merits of neural fields representation. We develop a general theoretical framework to analyze and design PNFs. We use this framework to design Fourier PNFs, which match state-of-the-art performance in signal representation tasks that use neural fields. In addition, we empirically demonstrate that Fourier PNFs enable signal manipulation applications such as texture transfer and scale-space interpolation. Code is available at https://github.com/stevenygd/PNF.

1 Introduction

Neural fields are neural networks that take as input spatial coordinates and output a function of the coordinates, such as image colors [10, 60], 3D signed distance functions [3, 50], or radiance fields [43]. Recent works have shown that such representations are compact [18, 39, 55], allow sampling at arbitrary locations [60, 63], and are easy to optimize within the deep learning framework. These advantages enabled their success in many spatial visual computing applications including novel view synthesis [43, 45, 47, 59, 72] and 3D reconstruction [20, 31, 51, 59, 60]. Most recent neural field based methods, however, treat the network as a black box. As such, one can only obtain information by querying it with spatial coordinates. This precludes the applications where we want to change the signal represented in a neural field. For example, it is difficult to remove high frequency noise or change the stationary texture of an image.

One way to enable signal manipulation is to decompose a signal into a number of interpretable components and use these components for manipulation. A general approach that works across a wide range of signals is to decompose them into frequency subbands [1, 9, 58]. Such decompositions are studied in the traditional signal processing literature, e.g., the use of Fourier or Wavelet transforms of the spatial signal. These transformations, however, usually assume the signal is densely sampled

[*]Equal contribution. Part of this work was done while Guandao was a student researcher at Google.

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

in a regular grid. As a result, it is non-trivial to generalize these approaches to irregular data (e.g., point clouds). Another shortcoming of these transforms is that they require a lot of terms to represent a signal faithfully, which makes it difficult to scale to signals of more than two dimensions, such as in the case of light fields. Interestingly, these are the very problems that can be solved by neural fields, which can represent irregular signals compactly and scale easily to higher dimensions. This leads to the central question of this paper: can we incorporate the interpretability and controllability of classical signal processing pipelines to neural fields?

Our goal is to design a new class of neural fields that allow for precise subband decomposition and manipulation as required by the downstream tasks. To achieve that, our network needs to have the ability to control different parts of the network to output signal that is limited by desirable subbands. At the same time, we want the network to inherit the usual advantages of neural fields, namely, being compact, expressive, and easy to optimize. The most relevant prior works with related aims are Multiplicative Filter Network (MFN) [22] and BACON [36]. While MFNs enjoy the advantages of neural fields and are easy to analyze, they do not use this property to control the network's output for subband decomposition. BACON [36] extends the MFN architecture to enforce that its outputs are upper-band limited, but it lacks the ability to provide subband control beyond upper band limits. This hinders BACON's applicability to tasks that requires more precise control of subbands, such as manipulating stationary textures (shown in Sec. 4.2).

To address these issues, we propose a novel class of neural fields called **polynomial neural fields (PNFs)**. PNF is a polynomial neural network [12] evaluated using a set of basis functions. PNFs are compact, easy to optimize, and can be sampled at arbitrary locations as with general neural fields. Moreover, PNFs enjoy interpretability and controllability of signal processing methods. We provide a theoretical framework to analyze the output of PNFs. We use this framework to design the Fourier PNF, whose outputs can be localized in the frequency domain with both upper and lower band limits, along with orientation specificity. To the best of our knowledge, this is the first neural field architecture that can achieve such a fine-grained decomposition of a signal. Empirically, we demonstrate that Fourier PNFs can achieve subband decomposition while reaching state-of-the-art performance for signal representation tasks. Finally, we demonstrate the utility of Fourier PNFs in signal manipulation tasks such as texture transfer and scale-space interpolation.

2 Related Works

Our method are built on three bodies of prior works: signal processing, neural fields, and polynomial neural networks. In this section, we will focus on the most relevant part of these prior works. For further readings, please refer to Orfanidis [49] for signal processing, Xie et al. [70] for neural fields, and Chrysos et al. [15] for polynomial neural networks.

Fourier and Wavelet Transforms. In traditional signal processing pipeline, one usually first transforms the signal into weighted sums of functionals from certain basis before manipulating and analyzing the signal [49]. The Fourier and Wavelet transformation are most relevant to our work. In particular, prior works has leveraged Fourier and Wavelet transforms to organize image signal into meaningful and manipulable components such as the Laplacian [9] and Steerable [58] pyramids. In our paper, we analyze the signal in terms of the basis functions studied by Fourier and Wavelet transforms. Our manipulable components are also inspired by the subband used in Steerable Pyramid [57]. While this signal processing pipeline is very interpretable, it's non-trivial to make it work on irregular data because these transformations usually assume the signal to be densely sampled in a regular grid. In this paper, we tries to combines the interpretability of traditional signal processing pipeline with the merits of neural fields, which is easy to optimize even with irregular data.

Neural Fields. Neural Fields are neural networks that maps spatial coordinate to a signal. Recent research has shown that neural fields are effective in representing a wide variety of signals such as images [60, 62], 3D shapes [3, 17, 41, 50, 71], 3D scenes [20, 27, 31, 51, 59]. and radiance fields [4, 5, 6, 23, 32, 35, 37, 38, 40, 43, 44, 45, 47, 48, 52, 54, 59, 61, 67, 72, 73, 74, 76]. However, neural fields typically operate as black boxes, which hinders the application of neural fields to some signal decomposition and manipulation tasks as discussed in Sec. 1. Recent works have to alleviate such issue by designing network architecture that are partially interpretable. A common technique is to encode the input coordinate with a positional encoding where one can control spectrum properties such as the frequency bandwidth inputting into the network [4, 43, 60, 63, 77]. But these positional

encodings are passed through a black-box neural networks, making it difficult to analyze properties of the final output. The most relevant works is BACON [36], which propose an initialization schema for multiplicative filter networks (MFNs) [21] that ensures the output to be upper-limited by certain bandwidth. This work generalizes BACON in two ways. First, our theory can be applied to a more general set of basis function and network topologies. Second, our network enables more precise subband controls, which include band-limiting from above, below, and among certain orientation.

Polynomial Neural Networks. Polynomial neural networks (PNNs) are generally referred to neural networks composed of polynomials [15]. The study of PNNs can be dated back to higher-order boltzmann machine [56] and Mapping Units [29]. Recently, research has shown that PNNs can train very strong generative models [12, 14, 16, 26, 69] and recognition models [30, 68]. The empirical success has also followed with deeper theoretical analysis. For example, [16] reveal how PNNs' architecture relates to polynomial factorization. Kileel et al. [33] and Choraria et al. [11] studies the expressive power of PNNs. Our work establish the connection between PNNs and many neural fields such as MFN [22] and BACON [36]. We further extends polynomial neural networks by evaluating the polynomial with a set of basis functions such as the Fourier basis.

3 Method

In this section, we will provide a definition for polynomial neural fields (Sec. 3.1). From this definition, we derive a theoretical framework to analyze their outputs in terms of subbands (Sec. 3.2). We then use this framework to design Fourier PNFs, a novel neural fields architecture to represent signals as a composition of fine-grain subbands in frequency spaces (Sec. 3.3).

3.1 Polynomial Neural Fields

Recall that we would like to maintain the merits of the neural field representation while adding the ability to partition it into analyzable components. As with all neural networks, to guarantee expressivity, we want to base our neural fields on function compositions [53]. At the same time, we want our neural fields to be interpretable in terms of a set of basis functions that have known properties, as in the signal processing literature. We propose the following class of neural fields:

Definition 3.1 (PNF)**.** *Let $\mathcal{B}$ be a basis for the vector space of functions for $\mathbb{R}^n \to \mathbb{R}$. A* Polynomial neural field of basis *$\mathcal{B}$ is a neural network $f = g_L \circ \cdots \circ g_1 \circ \gamma$, where $\forall i, g_i$ are finite degree multivariate polynomials, and $\gamma : \mathbb{R}^n \to \mathbb{R}^d$ is a d-dimensional feature encoding using basis $\mathcal{B}$: $\gamma(x) = [\gamma_1(x) \ \dots \ \gamma_d(x)]^T, \gamma_i \in \mathcal{B}, \forall i$.*

This definition allows a rich design space that subsumes several prior works. For example, MFN [22] and BACON [36] can be instantiated by setting g_i to be either the linear layer or a masked multiplication layer. Similarly, if we set the basis to be $\mathcal{B} = \{x^n\}_{n \in \mathbb{N}}$, then we can show architectures proposed in Π-Net [16] are a subclass of PNF. This rich design space can potentially allow us to tailor the architecture to the application of interests, as demonstrated later in Sec. 3.3 and Sec. 4.3. Moreover, such rich space also contains many expressive neural networks, as shown by both the prior works [12, 16, 22, 36] and by our experiment in Sec. 4.1.

Furthermore, as long as the span of the basis is closed under multiplication, PNF yields a linear combination of basis functions and is thus easy to analyze:

Theorem 1. *Let F be a PNF with basis $\mathcal{B}$ s.t $\forall b_1, b_2 \in \mathcal{B}, b_1(x)b_2(x) = \sum_{i \in I} a_i b_i(x), |I| < \infty$. Then the output of F is a finite linear sum of the basis functions from $\mathcal{B}$.*

Many commonly used bases, such as Fourier, Gabor, and spherical harmonics, all satisfy this property. Please refer to the appendix for proofs for a variety of different bases.

3.2 Controllable Subband Decomposition

Theorem 1 is not enough to control or manipulate the signal represented by the network, because any single neuron in the network may potentially be working with an arbitrary set of basis functions. This is a problem if we want to manipulate the signal through these neurons. For example, if we want to discard the contributions of some high-frequency components in the Fourier basis, we cannot decide which neurons to discard if each of them are contributed to a variety of frequency bands.

To allow better manipulation, we use the notion of *subbands* from traditional frequency domain analysis [66]. In the most general sense, a subband is simply a *subset* of the basis. Manipulations such as smoothing or sharpening can then be done by discarding or enhancing the contributions of one or more such subbands. One way to instantiate these ideas with a PNF is to represent the signal as a sum of different PFNs, each of which is limited to only a specific subband; then one can manipulate these component PFNs separately.

Definition 3.2. *A PNF F of basis $\mathcal{B}$ is limited by a subband $S \subset \mathcal{B}$ if F is in the span of S.*

A key challenge is to construct a subband limited PNF for certain subbands. To this end, we need to understand how the subband-limited PNF transforms under different network operations. Fortunately, for PNFs we only need to study two types of operations: multiplication and addition. To this end, we need the notion of a *PNF-controllable set of subbands*:

Definition 3.3 (PNF-controllable Set of Subbands)**.** *$\mathcal{S} = \{S_\theta | S_\theta \subset \mathcal{B}\}_\theta$ is a PNF-Controllable Set of Subbands for basis $\mathcal{B}$ if (1) $S_{\theta_1} \cup S_{\theta_2} \in \mathcal{S}$ and (2) there exists a binary function $\otimes : \mathcal{S} \times \mathcal{S} \to \mathcal{S}$ such that if $b_1 \in S_{\theta_1}, b_2 \in S_{\theta_2} \implies b_1 b_2 \in S_{\theta_1} \otimes S_{\theta_2}$.* [2]

Intuitively, a PNF-controllable set of subbands lends some *predictability* to what happens if two PNFs, limited to different subbands, are put together into a larger PNF:

Theorem 2. *Let $\mathcal{S}$ be a PNF-controllable set of subbands of basis $\mathcal{B}$ with binary relation $\otimes$. Suppose F_1 and F_2 are polynomial neural fields of basis $\mathcal{B}$ that maps $\mathbb{R}^n$ to $\mathbb{R}^m$. Furthermore, suppose F_1 and F_2 are subband limited by $R_1 \in \mathcal{S}$ and $R_2 \in \mathcal{S}$. Then we have the following:*

1. *$W_1 F_1(x) + W_2 F_2(x)$ is a PNF of $\mathcal{B}$ limited by subband $R_1 \cup R_2$ with $W_1, W_2 \in \mathbb{R}^{m \times n}$;*
2. *$F_1(x)^T W F_2(x)$ is a PNF of $\mathcal{B}$ limited by subband $R_1 \otimes R_2$ with $W \in \mathbb{R}^{m \times m}$.*

Many structures used in the signal processing literature, such as the Steerable Pyramid, use subbands that are PNF-controllable [9, 58]. Please refer to the appendix for the derivations of PNF-controllable sets of subbands. In the following sections, we will focus on using Fourier bases and show how to build PNFs that instantiate subband manipulation efficiently.

3.3 Fourier PNF

In this section, we demonstrate how to build a PNF that can decompose a signal into frequency subbands in the following three steps. First, we identify a PNF-controllable sets of subband for the Fourier basis. Second, we choose a finite collection of subbands for the PNF to output, and organize them into controllable sets. The final step is to instantiate the PNF compactly using Theorem 2.

3.3.1 Controllable Subband Decomposition for Fourier Space

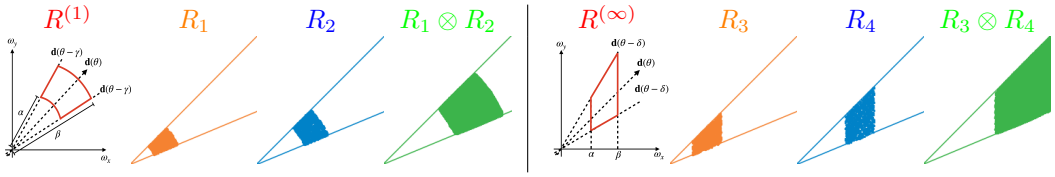

Figure 1: Illustration of PNF-controllable subbands in Fourier space. Left: 2-norm series of subbands; Right: infinite-norm series of subbands.

The Fourier basis can be written as: $b_{\boldsymbol{\omega}}(\mathbf{x}) = \exp(i\boldsymbol{\omega}^T \mathbf{x})$, where $\boldsymbol{\omega}, \mathbf{x} \in \mathbb{R}^d$ and i is the imaginary unit. It's easy to see that the fourier basis is complete under multiplication, which satisfies the condition for Theorem 1: $\exp(i\boldsymbol{\omega}_1^T \mathbf{x}) \exp(i\boldsymbol{\omega}_2^T \mathbf{x}) = \exp(i(\boldsymbol{\omega}_1 + \boldsymbol{\omega}_2)^T \mathbf{x})$.

Now we need to divide frequency space of $\boldsymbol{\omega} \in \mathbb{R}^d$ into subbands that are easy to manipulate and meaningful for downstream tasks. We define the subband following Simoncelli and Freeman [58]. Formally, frequency space is decomposed into following sectors:

$$R^{(p)}(\alpha, \beta, \mathbf{d}, \gamma) = \left\{ \boldsymbol{\omega} | \alpha \leq \|\boldsymbol{\omega}\|_p \leq \beta, \|\mathbf{d}\| = 1, \boldsymbol{\omega}^T \mathbf{d} < \gamma \|\boldsymbol{\omega}\| \right\}, \tag{1}$$

[2] It's possible prove Theorem 2 with a more relax version of Definition 3.3: $b_1 b_2 \in Span(S_{\theta_1} \otimes S_{\theta_2})$.

where p describe which norm do we choose to describe the frequency bands. Intuitively, α defines the lower band limits and β defines the upper band limits. The vector $\mathbf{d}$ defines the orientation of the subband and γ defines the angular width of the subband. Fig. 1 provides illustrations of $R^{(p)}$.

This definition of subband allows us to organize them into controllable sets. For example, we can show that the following sets of subbands are controllable:

Theorem 3. *Let $\mathcal{S}$ be a set of subbands defined as $\mathcal{S} = \{R^{(2)}(\alpha, \beta, \mathbf{d}, \gamma) | \forall 0 \leq \alpha \leq \beta\}$. If $|\gamma| < \frac{\pi}{4}$, then $\mathcal{S}$ is PNF-controllable. Specifically, if $\boldsymbol{\omega}_1 \in R_1^{(2)}(\alpha_1, \beta_1) \in \mathcal{S}$, and $\boldsymbol{\omega}_2 \in R_2^{(2)}(\alpha_2, \beta_2) \in \mathcal{S}$, then $b_{\boldsymbol{\omega}_i} b_{\boldsymbol{\omega}_i} = b_{\boldsymbol{\omega}_3}$ implies that $\boldsymbol{\omega}_3 \in R^{(2)}(\sqrt{\cos(2\gamma)}(\alpha_1 + \alpha_2), \beta_1 + \beta_2) \in \mathcal{S}$.*

This theorem captures the intuition that the multiplication of two waves of similar orientations creates high-frequency waves at that orientation. It allows us to predict the spectrum properties of the output of the network when knowing the spectrum properties of the inputs. Fig. 1 provides illustrations of how two 2D subbands interact under multiplications.

3.3.2 Subband Tiling

In order to represent different signals, our networks need to be able to leverage basis functions with different orientations and within different bandwidths. With that said, we need to choose a set of PNF-controllable subbands to cover all basis functions we want to use. For example, we can tile the space with the set of controllable subbands in Theorem 3 in the following way:

$$\mathcal{T}_{circ} = \{S_{ij} = R^{(2)}(b_i, b_{i+1}, \mathbf{d}(\theta_j), \delta) | b_1 \leq \cdots \leq b_n, \theta_j = j\delta, \delta = \frac{\pi}{m}, 1 \leq j \leq 2m\}, \quad (2)$$

where $\mathbf{d}(\theta) = [\sin(\theta), \cos(\theta)]^T$ denotes unit vector rotate with angle θ, $\delta = \frac{\pi}{m}$, and m sufficiently small to allow the application of Theorem 3.

For 2D images, the region of interest is $[-N, N]^2$ where N is the bandwidth determined by the Nyquist Sampling Theorem [46]. To tile this rectangular region well without introducing unnecessary high-frequency details, we will use pesudo polar coordinate grids, which divide the space into vertical or horizontal sub-regions and tile those sub-regions according to l-∞ norm. Formally, the 2D pseudo polar coordinate tiling can be written as:

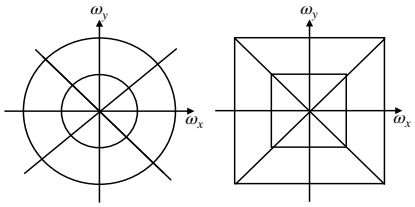

Figure 2: Left: $\mathcal{T}_{circ}$; Right: $\mathcal{T}_{rect}$.

$$\mathcal{T}_{rect} = \{S_{ij} = R^{(\infty)}(b_i, b_{i+1}, \mathbf{d}(\theta_j), \delta) | b_1 \leq \cdots \leq b_n, \theta_j = j\delta, 1 \leq j \leq 2m, j \neq m\}. \quad (3)$$

Note that we exclude certain regions to avoid having a subband to include orientation at $\frac{\pi}{4}$ and $\frac{3\pi}{4}$ in order for Theorem 3 to generalize to such tiling. Fig. 2 contains an illustration of these two types of tiling. We will show detailed derivation in the appendix.

Tiling the spectrum space with subbands from $\mathcal{T}_{circ}$ or $\mathcal{T}_{rect}$ allows us to organize information in a variety of meaningful ways. For example, this set of subbands can be grouped into different cones $\{S_{ij}\}_{j=1}^{2m}$. Each cone corresponds to a particular orientation of the signal. Alternatively, we can also organize this set of subbands into different rings $\{S_{ij}\}_{i=1}^{n}$, which corresponds to the decomposition of an image into the Laplacian Pyramid.

3.3.3 Network architecture

We now have decided on the set of subbands we want to produce by PNF. We want to design the final Fourier PNF as an ensemble of subband limited PNFs: $F(\mathbf{x}) = \sum_j \sum_i O_{ij} F_{ij}(\mathbf{x})$, where $F_{ij}(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R}^h$ is a PNF that's limited with subband S_{ij} defined in Eq. (3), and $O_{ij} \in \mathbb{R}^{h \times m}$ aggregates the output signals together. One way to achieve this is to naively define F_{ij} as a two layers PNF the feature encoding layer to include only the basis functions in the subband S_{ij} followed by a linear layer. However, such an approach fails to achieve good performance without a huge number of trainable parameters. Alternatively, we leverage Theorem 2 to factorize F:

$$F(\mathbf{x}) = \textstyle\sum_j F_j(\mathbf{x}), \; F_j(\mathbf{x}) = \displaystyle\sum_{k=1}^{n} G_j(\mathbf{x}, b_i, b_i) W_{jk} Z_{j,k}(\mathbf{x}), \quad (4)$$

$$Z_{j,1}(\mathbf{x}) = G_j(\mathbf{x}, 0, \Delta_1), \; Z_{j,k}(\mathbf{x}) = G_j(\mathbf{x}, 0, \Delta_k) W_i Z_{j,k-1}(\mathbf{x}), \quad (5)$$

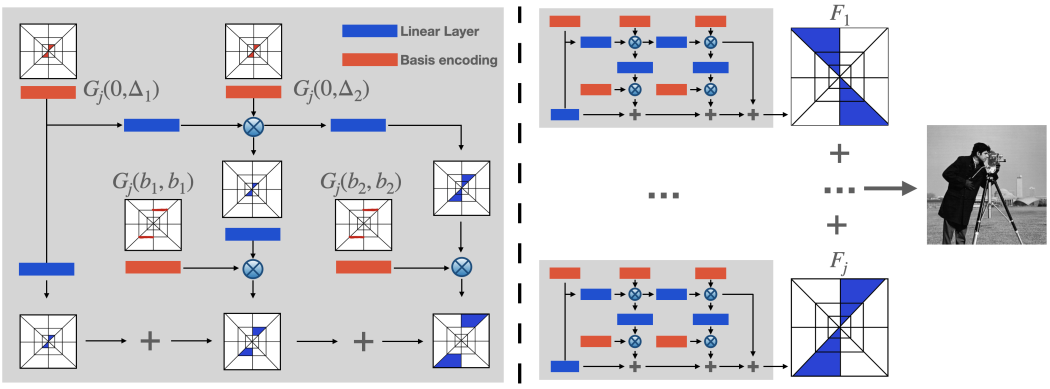

Figure 3: Illustration of Fourier PNF architecture. Fourier PNF is an ensemble model. The final result is summed over a series of PNFs F_j, whose structure is shown on the left side of the figure.

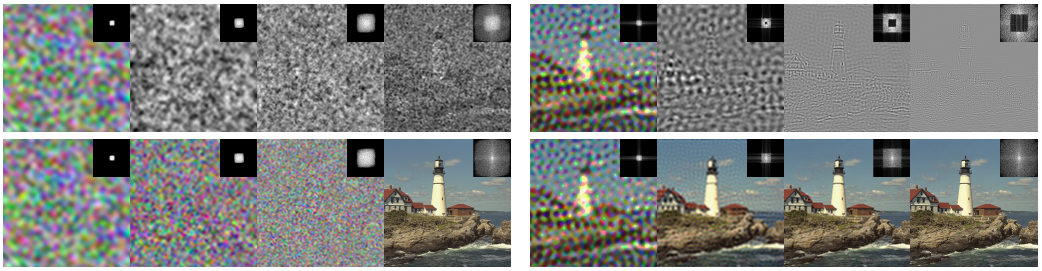

Figure 4: BACON: The bottom row shows the output of each layer which is upper band limited. The top row (columns 2-4) shows the difference between the output of a given layer and the one before it. PNF: The top row shows the output of each layer which is both upper and lower band limited. The bottom row (columns 2-4) shows the addition of the output at a given layer and the one before it.

where $G_j(\mathbf{x}, a, b)$ is subband limited in $R^{(\infty)}(a, b, d(\theta_j), \delta)$ and $\Delta_k = b_k - b_{k-1}$. This network architecture of F_i is illustrated in Fig. 3. We instantiate this architecture by setting $G_j(\mathbf{x}, a, b)$ into a linear transform of basis sampled from the subband to be limited:

$$G_j(\mathbf{x}, a, b) = W_i \gamma_j(\mathbf{x}), \gamma_j \in R^{(\infty)}(a, b, d(\theta_j), \delta)^d, W_i \in \mathbb{R}^{h \times d}, \tag{6}$$

where h and d is the dimension for the output and the feature encoding. We will provide more implementation details in the appendix.

4 Results

We demonstrate the applicability of our framework along three different axes: (*Expressivity*) For a number of different signal types, we demonstrate our ability to fit a given signal. We observe that our method enjoys faster convergence in terms of the number of iterations. (*Interpretability*) We visually demonstrate our learned PNF, whose outputs are localized in the frequency domain with both upper band-limit and lower-band limit. (*Decomposition*) Finally, we demonstrate the ability to control a PNF on the tasks of texture transfer and scale-space representation, based on our subband decomposition. For all presented experiments, full training details and additional qualitative and quantitative results are provided in the appendix.

Table 1: Image Fitting on the DIV2K dataset.

	PSNR	SSIM	# Params
RFF	28.72	0.834	0.26M
SIREN	29.22	0.866	0.26M
BACON	28.67	0.838	0.27M
BACON-L	29.44	0.871	0.27M
BACON-M	29.44	0.871	0.27M
PNF	**29.47**	**0.874**	0.28M

Table 2: 3D shape fitting. CD is Chamfer Distance ($\times 10^6$)

	CD	F-score	# Params
SIREN	9.00	99.76%	0.53M
BACON	2.60	99.84%	0.54M
BACON-L	2.60	99.85%	0.54M
BACON-M	2.61	99.85%	0.54M
PNF	**2.25**	**99.97%**	0.59M

Table 3: NeRF Fitting for 64^2 resolution. A comparison between PNF and bacon is shown for PSNR and SSIM at 300 and 500 epochs. as well as corresponding number of parameters used.

		1x	1/2x	1/4x	1/8x	Avg
300 epochs PSNR/SSIM	BACON	28.07/0.936	29.95/0.943	30.75/**0.939**	31.37/**0.927**	30.04/**0.936**
	PNF	**29.89/0.937**	**31.16/0.946**	**32.11**/0.934	**32.00**/0.920	**31.29**/0.934
500 epochs PSNR/SSIM	BACON	28.51/0.932	30.77/0.941	**31.74/0.940**	**32.10/0.928**	30.78/0.935
	PNF	**29.33/0.950**	**31.19/0.950**	31.48/0.936	31.37/0.926	**30.84/0.940**
# Params	BACON	0.54M	0.41M	0.27M	0.14M	0.34M
	PNF	0.46M	0.34M	0.23M	0.12M	0.29M

4.1 Expressivity

We demonstrate that a PNF is capable of representing signals of different modalities including images, 3D signed distance fields, and radiance fields. We compare our Fourier PNF with state-of-the-art neural field representations such as BACON [36], SIREN [60], and Random Fourier Features [63].

Images Following BACON [36], we train a PNF and the baselines to fit images from the DIV2K [2] dataset. During training, images are downsampled to 256^2. All networks are trained for 5000 iterations. At test time, we sample the fields at 512^2 and compare with the original resolution images.

Different from other networks, BACON is supervised with the training image in all output layers. For fair comparison, we also include two BACON variants, which are only supervised either at the last output layer ("BAC-L") or using the average of all output layers ("BAC-M"). We report the PSNR and Structural Similarity (SSIM) scores of these methods in Tab. 1. Fourier PNF improves on the performance of the previous state-of-the-arts.

In addition to its expressivity, PNF is also able to localize signals in different regions. Specifically, Fourier PNF decomposes an image into subbands in the frequency domain. For example, Fig. 4 shows that the output branches of the Fourier PNF correspond to specific frequency bands. Note that such decomposition forms for all layers while training is only performed for the last layer of the PNF. Such subband control is reminiscent of the traditional signal processing analogue of Laplacian Pyramids [9]. Note that each layer of the PNF is guaranteed to be both lower and upper frequency band limted, while this is not achievable by BACON. For comparison Fig. 4 shows the corresponding result of training BACON only for the last layer. In Sec. 4.2 and Sec. 4.3 we demonstrate how to leverage such fine-grain localization for texture transfer and scale space interpolation.

3D Signed Distance Field One advantage of neural fields is their ability to represent irregular data such as 3D point cloud or signed distance fields with high fidelity. We demonstrate that PNFs can represent 3D shapes via signed distance fields expressively. We follow BACON's experimental setting to fit a range of 3D shapes from the Stanford 3D scanning repository [64] (a slightly different normalization is used, see appendix for details). During training, we sample $10k$ oriented points from the ground truth surface and perturb them with noise to compute an estimate of SDF as ground truth. All models are trained with reconstruction loss for the same number of iterations. A quantitative comparison of the different methods is reported in Tab. 2. We observe that PNF achieves slightly

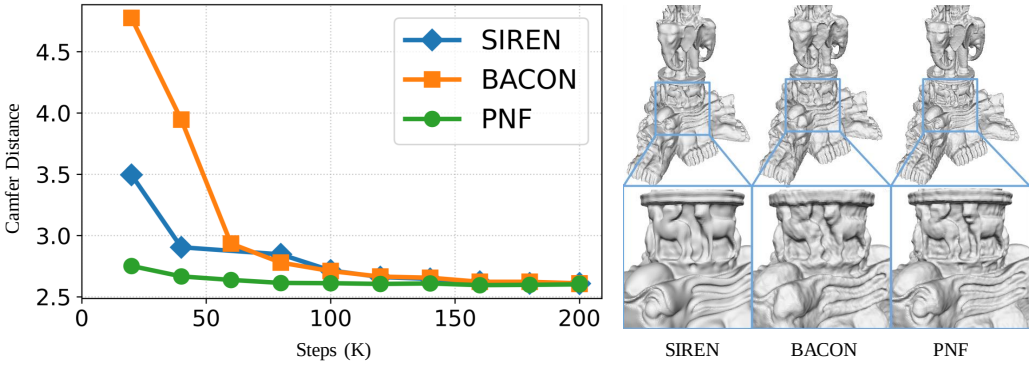

Figure 5: LHS: Convergence time in terms of number of steps/iterations for the *Thai Statue* model [64]. The x-axis shows the number of steps (K) and the y-axis shows the validation Chamfer Distance ($\times 10^6$). RHS: Qualitative comparison for the *Thai Statue* model [64].

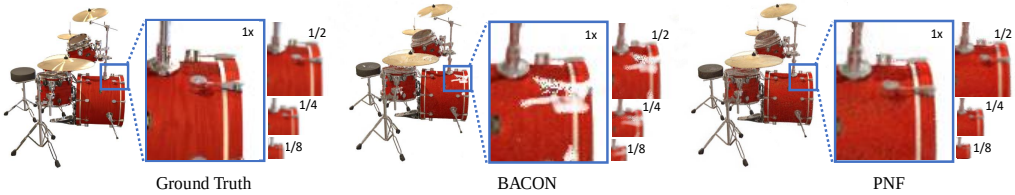

Figure 6: Qualitative comparison of neural radiance fields for the *Drums* scene given in [4] at the 300 epochs. The results suggests that our model can achieve better quality in early iterations.

higher fitting quality than other methods. Next, we investigate convergence behavior in Fig. 5 (Left) on the *Thai Statue* model [64]. Here, we observe that PNFs converge much more quickly than SIREN and BACON while achieving comparable quality. In Fig. 5 (Right) we provide a qualitative comparison of our result, BACON, and SIREN.

Neural Radiance Field In the context of neural radiance fields, we show that PNFs can represent signals in higher dimensions compactly and faithfully. We follow BACON's setting and train a PNF to model the radiance field of a set of Blender scenes [43], see appendix for details. Specifically, the PNF outputs a 4D vector of RGB and density values. These values are used by a volumetric renderer proposed of NeRF [43] to produce an image, which is supervised with reconstruction loss. At test time, we use the same volumetric renderer to produce images from different camera poses and evaluate them with the known ground truth images. The results, in comparison to BACON, are presented in Tab. 3. A Qualitative comparison is also given in Fig. 6. Similarly to SDF, we are able to match or improve BACON's performance using about 60% of the total number of iterations. This can be potentially attributed to PNF's ability to disentangle coarse signal from finer one; and so at higher layers, BACON needs to relearn low-frequency details in the image while the PNF does not.

Parameter Efficiency. One major advantage of using neural fields is that it can represent signal with high expressivity while remaining compact. In this section, we will demonstrate that PNF also retains this advantages. While choosing the hyperparameters for PNFs (e.g. hidden layer size) for the expressivity experiemnts, we make sure the PNF has a comparable number of parameters with the prior works. A comparison of the number of trainable parameters used for different model is included in Tab. 1-Tab. 3. We can see that PNFs are achieving comparable performance with the state-of-the-arts neural fields with roughly the same amount of trainable parameters.

Training and Inference Time While we observe that Fourier PNF can converge in fewer iterations, but due to the ensemble nature of Fourier PNF, each forward and backward pass of PNF requires longer time to evaluate. This leads to the question, can we actually achieve faster convergence in wall time? We profile the training and inference time of the image fitting experiment in Tab. 4 for the

Table 4: A comparison of the training and inference time for image fitting of a cameramen image.

	Time(s)/Step	Time(s) to 36 PSNR	Final PSNR	Final SSIM
BACON	0.16	177	37.45	97.33
PNF	0.64	96	37.45	97.44
SIREN	0.10	163	36.90	97.50
RFF	0.08	275	36.23	95.05

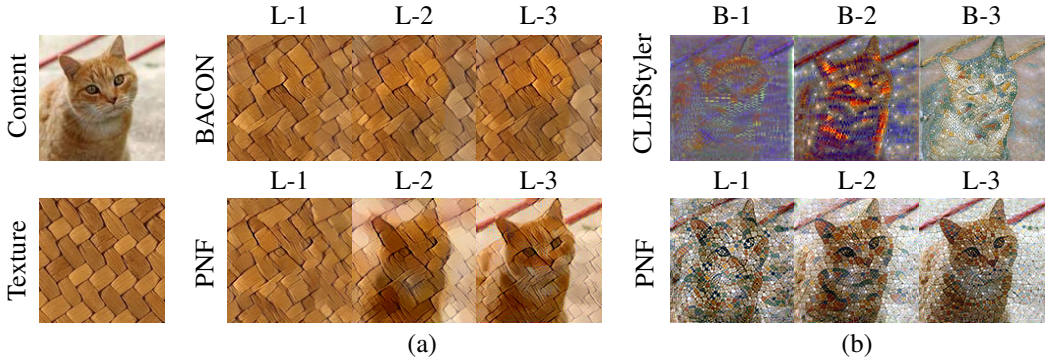

Figure 7: Texture transfer. (a). We optimize specific layers of the neural field. L-1 (layers 1-4), L-2 (layers 2-4), L-3 (layers 3-4). The texture contains stationary texture (in high frequency) and structured texture (in low frequency). As opposed to BACON, our method can isolate the stationary texture. (b) We consider the text based texture transfer objectives of CLIPStyler [34] for the cat content image. For PNF, we consider the text prompt "Mosaic" and apply the same layer-based optimization as in (a). For comparison, we apply CLIPStyler of "Low frequency mosaic" (B-1), "High frequency mosaic" (B-3) and "Mosaic" (B-3).

camera men image. The results show that it's possible for PNF to converge faster even in terms of clock time because PNFs can converge is drastically fewer number of steps.

4.2 Texture Transfer

Traditional approaches for image manipulation assume the input and output images are represented using a regular grid [19, 25]. Recent work has opted to use neural fields instead, for example allowing fine detailed texturing of 3D meshes [42]. Our formulation allows for an additional layer of control. In particular, due to our subband decompositionality, one can restrict the manipulation to particular subbands. The manipulation can then be driven by optimizing the PNF weights corresponding to those subbands, using various loss objectives.

We demonstrate this in the setting of texture transfer. We consider a Fourier PNF with four layers of the following frequency ranges (in Hz): (1) $[0, 8]$, (2) $[4, 16]$, (3) $[12, 32]$ and (4) $[28, 64]$. We consider a *content image* C of 128^2 resolution. In the first stage, we train a network to fit C as in Sec. 4.1. In the second stage, we optimize only the parameters of specifics layers. To optimize these parameters, we query the network on a 128^2 image grid producing image I. We then consider two sets of objectives: (a) Content and style loss objectives as given in [24]. (b), Text-based texture manipulation objectives as given by CLIPStyler [34]. See further details in the appendix.

In Fig. 7, we illustrate the result of optimizing layers 1 to 4, 2 to 4 or 3 to 4, corresponding to frequency ranges [0, 64], [4, 64] and [12, 64] respectively. We consider a texture which contains both stationary texture in the high frequency range and structured texture in the low frequency range. As opposed to BACON, our method can isolate the stationary texture. As can be seen in Fig. 7(b), for text based manipulation, our method generates texture that is in the correct frequency range. For comparison, we consider CLIPStyler [34], with text prompts of "Low frequency mosaic" (B-1), "High frequency mosaic" (B-3) and "Mosaic" (B-3), resulting in non-realistic texturing. As can be seen, simply specifying the frequency in the text does not result in a satisfactory result.

4.3 Scale-space Representation

In many visual computing applications such as volumetric rendering, one is usually required to aggregate information from the neural fields using operations such as Gaussian convolution [4]. It is useful to model signals as a function of both the spatial coordinates and the scale: $f(\mathbf{x}, \Sigma) = \mathbb{E}_{\mathbf{x}\sim\mathcal{N}(\mathbf{x},\Sigma)}[g(\mathbf{x})]$, where g is the assumed ground truth signal. Existing works try to approximate the scale-space by using a black-box MLP with intergrated positional encoding, which computes the analytical Gaussian convolved Fourier basis functions [4, 65]. While such approaches demonstrate success in volumetric rendering applications which requires integrating different scales, they depend on supervision in multiple scales, possibly because their ability to interpolate correctly between scales is hindered by the black-box MLP.

In this section we want to demonstrate the PNF's ability to better model this scale space with limited supervision. Suppose the signal of interest can be represented by Fouier bases as $g(\mathbf{x}) = \sum_n \alpha_n \exp(\omega_i^T \mathbf{x})$, then we know analytically the Gaussian convolved version should be $f(\mathbf{x}, \Sigma) = \sum_n \alpha_n \exp(\omega_i^T \Sigma \omega_i) \exp(i\omega_i^T \mathbf{x})$. If we assume that our Fourier PNF can learn the ground truth representation well, then one potential way to achieve this is setting $\gamma(\mathbf{x}, \Sigma)_n = \exp(-\frac{1}{2}\omega_n^T \Sigma \omega_n) \exp(-i\omega_n^T \mathbf{x})$. We also multiply the output of F_i (Eq. (71)) with a correction term addressing the error arises from missing the interference terms in the form of $\exp(-\frac{1}{2}\omega_i^T \Sigma \omega_j)$. We show in the appendix how to derive and approximate these missing terms using Fourier PNF.

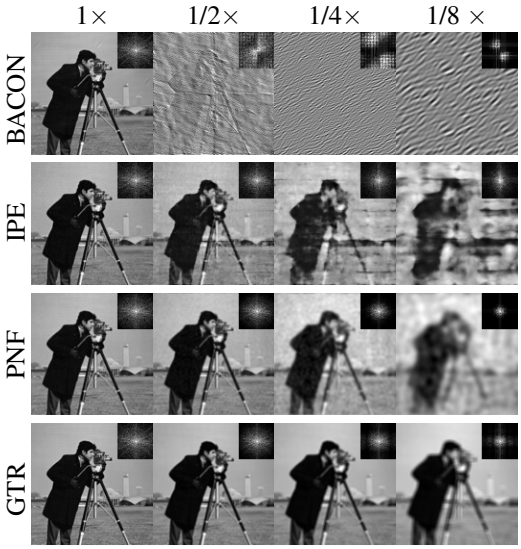

Figure 8: Scale-space representation. Networks are trained on full resolution image (1x) and test on the other reolutions. GTR is produced by applying Gaussian smoothing on (1x) image.

We test the effectiveness of Fourier PNF to learn the scale-space of a 2D image. In this application, the network is trained with the image signal in full resolution (finest scale). At test time, the network is asked to produce image with different scales and compared to the ground truth Gaussian smoothed image. We compared our method with IPE [4] as well as BACON with IPE as filter function. The results are shown in Fig. 8. Our model can represent a signal reasonably well when testing with a lower resolution while other methods degrade more quickly.

Limitations Currently, the activation memory of PNF scales linearly in the number of subbands, and so interpretability and decompositiality gained by PNF comes at a "cost" of a larger memory footprint. Further, it is also non trivial to tile higher dimensional space with controllabel subbands.

5 Conclusion

We proposed a novel class of neural fields (PNFs) that are compact and easy to optimize and enjoy the interpretability and controllability of signal processing methods. We provided a theoretical framework to analyse the output of a PNF and design a Fourier PNF, whose outputs can be decomposed in a fine-grained manner in the frequency domain with user-specified lower and upper limits on the frequencies. We demonstrated that PNFs matches state-of-the-art performance for signal representation tasks. We then demonstrated the use of PNF's subband decomposition in the settings of texture transfer and scale-space representations. As future work, the ability to generalize our representation to represent multiple higher-dimensional signals (such as multiple images) can enable applications in recognition and generation, where one can leverage our decomposeable architecture to impose a prior or regularize specific subbands to improve generalization.

Acknowledgement This research was supported by the Pioneer Centre for AI, DNRF grant number P1. Guandao's PhD was supported in part by research gifts from Google, Intel, and Magic Leap. Experiments are supported in part by Google clouds platform and GPUs donated by NVIDIA.

References

[1] Edward H Adelson, Charles H Anderson, James R Bergen, Peter J Burt, and Joan M Ogden. Pyramid methods in image processing. *RCA engineer*, 29(6):33–41, 1984. 1

[2] Eirikur Agustsson and Radu Timofte. Ntire 2017 challenge on single image super-resolution: Dataset and study. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 126–135, 2017. 7, 28, 34

[3] Matan Atzmon and Yaron Lipman. SAL: Sign agnostic learning of shapes from raw data. In *Proc. CVPR*, 2020. 1, 2

[4] Jonathan T Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P Srinivasan. Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5855–5864, 2021. 2, 8, 10, 29, 33, 34

[5] Mark Boss, Raphael Braun, Varun Jampani, Jonathan T Barron, Ce Liu, and Hendrik Lensch. Nerd: Neural reflectance decomposition from image collections. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12684–12694, 2021. 2

[6] Mark Boss, Varun Jampani, Raphael Braun, Ce Liu, Jonathan Barron, and Hendrik Lensch. Neural-pil: Neural pre-integrated lighting for reflectance decomposition. *Advances in Neural Information Processing Systems*, 34, 2021. 2

[7] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000. 33

[8] D. M. Brink and G. R. Satchler. Angular momentum. Oxford University Press, 1962. 18

[9] Peter J Burt and Edward H Adelson. The laplacian pyramid as a compact image code. In *Readings in computer vision*, pages 671–679. Elsevier, 1987. 1, 2, 4, 7

[10] Yinbo Chen, Sifei Liu, and Xiaolong Wang. Learning continuous image representation with local implicit image function. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8628–8638, 2021. 1

[11] Moulik Choraria, Leello Tadesse Dadi, Grigorios Chrysos, Julien Mairal, and Volkan Cevher. The spectral bias of polynomial neural networks. *arXiv preprint arXiv:2202.13473*, 2022. 3

[12] Grigorios Chrysos, Stylianos Moschoglou, Yannis Panagakis, and Stefanos Zafeiriou. Polygan: High-order polynomial generators. *arXiv preprint arXiv:1908.06571*, 2019. 2, 3

[13] Grigorios Chrysos, Stylianos Moschoglou, Giorgos Bouritsas, Yannis Panagakis, Jiankang Deng, and Stefanos Zafeiriou. $\pi-$nets: Deep polynomial neural networks. In *CVPR*, 2020. 17

[14] Grigorios Chrysos, Markos Georgopoulos, and Yannis Panagakis. Conditional generation using polynomial expansions. *Advances in Neural Information Processing Systems*, 34:28390–28404, 2021. 3

[15] Grigorios Chrysos, Markos Georgopoulos, Razvan Pascanu, and Volkan Cevher. Tutorial on high-degree polynomial networks. CVPR 2022 Tutorial, 2022. URL https://www.slideshare.net/GrigorisChrysos/tutorial-on-polynomial-networks-at-cvpr22. 2, 3

[16] Grigorios G Chrysos, Stylianos Moschoglou, Giorgos Bouritsas, Yannis Panagakis, Jiankang Deng, and Stefanos Zafeiriou. P-nets: Deep polynomial neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7325–7335, 2020. 3

[17] Thomas Davies, Derek Nowrouzezahrai, and Alec Jacobson. On the effectiveness of weight-encoded neural implicit 3D shapes. *arXiv preprint arXiv:2009.09808*, 2020. 2, 29

[18] Emilien Dupont, Adam Goliński, Milad Alizadeh, Yee Whye Teh, and Arnaud Doucet. Coin: Compression with implicit neural representations. *arXiv preprint arXiv:2103.03123*, 2021. 1

[19] Alexei A Efros and William T Freeman. Image quilting for texture synthesis and transfer. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 341–346, 2001. 9, 30

[20] S. M. 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, 2

[21] Rizal Fathony, Anit Kumar Sahu, Devin Willmott, and J Zico Kolter. Multiplicative filter networks. In *Proc. ICLR*, 2020. 3, 17

[22] Rizal Fathony, Anit Kumar Sahu, Devin Willmott, and J. Zico Kolter. Multiplicative filter networks. In *ICLR*, 2021. 2, 3, 17, 18, 22, 26, 32

[23] Stephan J Garbin, Marek Kowalski, Matthew Johnson, Jamie Shotton, and Julien Valentin. FastNeRF: High-fidelity neural rendering at 200fps. In *Proc. ICCV*, 2021. 2

[24] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. A neural algorithm of artistic style. *arXiv preprint arXiv:1508.06576*, 2015. 9, 30, 32

[25] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2414–2423, 2016. 9

[26] Markos Georgopoulos, Grigorios Chrysos, Maja Pantic, and Yannis Panagakis. Multilinear latent conditioning for generating unseen attribute combinations. *arXiv preprint arXiv:2009.04075*, 2020. 3

[27] Amos Gropp, Lior Yariv, Niv Haim, Matan Atzmon, and Yaron Lipman. Implicit geometric regularization for learning shapes. In *Proc. ICML*, 2020. 2

[28] Laura K. Gross and O'Malley Jr. Robert E. Essential mathematical methods for physicists. SIAM Review, 2005. 18

[29] Geoffrey E Hinton and Kevin J Lang. Shape recognition and illusory conjunctions. In *IJCAI*, pages 252–259, 1985. 3

[30] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7132–7141, 2018. 3

[31] Chiyu Jiang, Avneesh Sud, Ameesh Makadia, Jingwei Huang, Matthias Nießner, and Thomas Funkhouser. Local implicit grid representations for 3D scenes. In *Proc. CVPR*, 2020. 1, 2

[32] Yue Jiang, Dantong Ji, Zhizhong Han, and Matthias Zwicker. SDFDiff: Differentiable rendering of signed distance fields for 3D shape optimization. In *Proc. CVPR*, 2020. 2

[33] Joe Kileel, Matthew Trager, and Joan Bruna. On the expressive power of deep polynomial neural networks. *Advances in neural information processing systems*, 32, 2019. 3

[34] Gihyun Kwon and Jong Chul Ye. Clipstyler: Image style transfer with a single text condition. *arXiv preprint arXiv:2112.00374*, 2021. 9, 30, 32

[35] David B. Lindell, Julien N. P. Martel, and Gordon Wetzstein. AutoInt: Automatic integration for fast neural volume rendering. In *Proc. CVPR*, 2021. 2

[36] David B. Lindell, Dave Van Veen, Jeong Joon Park, and Gordon Wetzstein. Bacon: Band-limited coordinate networks for multiscale scene representation. *CVPR*, 2022. 2, 3, 7, 17, 22, 27, 28, 29, 30

[37] Lingjie Liu, Jiatao Gu, Kyaw Zaw Lin, Tat-Seng Chua, and Christian Theobalt. Neural sparse voxel fields. In *NeurIPS*, 2020. 2

[38] Shaohui Liu, Yinda Zhang, Songyou Peng, Boxin Shi, Marc Pollefeys, and Zhaopeng Cui. DIST: Rendering deep implicit signed distance function with differentiable sphere tracing. In *Proc. CVPR*, 2020. 2

[39] 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. *ACM Trans. Graph. (SIGGRAPH)*, 40(4), 2021. 1

[40] Ricardo Martin-Brualla, Noha Radwan, Mehdi S. M. Sajjadi, Jonathan T. Barron, Alexey Dosovitskiy, and Daniel Duckworth. NeRF in the Wild: Neural Radiance Fields for Unconstrained Photo Collections. In *Proc. CVPR*, 2021. 2

[41] Mateusz Michalkiewicz, Jhony K. Pontes, Dominic Jack, Mahsa Baktashmotlagh, and Anders Eriksson. Implicit surface representations as layers in neural networks. In *Proc. ICCV*, 2019. 2

[42] Oscar Michel, Roi Bar-On, Richard Liu, Sagie Benaim, and Rana Hanocka. Text2mesh: Text-driven neural stylization for meshes. *arXiv preprint arXiv:2112.03221*, 2021. 9

[43] 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 *Proc. ECCV*, 2020. 1, 2, 8, 28, 29

[44] 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*, 40(4), 2021. 2

[45] Michael Niemeyer, Lars Mescheder, Michael Oechsle, and Andreas Geiger. Differentiable volumetric rendering: Learning implicit 3D representations without 3D supervision. In *Proc. CVPR*, 2020. 1, 2

[46] Harry Nyquist. Certain topics in telegraph transmission theory. *Transactions of the American Institute of Electrical Engineers*, 47(2):617–644, 1928. 5

[47] Michael Oechsle, Lars Mescheder, Michael Niemeyer, Thilo Strauss, and Andreas Geiger. Texture fields: Learning texture representations in function space. In *Proc. ICCV*, 2019. 1, 2

[48] Michael Oechsle, Songyou Peng, and Andreas Geiger. UNISURF: Unifying neural implicit surfaces and radiance fields for multi-view reconstruction. In *Proc. ICCV*, 2021. 2

[49] Sophocles J Orfanidis. *Introduction to signal processing*. Prentice-Hall, Inc., 1995. 2

[50] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. DeepSDF: Learning continuous signed distance functions for shape representation. In *Proc. CVPR*, 2019. 1, 2

[51] Songyou Peng, Michael Niemeyer, Lars Mescheder, Marc Pollefeys, and Andreas Geiger. Convolutional occupancy networks. In *Proc. ECCV*, 2020. 1, 2

[52] Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. D-NeRF: Neural radiance fields for dynamic scenes. In *Proc. CVPR*, 2021. 2

[53] Maithra Raghu, Ben Poole, Jon Kleinberg, Surya Ganguli, and Jascha Sohl-Dickstein. Survey of expressivity in deep neural networks. *arXiv preprint arXiv:1611.08083*, 2016. 3

[54] Shunsuke Saito, Zeng Huang, Ryota Natsume, Shigeo Morishima, Angjoo Kanazawa, and Hao Li. PIFu: Pixel-aligned implicit function for high-resolution clothed human digitization. In *Proc. ICCV*, 2019. 2

[55] Vishwanath Saragadam, Jasper Tan, Guha Balakrishnan, Richard G Baraniuk, and Ashok Veeraraghavan. Miner: Multiscale implicit neural representations. *arXiv preprint arXiv:2202.03532*, 2022. 1

[56] Terrence J Sejnowski. Higher-order boltzmann machines. In *AIP Conference Proceedings*, volume 151, pages 398–403. American Institute of Physics, 1986. 3

[57] Eero P Simoncelli and Edward H Adelson. Subband transforms. In *Subband image coding*, pages 143–192. Springer, 1991. 2

[58] Eero P Simoncelli and William T Freeman. The steerable pyramid: A flexible architecture for multi-scale derivative computation. In *Proceedings., International Conference on Image Processing*, volume 3, pages 444–447. IEEE, 1995. 1, 2, 4

[59] Vincent Sitzmann, Michael Zollhöfer, and Gordon Wetzstein. Scene representation networks: Continuous 3D-structure-aware neural scene representations. In *Proc. NeurIPS*, 2019. 1, 2

[60] Vincent Sitzmann, Julien N. P. Martel, Alexander W. Bergman, David B. Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. In *Proc. NeurIPS*, 2020. 1, 2, 7, 28, 29

[61] Pratul P. Srinivasan, Boyang Deng, Xiuming Zhang, Matthew Tancik, Ben Mildenhall, and Jonathan T. Barron. NeRV: Neural reflectance and visibility fields for relighting and view synthesis. In *CVPR*, 2021. 2

[62] Kenneth O Stanley. Compositional pattern producing networks: A novel abstraction of development. *Genetic programming and evolvable machines*, 8(2):131–162, 2007. 2

[63] Matthew Tancik, Pratul P. Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan T. Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. In *Proc. NeurIPS*, 2020. 1, 2, 7, 28

[64] Stanford University. 3D scanning repository. http://graphics.stanford.edu/data/3Dscanrep/, 1996. 7, 8

[65] Dor Verbin, Peter Hedman, Ben Mildenhall, Todd E. Zickler, Jonathan T. Barron, and Pratul P. Srinivasan. Ref-nerf: Structured view-dependent appearance for neural radiance fields. *ArXiv*, abs/2112.03907, 2021. 10

[66] Martin Vetterli and Jelena Kovacevic. Wavelets and subband coding. In *Prentice Hall Signal Processing Series*, 1995. 4

[67] Peng Wang, Lingjie Liu, Yuan Liu, Christian Theobalt, Taku Komura, and Wenping Wang. NeuS: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. *Proc. NeurIPS*, 2021. 2

[68] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7794–7803, 2018. 3

[69] Yongtao Wu, Grigorios G Chrysos, and Volkan Cevher. Adversarial audio synthesis with complex-valued polynomial networks. *arXiv preprint arXiv:2206.06811*, 2022. 3

[70] Yiheng Xie, Towaki Takikawa, Shunsuke Saito, Or Litany, Shiqin Yan, Numair Khan, Federico Tombari, James Tompkin, Vincent Sitzmann, and Srinath Sridhar. Neural fields in visual computing and beyond. In *Computer Graphics Forum*, volume 41, pages 641–676. Wiley Online Library, 2022. 2

[71] Guandao Yang, Xun Huang, Zekun Hao, Ming-Yu Liu, Serge Belongie, and Bharath Hariharan. Pointflow: 3d point cloud generation with continuous normalizing flows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4541–4550, 2019. 2

[72] Lior Yariv, Yoni Kasten, Dror Moran, Meirav Galun, Matan Atzmon, Ronen Basri, and Yaron Lipman. Multiview neural surface reconstruction by disentangling geometry and appearance. In *Proc. NeurIPS*, 2020. 1, 2

[73] Lior Yariv, Jiatao Gu, Yoni Kasten, and Yaron Lipman. Volume rendering of neural implicit surfaces. In *Proc. NeurIPS*, 2021. 2

[74] Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. pixelNeRF: Neural radiance fields from one or few images. In *Proc. CVPR*, 2021. 2

[75] Kai Zhang, Wangmeng Zuo, Yunjin Chen, Deyu Meng, and Lei Zhang. Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising. *IEEE transactions on image processing*, 26(7):3142–3155, 2017. 33, 34

[76] Kai Zhang, Gernot Riegler, Noah Snavely, and Vladlen Koltun. Nerf++: Analyzing and improving neural radiance fields. *arXiv preprint arXiv:2010.07492*, 2020. 2

[77] Jianqiao Zheng, Sameera Ramasinghe, and Simon Lucey. Rethinking positional encoding. *arXiv preprint arXiv:2107.02561*, 2021. 2

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] All contributions accounted for in Sec. 3 and Sec. 4.
   (b) Did you describe the limitations of your work? [Yes] See end of Sec. 4.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See appendix.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] See Sec. 3 and appendix
   (b) Did you include complete proofs of all theoretical results? [Yes] See Sec. 3 and appendix
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] Data used is publicly available and cited. Code and instructions for experimental results will be released publicly.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] Full details in Sec. 4 and appendix
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] See appendix.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See appendix.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [Yes] See appendix.
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] Code will be released publicly.
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [Yes] See appendix.
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [Yes] See appendix.
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Theory

In this section, we will provide a detail derivation of the theories we used for building and analyzing PNF. We first study the definition of PNF in Appendix A.1. Then we prove the properties that PNF is linear sums of basis for different basis in Appendix A.2 and Appendix A.3. In Appendix A.4, we studies how to organize subset of basis, or subbands, in a controllable way to produce subband-limited PNFs. Finally, we show several different instantiation of PNFs in Appendix A.5.

A.1 Definition

Definition A.1 (PNF)**.** *Let $\mathcal{B}$ be a basis for the vector space of functions for $\mathbb{R}^n \to \mathbb{R}$. A* Polynomial neural field of basis *$\mathcal{B}$ is a neural network $f = g_L \circ \cdots \circ g_1 \circ \gamma$, where $\forall i, g_i$ are finite degree multivariate polynomials, and $\gamma : \mathbb{R}^n \to \mathbb{R}^d$ is a d-dimensional feature encoding using basis $\mathcal{B}$: $\gamma(x) = [\gamma_1(x), \ldots, \gamma_d(x)]^d, \gamma_i \in \mathcal{B}, \forall i$.*

We will show how this definition has included both Π-Net [13], MFN [21], and BACON [36].

Π-Net. Here we will set basis $\mathcal{B} = \{x^n\}_{n \geq 0}$. Then g_i can be set according to different factorization mentioned in Section 3.1 and 3.2.

MFN. [21] studied two types of MFNs - Fourier and Gabor MFN. For Fourier MFN that that takes $\mathbb{R}^d$ as input, we will set basis to be $\mathcal{B} = \{\sin(\omega^T x)\}_{\omega \in \mathbb{R}^d}$. Assume there are L_{mfn} layers of the multiplicative filter networks and each layer has hidden dimension of h. We will set $L = L_{mfn} + 2$ and define g_i, γ in the following manner:

$$\gamma \in \mathbb{R}^d \to \mathbb{R}^{L_{mfn}h}, \tag{7}$$

$$g_1(\gamma(x)) = [M_1\gamma(x), \gamma(x)] \in \mathbb{R}^{(L_{mfn}+1)h} \tag{8}$$

$$g_i([z, \gamma(x)]) = [(W_i z + b_i) \odot (M_i\gamma(x)), \gamma(x)], \quad \forall\, 2 \leq i \leq 2 + L_{mfn} \tag{9}$$

$$g_{L_{mfn}+2}([z, \gamma(x)]) = W_{out}x + b_{out}, \tag{10}$$

where $M_i \in \mathbb{R}^{L_{mfn}h \times h}$ selects the $(i-1)h$ to ih basis by setting each row of M_i to be an one-hot vector and only the $((i-1)h)^{th}$ to $(ih)^{th}$ columes are non-zeros. Similarly, for Gabor MFN, we use the same definition of g, but switch γ to sample from $\mathcal{B} = \{\exp(\gamma \|x - \mu\|^2) \sin(\omega^T x)\}_{(\gamma, \mu, \omega \in \mathbb{R}^d)}$.

BACON. The way to instantiate BACON will be similar to MFN. Basically each intermediate output layer of BACON is a MFN with γ to be sampled from specific subbands.

The definition of PNF is very general such that it not only include prior works but also allow potential design of new architectures with different network topology and differnet basis. Our design of Fourier PNF will leverage fourier basis with a modified network architecture. In this appendix, we will introduce several more variants of the PNFs with different architectures and basis choices.

A.2 Basis function

Definition A.2 (Span of Basis is Closed Under Multiplication)**.** *We call a basis $\mathcal{B}$'s span closed under multiplication if : $\forall b_1, b_2 \in \mathcal{B}, b_1(x)b_2(x) = \sum_{i \in I} a_i b_i(x), |I| < \infty$.*

Note that this is the same requirement as Definition 1 in the appendix of MFN paper [22]. We will extend the analysis of MFN in several ways. First, we will show that several commonly used basis functions satisfies Definition A.2.

Lemma 4 (Fourier Basis)**.** *Assume the Fourier basis of functions $\mathbb{R}^d \to \mathbb{R}$ takes the form of $\mathcal{B}_{Fourier} = \{b_\omega = \exp(i\omega^T x) | \omega \in \mathbb{R}^d\}$. Then $\mathcal{B}_{Fourier}$'s span is closed under multiplication.*

Proof. It's enough to show that the multiplication of two Fourier basis function is still a Fourier basis funciton: $\exp(i\omega_1^T x)\exp(i\omega_2^T x) = \exp(i(\omega_1 + \omega_2)^T x) \in \mathcal{B}_{Fourier}$. □

Lemma 5 (RBF)**.** *Assume the Radial basis functions for real-value functions $\mathbb{R}^d \to \mathbb{R}$ takes the form of $\mathcal{B}_{RBF} = \{b_{\gamma,\mu} = \exp(-\frac{1}{2}\gamma \|x - \mu\|^2)$. Then $\mathcal{B}_{RBF}$'s span is closed under multiplication.*

Proof. Similar to Fourier Basis, we will show that the multiplication of two RBF functions is still in $\mathcal{B}_{RBF}$. This is shown in Equation 24 of Supplementary of MFN [22]:

$$\exp\left(-\frac{1}{2}\gamma_1 \|x-\mu_1\|^2\right)\exp\left(-\frac{1}{2}\gamma_2 \|x-\mu_2\|^2\right) \tag{11}$$

$$= \exp\left(\frac{-\gamma_1\gamma_2\|\mu_1-\mu_2\|}{2(\gamma_1+\gamma_2)}\right)\exp\left(-\frac{1}{2}(\gamma_1+\gamma_2)\left\|x-\frac{\gamma_1\mu_1+\gamma_2\mu_2}{\gamma_1+\gamma_2}\right\|^2\right) \tag{12}$$

$$= c(\gamma_1,\gamma_2,\mu_1,\mu_2)\exp\left(-\frac{1}{2}\gamma' \|x-\mu'\|^2\right), \tag{13}$$

where $c(\gamma_1,\gamma_2,\mu_1,\mu_2) = \exp\left(\frac{-\gamma_1\gamma_2\|\mu_1-\mu_2\|}{2(\gamma_1+\gamma_2)}\right)$, $\mu' = \frac{\gamma_1\mu_1+\gamma_2\mu_2}{\gamma_1+\gamma_2}$, and $\gamma' = \gamma_1+\gamma_2$. □

Lemma 6 (Gabor Basis)**.** *Assume the Gabor basis of functions $\mathbb{R}^d \to \mathbb{R}$ takes the form of $\mathcal{B}_{Gabor} = \{b_{\gamma,\mu,\omega} = \exp(-\frac{1}{2}\gamma\|x-\mu\|^2)\exp(i\omega^T x) | \omega \in \mathbb{R}^d, \mu \in \mathbb{R}^d, \gamma \geq 0\}$. Then $\mathcal{B}_{Gabor}$'s span is closed under multiplication.*

Proof. Using Eq. (13), we can compute the the multiplication of two Gabor basis functions:

$$b_{\gamma_1,\mu_1,\omega_1} b_{\gamma_2,\mu_2,\omega_2} = c(\gamma_1,\gamma_2,\mu_1,\mu_2)\exp\left(-\frac{1}{2}\gamma' \|x-\mu'\|^2\right)\exp(i\omega'^T x) \propto b_{\gamma',\mu',\omega'} \in \mathcal{B}_{Gabor}, \tag{14}$$

where $\omega' = \omega_1 + \omega_2$. The output of the multiplication is still a Gabor. □

The following Lemma will show that Definition A.2 can be extended to analyzing functions from different domain. We will show that for complex-value function that maps from a sphere, there is a basis function that satisfies Definition A.2.

Lemma 7 (Spherical Harmonics)**.** *We will consider the basis function for real functions that takes spherical coordinate (i.e. $S^2 \to \mathbb{C}$ where $S^2 = \{(\theta,\phi)|0 \leq \theta \leq \pi, 0 \leq \phi \leq 2\pi\}$. Moreover, we will consider Laplace's spherical harmonics as basis:*

$$\mathcal{B}_{SH} = \{Y_l^m(\theta,\phi) = e^{im\phi}P_l^m(\cos(\theta)) | 0 \leq l, l \in \mathbb{Z}, -l \leq m \leq l, m \in \mathbb{Z}\}, \tag{15}$$

where $P_l^m : [-1,1] \to \mathbb{R}$ is an associated Legendre polynomial. $\mathcal{B}_{SH}$ satisfies Definition A.2.

Proof. From the multiplication rule of Spherical Harmonics [8], we have:

$$Y_{l_1}^{m_1}(\theta,\phi)Y_{l_2}^{m_2}(\theta,\phi) \propto \sum_{l=0}^{\infty}\sum_{m=-c}^{c}(-1)^m\sqrt{2l+1}\begin{pmatrix} l_1 & l_2 & l \\ m_1 & m_2 & -m \end{pmatrix}\begin{pmatrix} l_1 & l_2 & l \\ 0 & 0 & 0 \end{pmatrix} Y_l^m(\theta,\phi), \tag{16}$$

where $\begin{pmatrix} j_1 & j_2 & j_3 \\ m_1 & m_2 & m_3 \end{pmatrix}$ denotes the $3j$-syombols. Now we need to show that the infinite sum contains only finite number of non-zero terms. By the selection rules of the $3j$-symbols [28], we know that $\begin{pmatrix} j_1 & j_2 & j_3 \\ m_1 & m_2 & m_3 \end{pmatrix}$ is zero if any of the following rules is not satisfies: 1) $|j_1 - j_2| \leq j_3 \leq j_1 + j_2$; and 2) $m_1 + m_2 + m_3 = 0$. This implies that for all terms $l \geq l_1 + l_2$, and $m \neq -(m_1+m_2)$, the term $\begin{pmatrix} l_1 & l_2 & l \\ m_1 & m_2 & -m \end{pmatrix} = 0$. With this said, the Eq. (16) can be written as a finite sum:

$$Y_{l_1}^{m_1}(\theta,\phi)Y_{l_2}^{m_2}(\theta,\phi) \propto \sum_{l=|l_1-l_2|}^{l_1+l_2}\begin{pmatrix} l_1 & l_2 & l \\ m_1 & m_2 & -(m_1+m_2) \end{pmatrix}\begin{pmatrix} l_1 & l_2 & l \\ 0 & 0 & 0 \end{pmatrix} Y_l^m(\theta,\phi). \tag{17}$$

□

There are several rules we can used to generate more basis that satisfies the properties. Here we will show one of them:

Theorem 8 (Basis-multiplication.)**.** *If $\mathcal{B}_1$ and $\mathcal{B}_2$ be the basis of $\mathbb{R}^d \to \mathbb{R}$ and satisfy Definition A.2, then $\mathcal{B}_{mult} = \{b_1 b_2 | b_1 \in \mathcal{B}_1, b_2 \in \mathcal{B}_2\}$ also satisfy Definition A.2.*

Proof. Let $b_1 b_2$ and $b_3 b_4$ from $\mathcal{B}_{mult}$. Assume $b_i = \sum_n a_{i,n} b_{k_{i,n}}$, where $i \in \{1, 2, 3, 4\}$, $k_{i,n}$ is an index into $\mathcal{B}_1$ if $i \in \{1, 2\}$ and $\mathcal{B}_2$ if $i \in \{3, 4\}$, and $a_{i,n}$ are coefficients. Then we have

$$(b_1 b_2)(b_3 b_4) = \left(\sum_n a_{1,n} b_{k_{1,n}}\right)\left(\sum_n a_{2,n} b_{k_{2,n}}\right)\left(\sum_n a_{3,n} b_{k_{3,n}}\right)\left(\sum_n a_{4,n} b_{k_{4,n}}\right) \tag{18}$$

$$= \left(\sum_{n,n'} a_{1,n} a_{3,n'} b_{k_{1,n}} b_{k_{3,n'}}\right)\left(\sum_{n,n'} a_{2,n} a_{4,n'} b_{k_{2,n}} b_{k_{4,n'}}\right). \tag{19}$$

Since both $b_{k_{1,n}}$ and $b_{k_{3,n'}}$ are in $\mathcal{B}_1$ and $\mathcal{B}_1$ satisfies Definition A.2, so we can assume the following (and similarly logics can be applied to $b_{k_{2,n}}$ and $b_{k_{4,n'}}$ for $\mathcal{B}_2$):

$$b_{k_{1,n}} b_{k_{3,n'}} = \sum_m c_{1,n,n',m} b_{l_{1,n,n',m}}, \quad b_{l_{1,n,n',m}} \in \mathcal{B}_1 \tag{20}$$

$$b_{k_{2,n}} b_{k_{4,n'}} = \sum_m d_{n,n',m} b_{l_{2,n,n',m}}, \quad b_{l_{2,n,n',m}} \in \mathcal{B}_2. \tag{21}$$

Plugging abovementioned equations into Eq. (19) we get

$$\sum_{n,n'} a_{1,n} a_{3,n'} b_{k_{1,n}} b_{k_{3,n'}} = \sum_{n,n'} a_{1,n} a_{3,n'} \left(\sum_m c_{1,n,n',m} b_{l_{1,n,n',m}}\right) \tag{22}$$

$$= \sum_{n,n',m} a_{1,n} a_{3,n'} c_{1,n,n',m} b_{l_{1,n,n',m}} = \sum_{n,n',m} \tilde{c}_{1,n,n',m} b_{l_{1,n,n',m}}, \tag{23}$$

for $\tilde{c}_{1,n,n',m} = a_{1,n} a_{3,n'} c_{1,n,n',m}$. Similarly, we have

$$\sum_{n,n'} a_{2,n} a_{4,n'} b_{k_{2,n}} b_{k_{4,n'}} = \sum_{n,n'} a_{2,n} a_{4,n'} \left(\sum_m c_{2,n,n',m} b_{l_{2,n,n',m}}\right) \tag{24}$$

$$= \sum_{n,n',m} a_{2,n} a_{4,n'} c_{2,n,n',m} b_{l_{2,n,n',m}} = \sum_{n,n',m} \tilde{c}_{2,n,n',m} b_{l_{2,n,n',m}}, \tag{25}$$

for $\tilde{c}_{2,n,n',m} = a_{2,n} a_{4,n'} c_{2,n,n',m}$. Finally we can put these two together:

$$(b_1 b_2)(b_3 b_4) = \left(\sum_{n,n',m} \tilde{c}_{1,n,n',m} b_{l_{1,n,n',m}}\right)\left(\sum_{n,n',m} \tilde{c}_{2,n,n',m} b_{l_{2,n,n',m}}\right) \tag{26}$$

$$= \sum_{n,n',m,n'',n''',m'} \tilde{c}_{1,n,n',m} \tilde{c}_{2,n'',n''',m'} b_{l_{1,n,n',m}} b_{l_{2,n'',n''',m'}}, \tag{27}$$

which is a linear combination of basis $\mathcal{B}_{mult}$ since $b_{l_{1,n,n',m}} \in \mathcal{B}_1$ and $b_{l_{1,n'',n''',m'}} \in \mathcal{B}_2$. □

A.3 PNF as Linear Sum of Basis

In this section, we will show that PNF can be expressed as linear sum of basis if the basis function satisfies Definition A.2.

Lemma 9 (Power-product of Basis)**.** *If $\mathcal{B}$ satisfies Definition A.2, then power-products of the form $\prod_{n=1}^{N} b_n^{\alpha_n}$, $0 \le \alpha_n < \infty$ is a linear sum of the basis $\mathcal{B}$.*

Proof. We will show by induction on the degree of the power-product: $d = \sum_n \alpha_n$.

Base cases. If $d = 0$, then $\prod_{n=1}^{N} b_n^{\alpha_n} = 1 \in \mathcal{B}$ since $\mathcal{B}$ is a basis function of $\mathbb{R}^d \to \mathbb{R}$.

Inductive case. The inductive hypothesis is: assume that for some degree d such that $d \ge 0$, all finite power-product of degree d are linear sums of basis $\mathcal{B}$. We want to show that all finite power-product

of degree $d+1$ is also linear sums of basis $\mathcal{B}$. Condider a power-product of degree $d+1$ in the form of $\prod_{n=1}^{N} b_n^{\alpha_n}$ and assume without lost of generality that $\alpha_1 \geq 1$. Then we have

$$\prod_{n=1}^{N} b_n^{\alpha_n} = b_1 \left(b_1^{(\alpha_n - 1)} \prod_{n=2}^{N} b_n^{\alpha_n} \right) = b_1 p, \tag{28}$$

where $p = b_1^{(\alpha_n-1)} \prod_{n=2}^{N} b_n^{\alpha_n}$. It's easy to see that the degree of p is d, so p can be written as a linear sum of $\mathcal{B}$: $\sum_m a_m b_m$. With that we have:

$$\prod_{n=1}^{N} b_n^{\alpha_n} = b_1 \sum_m a_m b_m = \sum_m a_m b_1 b_m, \tag{29}$$

where $b_i \in \mathcal{B}$. Since $\mathcal{B}$ satisfies Definition A.2, so $b_1 b_m$ can be written as a linear sum of basis $\mathcal{B}$: $\sum_j c_j b_j$. Putting this in Eq. (29) gives a linear sum of basis $\mathcal{B}$. □

Theorem 10. *Let F be a PNF with basis $\mathcal{B}$. $\forall b_1, b_2 \in \mathcal{B}, b_1(x) b_2(x) = \sum_{i \in I} a_i b_i(x), |I| < \infty$, then the output of F is a finite linear sum of the basis functions from $\mathcal{B}$.*

Proof. Let $F = g_L \circ \cdots \circ g_1 \circ \gamma$, with g_i being a multivariate polynomial. Let $z_0 = \gamma$. Let $z_i = g_i \circ \cdots \circ g_1 \circ \gamma$ for $i \geq 1$. Let $z_i[j]$ be the jth dimensional value of z_i. We will show by induction that for all i, $z_i[j]$ is a linear sum of basis functions from $\mathcal{B}$ for all j.

Base case : $i = 0$. $z_i[j] = \gamma_j(x) \in \mathcal{B}$ by definition of γ in Definition A.1.

Inductive case. The inductive hypothesis is: for $k \geq 1$, if $z_k[j]$ is linear sum of $\mathcal{B}$ for all j, then $z_{k+1}[l]$ is linear sums of $\mathcal{B}$ for all l.

By definition of z, we know that $z_{k+1} = g_k(z_k)$, where g_k is a multivariate polynomial of finite degree d. With that said, we can assume $z_{k+1}[l] \in \mathbb{R}$ is a linear sum of power-product terms in the form of $\prod_j z_k[i]^{\alpha_{lj}}$, where $\alpha_{lj} \geq 0$ and $\sum_l \alpha_l \leq d$. It's sufficient to show that each of this term is linear sum of the basis function $\mathcal{B}$.

By the inductive hypothesis, we can assume that $z_k[j] = \sum_n \beta_{jn} b_{jn}$, with $b_{jn} \in \mathcal{B}$. Then we have:

$$\prod_j z_k[i]^{\alpha_{lj}} = \prod_j \left(\sum_n \beta_{jn} b_{jn} \right)^{\alpha_{lj}} = \sum_m a_m \prod_n b_n^{\hat{\alpha}_{nm}}, \tag{30}$$

for some $a_m \in \mathbb{R}$, and $\hat{\alpha}_{nm} \geq 0$ and $\sum_m \hat{\alpha}_{nm} \leq d$. By Theorem 9, the terms $\prod_n b_n^{\hat{\alpha}_{nm}}$ are linear sums of $\mathcal{B}$. Then Eq. (30) is linear sums of linear sums of $\mathcal{B}$, which will be linear sums of $\mathcal{B}$. □

A.4 Controllable Sets of Subbands

In this section, we will develop theories to build controllable sets of subbands and use that to design PNFs. In this section, we will use definition that a subband is a subset of the basis function.

Definition A.3 (Subband)**.** *A subband S is a subset of $\mathcal{B}$.*

Definition A.4 (Subband limited PNF)**.** *A PNF F of basis $\mathcal{B}$ is limited by subband $S \subset \mathcal{B}$ if each dimension of the output (i.e. F_i) is in the span of S.*

One naive way to construct a subband limited PNF is to restrict γ to take only basis functions from the subband: $\gamma_i \in S$, and then restricted no multiplication in the layers g_i (since multiplication is the only operation that can change the composition of the basis functions used in the linear sum).

But this will simply create a very shallow network as the composition of linear layers amounts to only one linear layer. As a result, we will need a very wide network in order to achieve expressivity. This means the number of basis function we used will grow linear with the number of network parameters. When it requires exponential number of basis functions to approximate a signal well, then we are not capable of achieving it compactly, throwing away a key virtual of neural fields.

In order to achieve compactness, we want to be able to generate a lots of basis, all of with within the subband of interest, without instantiating a lot of network parameters. One way to achieve this

is to use function composition with non-linearity. In our case, since the functions are restricted to be polynomials, then the only non-linearity we can use is multiplication. But multiplication can potentially create basis functions outside of the subband as Definition A.2 does not restrict the linear sum to be within certain subband. As a result, we need to study how multiplication transform the subband. More specifically, we will first define a set of subbands, within which multiplication in function space can translate in someway to an operation of subbands:

Definition A.5 (PNF-controllable Set of Subbands)**.** *$\mathcal{S} = \{S_\theta | S_\theta \subset \mathcal{B}\}_\theta$ is a PNF-Controllable Set of Subbands for basis $\mathcal{B}$ if (1) $S_{\theta_1} \cup S_{\theta_2} \in \mathcal{S}$ and (2) there exists a binary function $\otimes : \mathcal{S} \times \mathcal{S} \to \mathcal{S}$ such that if $b_1 \in S_{\theta_1}, b_2 \in S_{\theta_2} \implies b_1 b_2 = \sum_n a_n b_n, b_n \in S_{\theta_1} \otimes S_{\theta_2}$ for some coefficients $a_n \in \mathbb{R}$ (i.e. $b_1 b_2$ is in the span of $S_{\theta_1} \otimes S_{\theta_2}$).*

Theorem 11. *Let $\mathcal{S}$ be a PNF-controllable set of subbands of basis $\mathcal{B}$ with its corresponding binary function $\otimes$. Suppose F and G are polynomial neural fields of basis $\mathcal{B}$ that maps $\mathbb{R}^n$ to $\mathbb{R}^{m_1}$ and $\mathbb{R}^{m_2}$ respectively. Furthermore, suppose F and G are subband limited by $S_1 \in \mathcal{S}$ and $S_2 \in \mathcal{S}$. Then we have the following:*

1. *$A(x) = \mathbf{w}_1^T F(x) + \mathbf{w}_2 G(x)$ is a PNF of $\mathcal{B}$ limited by subband $S_1 \cup S_2$ with $\mathbf{w}_1 \in \mathbb{R}^{m_1}$ and $\mathbf{w}_2 \in \mathbb{R}^{m_2}$[3]; and*
2. *$M(x) = F(x)^T W G(x)$ is a PNF of $\mathcal{B}$ limited by subband $S_1 \otimes S_2$ with $W \in \mathbb{R}^{m_1 \times m_2}$.*

Proof. By Definition A.4, we can assume that

$$\forall 1 \le i \le m_1, F_i(x) = \sum_j a_{f,j} b_{f,j}(x), \quad b_{f,j} \in S_1 \tag{31}$$

$$\forall 1 \le i \le m_2, G_i(x) = \sum_j a_{g,j} b_{g,j}(x), \quad b_{g,j} \in S_2. \tag{32}$$

Then we can compute $A(x)$:

$$A(x) = \sum_{i=1}^{m_1} \mathbf{w}_1[i] F_i(x) + \sum_{i=1}^{m_2} \mathbf{w}_2[i] G_i(x) \tag{33}$$

$$= \sum_{i=1}^{m_1} \mathbf{w}_1[i] \left(\sum_j a_{f,j} b_{f,j}(x) \right) + \sum_{i=1}^{m_2} \mathbf{w}_2[i] \left(\sum_j a_{g,j} b_{g,j}(x) \right) \tag{34}$$

$$= \underbrace{\left(\sum_{i=1}^{m_1} \sum_j (\mathbf{w}_1[i] a_{f,j}) b_{f,j}(x) \right)}_{\text{Subband limited by } S_1} + \underbrace{\left(\sum_{i=1}^{m_2} \sum_j (\mathbf{w}_2[i] a_{g,j}) b_{g,j}(x) \right)}_{\text{Subband limited by } S_2}. \tag{35}$$

It's easy to see that Eq. (35) is subband limited by $S_1 \cup S_2$.

Similarly, we can apply the computation to $M(x)$:

$$M(x) = \sum_{i,j \ge 1}^{i,j \le m_1, m_2} W[i,j] F_i(x) G_j(x) \tag{36}$$

$$= \sum_{i,j \ge 1}^{i,j \le m_1, m_2} W[i,j] \left(\sum_k a_{f,k} b_{f,k}(x) \right) \left(\sum_k a_{g,k} b_{g,k}(x) \right) \tag{37}$$

$$= \sum_{i,j \ge 1}^{i,j \le m_1, m_2} W[i,j] \left(\sum_{k,k'} a_{f,k} a_{g,k'} b_{f,k}(x) b_{g,k'}(x) \right) \tag{38}$$

$$= \sum_{i,j \ge 1, k, k'}^{i,j \le m_1, m_2} (W[i,j] a_{f,k} a_{g,k'}) b_{f,k}(x) b_{g,k'}(x). \tag{39}$$

[3] In the main text, we show a o-dimensional version: $W_1 F(x) + W_2 G(x)$, where $W_1 \in \mathbb{R}^{m_1 \times o}$ and $W_2 \in \mathbb{R}^{m_2 \times o}$. The o-dimensional version can be seen as an easy extension of this single dimensional version as we can view $W_1 = [\mathbf{w}_{11}^T; \ldots; \mathbf{w}_{1o}^T]$ where $\mathbf{w}_{1i} \in \mathbb{R}^{m_1}$ and similarly for W_2.

By Definition A.5, we can assume

$$b_{f,k}(x)b_{g,k'}(x) = \sum_l c_{k,k',l} b_{k,k',l}(x), \quad b_{k,k',l}(x) \in S_1 \otimes S_2. \tag{40}$$

Putting it into Eq. (39), we have:

$$M(x) = \sum_{i,j \geq 1,k,k'}^{i,j \leq m_1,m_2} (W[i,j] a_{f,k} a_{g,k'}) \sum_l c_{k,k',l} b_{k,k',l}(x) \tag{41}$$

$$= \sum_{i,j \geq 1,k,k',l}^{i,j \leq m_1,m_2} (W[i,j] a_{f,k} a_{g,k'} c_{k,k',l}) b_{k,k',l}(x), \tag{42}$$

which is subband limited by $S_1 \otimes S_2$ since $b_{k,k',l} \in S_1 \otimes S_2$. □

With this theorem, we are able to leverage the PNF-controllable sets of subbands to compose PNFs with multiplication and additions in a band-limited ways. For example, if we want to construct a PNF that's band-limited by S, we can do it in the following steps:

1. Identify a PNF-controllable set of subbands that contains S, let that be $\mathcal{S}$ with the associated operation to be $\otimes$.
2. Factorize S into a series of subband by the associated operation : $S = S_1 \otimes S_2 \otimes \cdots \otimes S_n$.
3. For each S_i, we can creates a shallow PNF with $F_i = g^{(i)} \circ \gamma^{(i)}$, where $\gamma_j^{(i)} \in S_i$ for all j and $g^{(i)}(z) = Wz$ is a linear layer (without bias).
4. Composed these layers together using rule-2 of Theorem 11.

One can see that the abovementioned way to construct PNF has the ability to create exponential number of basis functions, all of which within S. Following is the intuitive reason why that's the case. Suppose each neuron F_i is a linear sum of d different basis functions (i.e. the set of basis chosen for F_i is different from those of F_j if $i \neq j$). Furthermore, suppose that Definition A.2 creates different sets of basis in the right-hand-side of the multiplication. then every time we apply rule-2 of Theorem 11 on d-dimensional inputs (i.e. assume that we have d-numer of W matrix in rule-2 to create an output of d-dimensional everytime), we creates d-number of different basis for every-single existing cases. As a result, if we apply rule-2 L-times, then we will have d^L number of different basis. At the same time, the number of parameters we used is d^2 for each F_i and d^3 for each multiplication. This means that the number of parameter is $O(Ld^3)$ while the number of basis functions we create in the final linear sum is $O(d^L)$. As a result, such construction can potentially lead to a compact, expressive, and subband-limited PNF.

Note that what's described above is merely an intuitive argument, since many of the conditions might be difficult to hold strictly (e.g. different basis are created for multiplication on the right-hand-sides of Definition A.2). Empirically, we found that while these conditions are only loosely held true, such construction is still capable of creating a large number of basis functions. This observation aligns with the analysis of MFN-like network can a large number of basis functions, presented in MFN [22] and BACON [36]. In the rest of the subsection, we will show some construction of PNF-controllable set of subbands with various basis functions.

A.4.1 Fourier Basis

In this section, we will consider Fourier basis of $\mathbb{R}^d \to \mathbb{R}$ parameterized by $\omega \in \mathbb{R}^d$: $\mathcal{B}_{Fourier} = \{\exp(i\omega^T x)\}_\omega$. A commonly used subband definition (as shown in Equation (1) of the main text) is following:

Definition A.6 (Fourier Subband)**.** [4] *The Fourier subband for Fourier basis of $\mathbb{R}^d \to \mathbb{R}$ functions can be defined with a lower band limit $\alpha \in \mathbb{R}^+$, an upper band limit $\beta \geq alpha$, an orientation $\mathbf{d} \in \mathbb{R}^d$ and an angelar width $\gamma \in \mathbb{R}^+$:*

$$R_F(\alpha, \beta, \mathbf{d}, \gamma, p) = \left\{ \boldsymbol{\omega} | \alpha \leq \|\boldsymbol{\omega}\|_p \leq \beta, \|\mathbf{d}\| = 1, \boldsymbol{\omega}^T \mathbf{d} \geq \cos(\gamma) \|\boldsymbol{\omega}\|_2 \right\}. \tag{43}$$

[4] There are two typos in Equation (1) of the main text. 1) the inequality sign is flipped; and 2) γ should be within the cos. Note that the last term $\boldsymbol{\omega}^T \mathbf{d} \geq \cos(\gamma) \|\boldsymbol{\omega}\|$ referrs to 2-norm, which we added explicitly here.

With these subbands, we can find PNF-controllable set of subbands in the following format:

Theorem 12 (Fourier PNF-Controllable Set of Subbands with L2 norm)**.** [5] *For $|\gamma| < \frac{\pi}{4}$, define $\mathcal{S}_{Fourier-L2}(\mathbf{d}, \gamma) = \{R_F(\alpha, \beta, \mathbf{d}, \gamma, 2) | \forall 0 \leq \alpha \leq \beta\}$. Each subband in $\mathcal{S}$ can be parameterized with tuple (α, β). $\mathcal{S}_{Fourier-L2}(\mathbf{d}, \gamma)$ is a PNF-Controllable Set of Subbands with the following definition of binary relation $\otimes_{FF}$:*

$$(\alpha_1, \beta_1) \otimes_{FF} (\alpha_2, \beta_2) = (\sqrt{\cos(2\gamma)}(\alpha_1 + \alpha_2), \beta_1 + \beta_2). \tag{44}$$

Proof. It's sufficient to show that $\omega_1 \in R(\alpha_1, \beta_1, \mathbf{d}, \gamma, 2)$, and $\omega_2 \in R(\alpha_2, \beta_2, \mathbf{d}, \gamma, 2)$, then $\omega_1 + \omega_2 \in R(\sqrt{\cos(2\gamma)}(\alpha_1 + \alpha_2), \beta_1 + \beta_2, \mathbf{d}, \gamma, 2)$. Let's start with the bound in the norm:

$$\|\omega_1 + \omega_2\| = \sqrt{\|\omega_1\| + \|\omega_2\| + 2\omega_1^T \omega_2} \tag{45}$$

$$= \sqrt{\|\omega_1\|^2 + \|\omega_2\|^2 + 2\|\omega_1\| \|\omega_2\| \cos(\theta)}, \tag{46}$$

where θ is the radius of the angle between ω_1 and ω_2. Since $\cos(\theta) \leq 1$, it's easy to show that:

$$\|\omega_1 + \omega_2\| = \sqrt{\|\omega_1\|^2 + \|\omega_2\|^2 + 2\|\omega_1\| \|\omega_2\| \cos(\theta)} \tag{47}$$

$$\leq \sqrt{\beta_1^2 + \beta_2^2 + 2\beta_1\beta_2} = \beta_1 + \beta_2. \tag{48}$$

Now we need to show that $\|\omega_1 + \omega_2\| \geq \alpha_1 + \alpha_2$. todo:By Eq. (43), we know that ω_1's angle with $\mathbf{d}$ is at most γ. Similarly, ω_2's angle with $\mathbf{d}$ is also at most γ. With that said, the largest angle between ω_1 and ω_2 should be less than 2γ. As a result, $\cos(\theta) \geq \cos(2\gamma)$. Note that $|\gamma| \leq \frac{\pi}{4}$, so $\cos(\theta) \geq \cos(2\gamma) \geq \cos(\frac{\pi}{2}) = 0$. With that, we can show:

$$\|\omega_1 + \omega_2\| = \sqrt{\|\omega_1\|^2 + \|\omega_2\|^2 + 2\|\omega_1\| \|\omega_2\| \cos(\theta)} \tag{49}$$

$$\geq \sqrt{\alpha_1^2 + \alpha_2^2 + 2\|\omega_1\| \|\omega_2\| \cos(\theta)} \tag{50}$$

$$\geq \sqrt{\alpha_1^2 + \alpha_2^2 + 2\alpha_1\alpha_2 \cos(2\gamma)} \quad \text{(Possible since } \cos(\theta) \geq \cos(2\gamma) \geq 0) \tag{51}$$

$$= \sqrt{\cos(2\gamma)(\alpha_1^2 + \alpha_2^2 + 2\alpha_1\alpha_2) + (1 - \cos(2\gamma))(\alpha_1^2 + \alpha_2^2)} \tag{52}$$

$$\geq \sqrt{\cos(2\gamma)(\alpha_1^2 + \alpha_2^2 + 2\alpha_1\alpha_2)} \quad \text{Since } \cos(2\gamma) \leq 1 \text{ and } \alpha_1, \alpha_2 \geq 0 \tag{53}$$

$$= \sqrt{\cos(2\gamma)}(\alpha_1 + \alpha_2). \tag{54}$$

Finally, we want to show that $(\omega_1 + \omega_2)^T \mathbf{d} \geq \cos(\gamma) \|\omega_1 + \omega\|$:

$$(\omega_1 + \omega_2)^T \mathbf{d} = \omega_1^T \mathbf{d} + \omega_2^T \mathbf{d} \tag{55}$$

$$\geq \cos(\gamma) \|\omega_1\| + \cos(\gamma) \|\omega_2\| \tag{56}$$

$$= \cos(\gamma)(\|\omega_1\| + \|\omega_2\|) \tag{57}$$

$$\geq \cos(\gamma) \|\omega_1 + \omega_2\| \quad \text{by Triangular inequiality of L2-norm.} \tag{58}$$

□

Intuitively, the smaller the angel γ is, the tighter we are able to guarantee the lower-bound compared to $\alpha_1 + \alpha_2$. With this said, if we want to build a subband with lower-bound l and angle γ, one way to achieve it is to set $\alpha_1 = 0$ and $\alpha_2 = \frac{l}{\sqrt{\cos(2\gamma)}}$. But we found empirically that the network perform better when the lower-band-limit is overlap with the upper-band-limit. We will discuss this more in Appendix B.

A.4.2 Fourier L1

[5] This is a paraphrase of Theorem 3 in the main-text. Theorem 3 in the main-text has several errors. First, the condition of γ is wrong. Second, it missing term $\cos(2\gamma)$ when Eq. (44).

As mentioned in Section 3.3.2 of the main paper, when working in image domain, a preferable way to define the PNF-controllable set of subbands is through L-∞ norm. This is because the L-∞ norm creates subband that can tile the corner of the frequency domain created by the image without going over the Nyquist rate. But to achieve it, we need to restrict each subband within a total-vertical or total-horizontal region. Intuitively, those regions contains vectors whose L-∞ norm is taking absolute value of the same dimension:

Figure 9: Visualization of the four L-∞ consistent regions in $\mathbb{R}^2$, one color for each.

Definition A.7 (L-∞ dimension consistent region)**.** *We call a region $R_\infty(n, s) \subset \mathbb{R}^d$ has consistent L-∞ dimension of n and sign $s \in \{+1, -1\}$ if:*

1. $\forall \mathbf{d} \in R, \|\mathbf{d}\|_\infty = |\mathbf{d}[n]|$, *where* $\mathbf{d}[n]$ *denotes the* n*th value of vector* $\mathbf{d}$*; and*
2. $\forall \mathbf{d}_1, \mathbf{d}_2 \in R$, $\text{sign}(\mathbf{d}_1[n]) = \text{sign}(\mathbf{d}_2[n]) = s$.

Please refer to Fig. 9 for the four L-∞ dimension consistent region in $\mathbb{R}^2$. In general, for $\mathbb{R}^d$, there are $2d$ number of such regions.

It's easy to see the following property of $R_\infty(n)$:

Lemma 13. *If* $\mathbf{d} \in R_\infty(n, s)$*, then* $a\mathbf{d} \in R_\infty(n, s)$ *for* $a \in \mathbb{R}$ *and* $a > 0$.

Proof.

$$\arg\max_i |(a\mathbf{d})[i]| \arg\max_i |a\mathbf{d}[i]| = \arg\max_i |a||\mathbf{d}[i]| = \arg\max_i |\mathbf{d}[i]| = n. \tag{59}$$

Since $a > 0$, $\text{sign}(a\mathbf{d}[n]) = \text{sign}(\mathbf{d}[n])$. □

With this, we are able to prove a similar version of Theorem 12 for L-∞ norm but restricted each set of subband to be within only a region with consistent L-∞ dimension:

Theorem 14 (Fourier PNF-Controllable Set of Subbands with L-∞ norm)**.** *Let $R_\infty(n, s)$ be a region of consistent L-∞ dimension of n defined for $\mathbb{R}^d$. Define the set of subbands as following:*

$$\mathcal{S}_{FF-L\infty}(\mathbf{d}, \gamma, n) = \{R_F(\alpha, \beta, \mathbf{d}, \gamma, \infty) | \forall 0 \leq \alpha \leq \beta, R_F(\alpha, \beta, \mathbf{d}, \gamma, \infty) \subset R_\infty(n, s)\}. \tag{60}$$

Each subband in $\mathcal{S}_{FF-L\infty}$ can be parameterized with tuple (α, β). $\mathcal{S}_{FF-L\infty}(\mathbf{d}, \gamma)$ is a PNF-Controllable Set of Subbands with the following definition of binary relation $\otimes_{FF}$:

$$(\alpha_1, \beta_1) \otimes_{FF\infty} (\alpha_2, \beta_2) = (\alpha_1 + \alpha_2, \beta_1 + \beta_2). \tag{61}$$

Proof. Similar to the proof in Theorem 12, we will show that if $\omega_1 \in R_F(\alpha_1, \beta_1, \mathbf{d}, \gamma, \infty) = R_1 \in \mathcal{S}_{FF-\infty}$ and $\omega_2 \in R_F(\alpha_2, \beta_2, \mathbf{d}, \gamma, \infty) = R_2 \in \mathcal{S}_{FF-\infty}$, then we will have $\omega_1 + \omega_2 \in R_F(\alpha_1 + \alpha_2, \beta_1 + \beta_2, \mathbf{d}, \gamma, \infty) \in \mathcal{S}_{FF-\infty}$.

First, we show that $\omega_1 + \omega_2$ is still within $R_\infty(n)$. As shown in the proof for Theorem 12, we have $(\omega_1 + \omega_2)^T \mathbf{d} \geq \cos(\gamma) \|\omega_1 + \omega_2\|$. This means that $\omega_1 + \omega_2 \in R_\infty(n)$ by Theorem 13.

Since R_1, R_2, and $\omega_1 + \omega_2$ are both in $\mathcal{S}_{FF-\infty}$, we will use this property to derive the upper bound:

$$\|\omega_1 + \omega_2\|_\infty = |(\omega_1 + \omega_2)[n]| \leq |\omega_1[n]| + |\omega_2[n]| = \|\omega_1\|_\infty + \|\omega_2\|_\infty = \beta_1 + \beta_2. \tag{62}$$

If $s > 0$, then we know $\omega_1[n], \omega_2[n] > 0$ and $(\omega_1 + \omega_2)[n] > 0$. As a result, $\|\omega_{1,2}\|_\infty = \omega_{1,2}[n] \geq \alpha_{1,2}$ respectively. This implies $\|\omega_1 + \omega_2\|_\infty = \omega_1[n] + \omega_2[n] \geq \alpha_1 + \alpha_2$.

Similar arguement can be applied when $s < 0$. □

Note that this theorem shows that the rectangular tiling mentioned in Equation (3) of the main text is actually operating under a PNF-controllable set of subbands. We also provide an illustration of these two subband binary functions in Fig. 10.

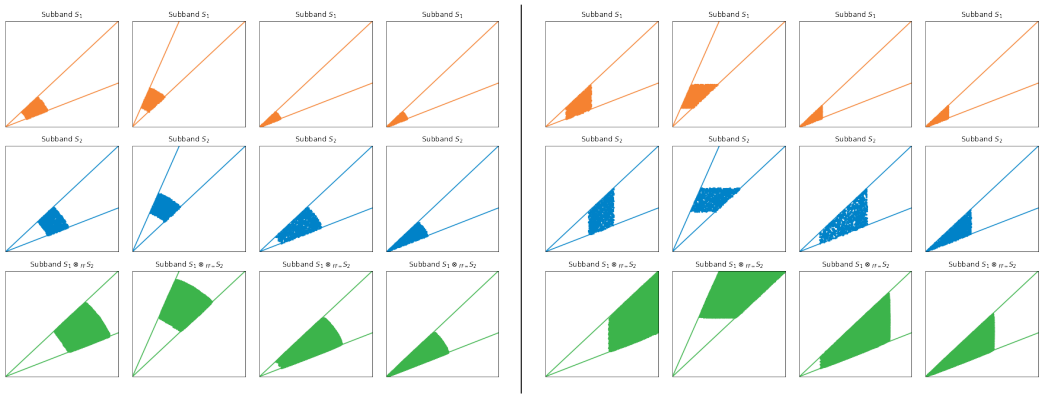

Figure 10: Illustration of how subbands within PNF-Controllable Sets of subbands transform under multiplication. We illustrate subbands defined in Eq. (43). So x-axis is $\omega[1]$ and y-axis is $\omega[2]$. The left four figures shows the operation under $\otimes_{FF}$ (Theorem 12) and the right four figures shows operations under $\otimes_{FF\infty}$ (Theorem 14). The top two rows are S_1 and S_2, and the last row is $S_1 \otimes S_2$ with $\otimes$ to be the corresponding binary functions defined in the PNF-controllable set of subbands.

A.4.3 RBF

Theorem 15 (RBF PNF-Controllable Set of Subbands)**.** *Assume the RBF-basis functions for $\mathbb{R}^d \to \mathbb{R}$ in the form $\mathcal{B}_{RBF} = \{\exp(-\frac{1}{2}\gamma \|x-\mu\|)\}$ parameterized by tuple (γ, μ), where $\gamma \in \mathbb{R}$ and $\mu \in \mathbb{R}^d$. Define subband in the following way:*

$$S_{RBF}(M, \gamma) = \{\exp(-\frac{1}{2}\gamma \|x-\mu\|) | \mu \in \mathrm{Cvx}(M)\}, \tag{63}$$

where $M \subset \mathbb{R}^d$ and $\mathrm{Cvx}(M)$ denotes the convex hull using all vectors of M. Then

$$\mathcal{S}_{RBF} = \{S_{RBF}(M, \gamma) | \forall M \subset \mathbb{R}^d, |M| < \infty, \gamma > 0\} \tag{64}$$

is a PNF-controllabl Set of Subbands with the following definition of binary relation $\otimes_{RBF}$:

$$(M_1, \gamma_1) \otimes_{RBF} (M_2, \gamma_2) = (M_1 \cup M_2, \gamma_1 + \gamma_2). \tag{65}$$

Proof. (Sketch) This is because by applying Eq. (13) , we have μ' is a weighted sum of μ_1 and μ_2 with weights normalized to 1. With that said, μ' is in the convex hull of where μ_1 and μ_2 is sampling from. The operation on γ is taken directly from Eq. (13). □

We can easily generalize Theorem 15 to subband definition where γ samples from an interval since it only makes sense when $\gamma > 0$. The abovementioned theory suggests that every-time we multiply two RBF basis PNF, we will increase the region of the convex hull of M and increase the γ (which is inverse to the scale of the RBF).

A.4.4 Gabor

We will show how to combine RBF and Fourier cases of the PNF-controllable set of subbands to create a PNF-controllable set of subband for Gabor basis.

Theorem 16 (Multiplication Rule of PNF-Controllable Set of Subbands)**.** *Let $\mathcal{B}_1$ and $\mathcal{B}_2$ be two basis for function $\mathbb{R}^d \to \mathbb{R}$ that satisfies Definition A.2. Let $\mathcal{S}_1, \otimes_1$ and $\mathcal{S}_2, \otimes_2$ be the PNF-controllable set of subbands for $\mathcal{B}_1$ and $\mathcal{B}_2$ respectively. Assume the subbands for $\mathcal{B}_1$ and $\mathcal{B}_2$ are parameterized by $\theta_1 \in \mathbb{R}^n$ and $\theta_2 \in \mathbb{R}^m$ correpsondingly. Define $\mathcal{B}_3 = \{b_1 b_2 | b_1 \in \mathcal{B}_1, b_2 \in \mathcal{B}_2\}$. Define subband as $S_3(\theta_1, \theta_2) = \{b_1 b_2 | b_1 \in S_1(\theta_1), b_2 \in S_2(\theta_2)\}$, where S_1 and S_2 are subbands for $\mathcal{B}_1$ and $\mathcal{B}_2$ correspondingly. Then following is a PNF-controllable set of subbands for basis $\mathcal{B}_3$:*

$$\mathcal{S}_3 = \{S_3(\theta_1, \theta_2) | S_1(\theta_1) \in \mathcal{S}_1, S_2(\theta_2) \in \mathcal{S}_2\}, \tag{66}$$

whose corresponding binary function is defined as :

$$(\theta_{1,a}, \theta_{2,a}) \otimes_3 (\theta_{1,b}, \theta_{2,b}) = (\theta_{1,a} \otimes_1 \theta_{1,b}, \theta_{2,a} \otimes_2 \theta_{2,b}). \tag{67}$$

Proof. (Sketch) It's easy to show that $\mathcal{B}_3$ satisfies Definition A.2 using Theorem 8. Then we can show that multiplication of two basis function in $\mathcal{B}_3$ moves S_3 according to $\otimes_3$ by leveraging the definition of $\mathcal{S}_1, \otimes_1$ and $\mathcal{S}_2, \otimes_2$. □

The PNF-controllable sets of subbands for Gabor basis can be obtained by applying Theorem 16 to combine a Fourier set of subbands (e.g. Theorem 12) and the RBF set of subbands (e.g. Theorem 15).

A.5 Different Instantiation of PNFs

In the previous sections, we've shown that 1) PNF allows different architectures, and 2) PNF allows different choices of basis, and 3) PNF can be designed to be subband-limited without losing its compositionality. In this section, we will show several instantiation of the PNFs. Specifically, for each design of the PNF, we will use the following steps:

1. Identify the subband of interests. These subbands should be able to cover all necessary basis functions needed to reconstruct the signal correctly.
2. Organize the subbands according to the PNF-Controllable sets of subbands.
3. For each PNF-Controllable set of subband, create a PNF whose outputs are subband-limited to the corresponding subband of interests in the set.
4. The final PNF is subs of all the previous PNF.

In this section we will use the abovementioned framework to show how PNF can be instantiated in a different forms, using different basis and network architectures. The detailed instantiation of Fourier PNF will be discussed in Appendix B.

Gabor PNF. Similar to MFN [22], we use the same network architecture as Fourier PNF, but changing the basis into a Gabor basis (as shown in Theorem 6). With the Gabor basis, the definition of the subbands requires to include a partition in the spatial domain as shown in Theorem 15. For simplicity, we can set $\gamma \in [0, \infty)$ (i.e not trying to control γ) and set M in the following way:

$$M = \left\{ s_i \mathbf{e}_i + s_j \mathbf{e}_j | i \neq j, s_{i,j} \in \{0.5, -0.5\}, \mathbf{e}_i[k] = \begin{cases} 1 & \text{if } k = i \\ 0 & \text{otherwise} \end{cases} \right\}. \tag{68}$$

Intuitively, $\mathbf{e}_i$ is an one-hot vector with the ith dimension to be 1. An example of M in 2D is $\{[0.5, 0.5], [0.5, -0.5], [-0.5, 0.5], [-0.5, -0.5]\}$, which includes all the points in the rectangle of $[-0.5, 0.5]^2$. Since all data are sampled withitn $[-0.5, 0.5]^d$, this means the only contorl we want to enforce on μ is that it should live within the boundary $[-0.5, 0.5]^d$. With this, we are able to create a subband-limited version of Gabor MFN, whose output is a linear sums of the Gabor basis.

RBF PNF. Here we will show an interesting way to design an RBF PNF which corresponds to subdividing a rectangular grid and interpolating it with a Gaussian. Here we will use the subband defined by Definition A.5. We are interesting in modeling the following sets of subbands, one for each level l: $S_{RBF}(l) = S_{RBF}(M_l, 2^n \gamma_0)$, where $M_l = \{\sum_{i=1}^{d} \frac{k_i}{2^l} \mathbf{e}_i | k_i \in [-2^{l-1}, 2^{l-1})\}$ is a set of grid with resolution 2^l. While M_l contains 2^{2l} number of basis, this can be created through function composition very compactly through PNF. First, define $f(x) \in \mathbb{R}^{d^2}$ that each $f(x)[i]$ is an RBF function which takes a corner in $[-0.5, 0.5]^d$ as μ and γ_0 as the scale. Then we define the network in the following recursive way:

$$F_0 = f(x), \quad F_{k+1} = (A_{k+1} F_k(x)) \odot (B_{k+1} F_k(x)), \tag{69}$$

where $\odot$ denotes Hardarmard product and A, B are real-value matrices. We can use Theorem 11 and Theorem 15 to show that F_k is subband limited by $S_{RBF}(k)$ and it creates all μ's that's defined by M_l. The output of F_k can be viewed as an Gaussian interpolated version of points M_l, where the value of each M_l is given by the network parameters A's and B's.

Pairwise Gabor PNF. For the previous definition of Gabor PNF, we are not attempting to control the spatial content of the signal (i.e. there is no ability to pull out a network output with specific γ and M region). Fortunately, the PNF allows us to design a netowrk architecture to enable such control for both the spatial RBF part and the Fourier part. The idea is to have a branch of the network to generate RBF PNF $H_{k,l}$ that's subband limited by different M_k at different levels of $2^l\gamma_0$. This is achievable by creating an essemble of network using equation Eq. (69). Similarly, Fourier PNF creates a series of PNFs $G_{n,m}$, each of which is subband limited in different scale (indexed by n) and different orientation (indexed by m). We want to design the Pairwise Gabor PNF to be the sum of pairwise product of $F_{k,l}$ and $G_{n,m}$:

$$F(x) = \sum_{k,l,n,m} a(k,l,n,m) H_{k,l}(x) G_{n,m}(x), \tag{70}$$

where $a_{k,l,n,m}$ are trainable parameters. It's easy to show that the output of such design is subband limited applying Theorem 16. And this network design allows us to remove either certain spatial area by setting $a_{k,l,\cdot,\cdot} = 0$, or to remove certain frequency content by setting $a_{\cdot,\cdot,n,m} = 0$.

We will show results of some of these designs in the image expressivity experiments (i.e. Fig. 12.

B Implementation Details

As mentioned in Sec. 3.3.3 of the main text, we leverage Theorem 2 to factorize F[6]:

$$F(\mathbf{x}) = \textstyle\sum_j F_j(\mathbf{x}),\ F_j(\mathbf{x}) = G_j(\mathbf{x}, b_j, b_j) W_{jn} Z_{j,n}(\mathbf{x}), \tag{71}$$

$$Z_{j,1}(\mathbf{x}) = G_j(\mathbf{x}, 0, \Delta_1),\ Z_{j,k}(\mathbf{x}) = G_j(\mathbf{x}, 0, \Delta_k) W_i Z_{j,k-1}(\mathbf{x}), \tag{72}$$

where $G_j(\mathbf{x}, a, b)$ is subband limited in $R_F(a, b, d(\theta_j), \delta, \infty)$ and $\Delta_k = b_k - b_{k-1}$. We instantiate this architecture by setting $G_j(\mathbf{x}, a, b)$ into a linear transform of basis sampled from the subband to be limited:

$$G_j(\mathbf{x}, a, b) = W_i \gamma_j(\mathbf{x}), \gamma_j \in R_F(a, b, d(\theta_j), \delta, \infty)^d, W_i \in \mathbb{R}^{h \times d}, \tag{73}$$

where h and d is the dimension for the output and the feature encoding. We realize the W_i as linear layers with no bias. $\gamma_j \in R_F(a, b, d(\theta_j), \delta, \infty)^d$ are initialized randomly, with θ and the radius chosen uniformly in $R_F(a, b, d(\theta_j), \delta, \infty)^d$.

For experiments involving 2D images (e.g., image fitting, texture manipulation), the output dimension h for $G_j(\mathbf{x}, b_j, b_j)$ is set to be 3 (RGB output values). The input dimension d for $Z_{j,1}(\mathbf{x}) = G_j(\mathbf{x}, 0, \Delta_1)$ is set to be 2 (x and y coordinates). Otherwise the hidden dimensions are chosen to be 128. For 3D SDF fitting, a hidden dimension of size 100 is chosen, and for NeRF, a hidden dimension of 86 is chosen. The input dimension for 3D SDF fitting and for NeRF is 3 (we follow BACON's setting of modeling irradiance fields). The output dimension is 1 for 3D SDF fitting and 4 for NeRF (RGB and Occupancy values).

Tiling For images, the region of interest chosen for tiling is $[-B, B]^2$ where B is band limit, set to be 64, following BACON [36]. Eq. 2 and Eq. 3 describe a potential tiling with no overlapping fans. In practice, for fitting tasks we found it beneficial to use overlapping fans, and so consider the following, modified tilings:

$$\begin{aligned}\mathcal{T}_{circ} = \{S_{ij} = R_F(b_i, b'_{i+1}, \mathbf{d}(\theta_j), \delta, 2) | b_1 \leq \cdots \leq b_{n-1}, b'_2 \leq \cdots \leq b'_n, \\ \theta_j = j\delta, \delta = \frac{\pi}{m}, 1 \leq j \leq 2m\},\end{aligned}$$

$$\begin{aligned}\mathcal{T}_{rect} = \{S_{ij} = R_F(b_i, b'_{i+1}, \mathbf{d}(\theta_j), \delta, \infty) | b_1 \leq \cdots \leq b_{n-1}, b'_2 \leq \cdots \leq b'_n, \\ \theta_j = j\delta, 1 \leq j \leq 2m, j \neq m\}.\end{aligned}$$

This allows us to cover the space of frequency basis more compactly and learn a more faitfull fitting of a given signal. In particular, we set $b_1 \leq \cdots \leq b_{n-1}$ to be $0, \frac{1}{16}, \frac{1}{8}, \frac{1}{4}$ and $b'_2 \leq \cdots \leq b'_n$ to be $0, \frac{1}{8}, \frac{1}{8}, \frac{1}{4}, \frac{1}{2}$. m is chosen to be $\frac{1}{8}$, covering the frequency band with 8 orientations. By default we use rectangular tiling for our experiments with PNF.

[6]Note the main text has a typo, using b_i instead of b_j.

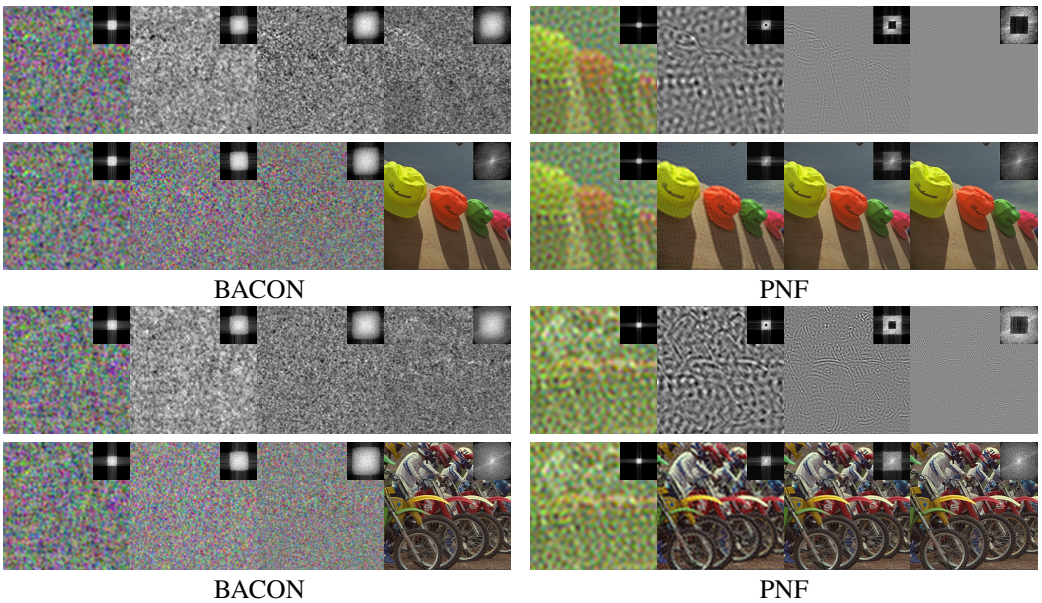

Figure 11: BACON: The bottom row shows the output of each layer which is upper band limited. The top row (columns 2-4) shows the difference between the output of a given layer and the one before it. PNF: The top row shows the output of each layer which is both upper and lower band limited. The bottom row (columns 2-4) shows the addition of the output at a given layer and the one before it.

For experiments that requires higher dimensional inputs (e.g. 3D), we mainly use a generalization of the rectangular tiling. We will first create a tile for each L-∞ dimensional consistent region. In the case of 3D, we will create one PNF-controllable set of subbands for each of the following: $R_\infty(1,1)$, $R_\infty(2,1)$, and $R_\infty(3,1)$. We use the same division of bandwidth within the construction of each of these PNF-controllable subband sets, but scaled it to correpsnding max bandwidth according to different applications. While this already covers all the basis function of interests, we found it improves the performance if we tile the frequency space in an overcomplete way. Specifically, for each pairs of the eight octants in $[-B, B]^3$ where B is the band-limit, we will create three non-overlapping PNF-controllable sets of subbands with the L-∞ norm. One interesting trick we leveraged is that the three non-overlapping PNF-controllabel sets of subband within the same octant can be implemented with one band-limited PNF. With this said, for 3D, we will need one band-limited PNF for each axis and one for each pairs of octants. This leads to an enssemble 7 band-limited PNFs.

C Experiment Details

C.1 Expressivity

Images For the image fitting task (Sec. 4.1, Tab. 1), we use the DIV2K [2] dataset and downsample images to 256^2 resolution. For evaluation, we sample the fields at 512 and compare with the original resolution images. We compare our method against state-of-the-art neural fields of BACON [36], Random Fourier Features [63] and SIREN [60]. Fig. 12 corresponds to Tab.1 of the main paper. We add here SD values corresponding to an averaged over all 25 images in DIV2K [2] dataset. In the paper we report values rectangular tiling. In Fig. 12 we also add the values corresponding to circular tiling. Additional results corresponding to Fig. 2 of the main paper are given in Fig. 11

Neural Radiance Field The method devised by NeRF [43] can be used for novel view synthesis. It operates on a dataset of images from different views with known camera parameters. NeRF queries a neural field M for an RGB and occupancy value, given a 3D point on a ray which passes through an image pixel that extends from a camera center. The RGB and occupancy values are then aggregated using standard volumetric rendering pipeline. After training, novel views are rendering by evaluating

Figure 12: Image Fitting on the DIV2K dataset. BAC stands for BACON. Rectangular stands for PNF using $\mathcal{T}_{rect}$ tiling, and Circular stands for PNF using $\mathcal{T}_{circ}$ tiling. Gabor is Gabor PNF and PWGabor is the Pairwise Gabor PNF, both are described in Appendix A.5

Method	PSNR	SSIM
RFF	28.72 ± 2.88	0.834 ± 0.053
SIREN	29.22 ± 3.08	0.866 ± 0.053
BAC	28.67 ± 2.83	0.838 ± 0.048
BAC-L	29.44 ± 3.03	0.871± 0.047
BAC-M	29.44 ± 3.03	0.871 ± 0.047
PNF (Rectangular)	**29.47 ± 3.08**	0.874 ± 0.047
PNF (Circular)	28.83 ± 3.04	0.856 ± 0.052
PNF (Gabor)	29.39 ± 2.97	**0.875 ± 0.047**
PNF (PWGabor)	29.22 ± 2.95	0.872 ± 0.047

the relevant rays. For test views, evaluation is performed by measuring the difference between generated views and ground truth views, using SSIM and PSNR measures.

For the choice of neural field M, we evaluate our method on the multiscale Blender dataset [4] with images at full (64×64), $1/2$, $1/4$, and $1/8$ resolution. We compare our method to BACON which is state-of-the-art on this task. We use the same training scheme as in BACON for this task. In Fig. 4, we also provide a visual comparison for the drums scene, trained at full (512×512), $1/2$, $1/4$, and $1/8$ resolution.

An Adam optimizer is used for training with $1e6$ training iterations. Learning rate is annealed logarithmically from $1e-3$ to $5e-6$. For BACON, 8 hidden layers are used with 256 hidden features. As mention in Appendix B, for our network, 4 layers are used with hidden dimension of 86. This results in the total memory which is slightly below that of BACON. Rays for the multiscale Blender dataset are in $[-4, 4]^3$. We follow follow BACON in setting the maximum bandwidth to be 64 cycles per unit interval and in evaluating without the viewing direction as input. We also adapt the hierarchical sampling of NeRF [43]. For a fair comparison to BACON we consider BACON's per-scale supervision using the loss of $\sum_{i,j,k} \|(\mathbf{I}_k(\mathbf{r}_i, \mathbf{t}_j) - \mathbf{I}_{\text{GT},k}(\mathbf{r}_i)\|_2^2$, for i, j, and k being index rays, ray positions, and dataset scales. Tab. 3 of the main text provides the result averaged over all scenes in the Blender dataset for $1x$ (64×64) resolution and for the average over $1x$, $1/2$, $1/4$, and $1/8$ resolution. Full results are provided in Tab. 5.

3D Signed Distance Field As mentioned in the main text, we evaluate the performance of our method against the Stanford 3D scanning repository[7]. In Tab. 2 of the main text we report the averaged for the scenes of *Armadillo*, *Dragon*, *Lucy*, and *Thai Statue*. Individual per-object scores are provided in Tab. 6 where we consider an additional evaluation metric of normal consistency. Normal consistency (NC), first computes the nearest points using Campfer Distance and then then computes whether the surface normal is within certain threshold. NC is the percentage that lands within the threshold. In Tab. 6, oracle corresponds to the upper bound for the performance, computed by sampling two sets of points form the ground truth and computing the evaluation metrics on them. We compare our method to SIREN [60] and BACON [36]. We train each network to fit a signed distance function (SDF). For BACON and SIREN, 8 hidden layers are used with 256 hidden features. The models are extracted at 512^3 resolution using marching cubes and evaluated using F-score and Chamfer distance.

We follow a similar training procedure to BACON [36]. Training data consists of sampled locations from the zero level set, where Laplacian noise is added for each point, as in [17]. As noted in BACON, the width of the Laplacian has a large performance impact, and so we use the same coarse and fine sampling procedure of BACON, where "fine" samples are produced using a small variance of $\sigma_{\text{L}}^2 =$ 2e-6 and "coarse" samples with $\sigma_{\text{L}}^2 =$ 2e-2. Samples are drawn in the domain $[-0.5, 0.5]^3$ and the following loss is used: $\lambda_{\text{SDF}} \|\mathbf{y}^c - \mathbf{y}^c_{\text{GT}}\|_2^2 + \|\mathbf{y}^f - \mathbf{y}^f_{\text{GT}}\|_2^2$, where $\mathbf{y}$ is the generated output,

[7] http://graphics.stanford.edu/data/3Dscanrep/

Table 5: NeRF Fitting for 64^2 resolution. Full results on 1x (64^2), 1/2, 1/4, 1/8 resolutions.

	BACON [36]				PNF (Ours)			
	300 epochs		500 epochs		300 epochs		500 epochs	
	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
LEGO 1x	29.62	0.969	30.07	0.973	30.40	0.967	31.10	0.965
LEGO 1/2x	29.90	0.955	30.65	0.957	30.41	0.962	31.17	0.951
LEGO 1/4x	29.91	0.939	30.46	0.939	31.08	0.947	31.75	0.944
LEGO 1/8x	28.87	0.913	28.85	0.917	30.39	0.921	30.86	0.934
Chair 1x	29.89	0.914	30.93	0.958	31.05	0.966	30.91	0.983
Chair 1/2x	30.97	0.898	34.93	0.968	35.00	0.949	34.88	0.975
Chair 1/4x	30.40	0.943	37.04	0.973	37.16	0.937	37.06	0.956
Chair 1/8x	29.65	0.947	35.92	0.964	36.29	0.929	36.55	0.956
Drums 1x	26.10	0.937	28.24	0.923	28.30	0.932	28.12	0.958
Drums 1/2x	28.38	0.942	30.20	0.956	30.05	0.939	30.16	0.960
Drums 1/4x	28.63	0.924	31.09	0.946	31.52	0.925	31.44	0.955
Drums 1/8x	32.79	0.950	32.25	0.940	32.64	0.954	32.96	0.946
Ficus 1x	25.61	0.925	28.18	0.953	29.31	0.965	30.34	0.976
Ficus 1/2x	31.49	0.955	33.12	0.975	30.37	0.957	30.45	0.962
Ficus 1/4x	35.48	0.973	35.55	0.979	30.96	0.945	30.33	0.943
Ficus 1/8x	37.98	0.971	38.00	0.977	30.35	0.922	28.09	0.912
Hotdog 1x	30.82	0.981	33.82	0.981	34.62	0.966	34.80	0.983
Hotdog 1/2x	29.39	0.973	34.37	0.974	34.15	0.949	34.35	0.975
Hotdog 1/4x	29.79	0.929	32.44	0.953	32.49	0.937	32.55	0.956
Hotdog 1/8x	28.72	0.939	32.28	0.940	31.87	0.929	32.15	0.956
Materials 1x	23.67	0.901	22.17	0.902	25.13	0.924	24.92	0.956
Materials 1/2x	25.70	0.946	24.17	0.915	28.23	0.935	27.85	0.972
Materials 1/4x	26.70	0.928	25.46	0.924	27.01	0.913	26.72	0.951
Materials 1/8x	26.01	0.806	24.47	0.842	27.68	0.860	26.62	0.863
Mic 1x	30.53	0.981	29.67	0.979	30.55	0.983	30.12	0.979
Mic 1/2x	34.02	0.972	31.72	0.973	33.98	0.975	33.52	0.976
Mic 1/4x	34.69	0.965	33.37	0.954	35.95	0.957	34.60	0.957
Mic 1/8x	35.54	0.958	36.00	0.954	35.35	0.953	35.52	0.952
Ship 1x	28.30	0.878	24.99	0.789	29.74	0.795	24.35	0.795
Ship 1/2x	29.74	0.901	27.02	0.812	27.11	0.899	27.17	0.829
Ship 1/4x	30.40	0.910	28.54	0.854	30.70	0.914	27.41	0.828
Ship 1/8x	31.42	0.931	29.00	0.892	31.40	0.891	28.17	0.888

$\mathbf{y}_{\text{GT}}$ is the ground truth value, the f and c indicate fine and coarse samples. As in BACON, λ_{SDF} is set to 0.01 for all experiments. SIREN and BACON are used as baselines, each trained for $200,000$ with a batch size of 5000 coarse and 5000 fine samples. The same optimization as for neural radiance field is used here.

C.2 Texture Transfer

For texture transfer, a random cat image is used (free-use licence) and texture image is used from [19]. As noted in the main text, we query the network on a 128^2 grid and apply either (a) Content and style loss objectives as given in [24], or (b) Text-based texture manipulation objectives as given by CLIPStyler [34]. For baseline comparison, the same objectives are applied directly on the cat image, and the cat image pixel values are directly optimized.

For (a), we use the same objectives (content and style loss), loss weights and optimizer as used in the public code of https://pytorch.org/tutorials/advanced/neural_style_

Table 6: 3D shape fitting. CD is Chamfer Distance ($\times 10^6$). FS stands for F-score and NC stands for normal consistency.

		Armadillo			Dragon		
	#Iters	CD	NC	FS	CD	NC	FS
SIREN		2.86	99.16	99.92%	11.9	97.03	99.43%
BACON	200k	2.86	99.12	99.91%	9.57	96.94	99.65%
BACON-last		2.88	99.05	99.91%	5.11	96.92	99.67%
BACON-mean		2.86	99.07	99.91%	9.80	96.83	99.57%
Ours	200k	2.87	99.14	99.91%	1.72	97.02	99.99%
	100k	2.86	99.14	99.92%	1.73	96.99	99.99%
	60k	2.87	99.13	99.91%	1.74	96.93	99.99%
	40k	2.88	99.11	99.90%	1.76	96.84	99.99%
	20k	2.91	99.05	99.91%	1.82	96.67	99.99%
Oracle		2.85	99.15	99.92%	1.67	97.84	100.00%

		Lucy			Thai		
	#Iters	CD	NC	FS	CD	NC	FS
SIREN		18.7	98.00	99.74%	2.61	94.71	99.96%
BACON	200k	18.1	97.79	99.82%	2.60	94.57	99.97%
BACON-last		4.95	97.03	99.73%	2.60	94.59	99.97%
BACON-mean		22.1	97.94	99.94%	2.61	94.42	99.97%
Ours	200k	1.75	97.89	100.00%	2.61	94.56	99.97%
	100k	1.75	97.87	100.00%	2.60	94.57	99.96%
	60k	1.76	97.81	100.00%	2.64	94.45	99.96%
	40k	1.77	97.78	100.00%	2.67	94.30	99.97%
	20k	1.81	97.66	100.00%	2.75	93.95	99.96%
Oracle		1.71	98.54	99.99%	2.54	95.55	99.97%

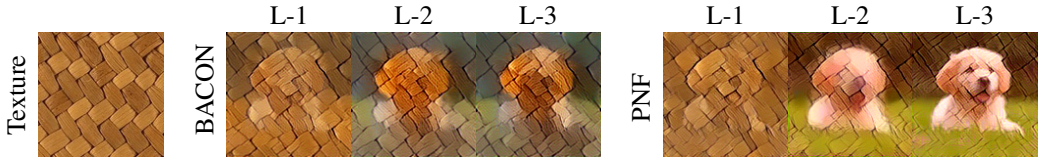

Figure 13: Additional texture transfer results. We optimize specific layers of the neural field. L-1 (layers 1-4), L-2 (layers 2-4), L-3 (layers 3-4).

tutorial.html and [24]. Similarly, for (b), we similarly use the same we objectives, loss weights and optimizer as used in the official code of CLIPStyler [34] in https://github.com/cyclomon/CLIPstyler. Additional texture transfer corresponding on the Fig. 5(a) is given in Fig. 13.

C.3 Scale-space Representation

In this section, we will first show the derivation of the PNF used for the scale-space representation. Then we will provide detailed description of the experiment and provide numerical results.

C.3.1 Derivation of Scale-space Fourier PNF

Suppose the signal of interest can be represented by Fouier bases as $g(\mathbf{x}) = \sum_n \alpha_n \exp(\omega_i^T \mathbf{x})$, then we know analytically the Gaussian convolved version should be $f(\mathbf{x}, \Sigma) = \sum_n \alpha_n \exp(\omega_i^T \Sigma \omega_i) \exp(i\omega_i^T \mathbf{x})$. If we assume that our Fourier PNF can learn the ground truth representation well, then one potential way to achieve this is setting $\gamma(\mathbf{x}, \Sigma)_n = \exp(-\frac{1}{2}\omega_n^T \Sigma \omega_n) \exp(-i\omega_n^T \mathbf{x})$. Doing this naively with Fourier PNF only yields a bad approximation, since each coefficient is off by an error term $E(n, I_n)$ from the ground truth. We will derive such error term in detail below.

For a particular F_k (Equation (4) in the main text), let $\tilde{F}_k(\mathbf{x}, \Sigma)$ be the output of naively replacing the Fourier basis encoding γ with the intergrated Fourier basis encoding: $\gamma(\mathbf{x}, \Sigma)$. Then we have:

$$\mathop{\mathbb{E}}_{\mathbf{x}\sim\mathcal{N}(\mu,\Sigma)}[F_k(\mathbf{x})] \tag{74}$$

$$= \sum_n \alpha_n \exp\left(-\frac{1}{2}\omega_n^T \Sigma \omega_n^T\right) \exp(i\omega_n^T x) \tag{75}$$

$$= \sum_n \exp\left(-\frac{1}{2}\omega_n^T \Sigma \omega_n^T\right) \sum_{I\in I_n} \alpha_{nI} \exp\left(i\left(\sum_{j\in I}\omega_j\right)^T \mathbf{x}\right) \tag{76}$$

$$= \sum_n \sum_{I\in I_n} \exp\left(-\frac{1}{2}\omega_n^T \Sigma \omega_n^T\right) \alpha_{nI} \exp\left(i\left(\sum_{j\in I}\omega_j\right)^T \mathbf{x}\right) \tag{77}$$

$$= \sum_{n, I\in I_n} \alpha_{nI} \exp\left(-\frac{1}{2}\sum_{j\in I}\omega_j^T \Sigma \omega_j^T\right) \exp\left(-\frac{1}{2}\sum_{j\neq l, j, l\in I}\omega_j^T \Sigma \omega_l^T\right) \exp\left(i\left(\sum_{j\in I}\omega_j\right)^T \mathbf{x}\right) \tag{78}$$

$$= \underbrace{\sum_{n, I\in I_n} \alpha_{nI}\left(\prod_{l\in I}\exp\left(-\frac{1}{2}\omega_l^T \Sigma \omega_l^T + i\omega_n^T \mathbf{x}\right)\right)}_{\text{PNF output: }\tilde{F}_k(\mathbf{x},\Sigma)} \underbrace{\exp\left(-\frac{1}{2}\sum_{l,j\in I, l\neq j}\omega_l^T \Sigma \omega_j^T\right)}_{\text{Error terms } E_k(n, I_n)}, \tag{79}$$

where I_n include all indexes to choose one basis per γ_j layer (as defined in Equation (6) of the main text) such that $\forall I \in I_n, \sum_{l\in I}\omega_l = \omega_n$. We use α_{nI} to denotes the coefficients gathered along the index of I. We can compute these coefficients analytically by applying similar analysis as MFN [22]. The formua above shows that $\tilde{F}_k(\mathbf{x}, \Sigma)$ is not a good approximation since each term $\alpha_{n,I}$ is off by different factor $E_k(n, I_n)$.

Fortunately, we know that all bases chosen from I_n must come from a specific set of that shares the same angle (i.e. the limiting-subbands all shared a direction vector $\mathbf{d}$ and an angular width γ by the definition of F_i), but with different range of for the norm $\|\omega\|_p$. This suggests that we can approximate the error terms in the following way:

$$E_i(\cdot, I) \approx \exp\left(-\frac{1}{2}\mathbf{d}_I^T \Sigma \mathbf{d}_I^T \sum_{k,l\in I, k\neq l} \bar{r}_k \bar{r}_l\right) = A_i(\Sigma), \tag{80}$$

Table 7: Numerical results for scale-space interpretation experiments. IPE: intergrated positional encoding [4]; RIPE: intergrated positional encoding with randomly fourier features; IPE-sup: supervised IPE with both 1x and 1/4x resolution. While PNF is only supervised with 1x, it's capable of inerpolating into smaller scale without breaking the image structure (as shown in good SSIM). This is more impressive as the basis functions for PNF is sampled from random directions (i.e. not necessarily aligned with the eigenvectors directions of the test-time Gaussian covariances). Doing the same thing to IPE results in worse performance as shown in RIPE.

	PSNR				SSIM			
Model	1x	1/2x	1/4x	1/8x	1x	1/2x	1/4x	1/8x
RIPE	39.41	6.35	6.54	6.63	96.95	20.43	22.73	23.8
IPE	37.58	24.62	21.94	14.68	94.67	73.88	58.51	37.95
BACON	40.63	8.88	7.19	7.2	97.39	43.11	28.16	29.66
PNF	36.45	**27.07**	29.74	**24.6**	95.98	**86.30**	83.33	**72.66**
RIPE-sup	**50.11**	11.91	26.64	10.69	**99.62**	15.51	73.24	11.27
IPE-sup	34.76	26.8	**30.00**	18.41	90.97	84.81	**89.65**	65.7

where $\bar{r}_k, \bar{r}_l$ are the medium radius of each subband within the subband series $\mathcal{S}_i$. Specifically, $\bar{r}_l = \frac{1}{2}(\alpha_l + \beta_l)$ if the lth subband used for F_k is $S_l \in \mathcal{S}_i$ lower- and upper- bounded by α_l and β_l. This way, the error estimation only depends on the PNF-controllable set of subbands used to developed the network F_k, but not the network parameter. This means we can modify the network architecture of each F_i in test-time to compute $A_k(\Sigma)$ to approximate error term $E_k(\cdot, I)$:

$$\mathop{\mathbb{E}}_{\mathbf{x}\sim\mathcal{N}(\mu,\Sigma)}[F_k(\mathbf{x})] \tag{81}$$

$$\approx \sum_{n,I\in I_n} \alpha_{nI} \left(\prod_{l\in I} \exp\left(-\frac{1}{2}\omega_l^T \Sigma \omega_l^T + i\omega_n^T \mathbf{x} \right) \right) A_k(\Sigma) \tag{82}$$

$$= A_i(\Sigma) \sum_{n,I\in I_n} \alpha_{nI} \sum_{n,I\in I_n} \left(\prod_{l\in I} \exp\left(-\frac{1}{2}\omega_l^T \Sigma \omega_l^T + i\omega_n^T \mathbf{x} \right) \right) \tag{83}$$

$$= A_i(\Sigma)\tilde{F}_k(\mathbf{x}, \Sigma) \tag{84}$$

With this, our scale-space interpolation Fourier PNF can be written as sums of outputs of all subband series $\mathcal{S}_k$:

$$\tilde{F}(\mathbf{x}, \Sigma) = \sum_k A_k(\Sigma)\tilde{F}_k(\mathbf{x}, \Sigma). \tag{85}$$

This approximation will be exact if $\gamma = 1$ and $\alpha = \beta$ for all subbands. But it also means we need infinite number of F_k to tile the space. The smaller γ is, the more error can this potentially occurs. Similarly, the larger $|\alpha - \beta|$ is, the more error this approximate can creates. This shows a trade-off between compute and the approximation error.

C.3.2 Details

The image used for Scale-space Representation is taken from the Set12 dataset [75] and downsampled to 128^2 resolution. We train all models for 1500 iterations with early stopping when the training PSNR reaches 40. All models use Adam optimizer with learning rate $1e-3$. The ground truth is generated from applying Gaussian filter of size 3×3 (for 1/2x), 5×5 (for 1/4x), and 65×65 (for 1/8x). We compute the ground truth using OpenCV [7]. For Σ, we use anisotropic covariance matrix of the form $\Sigma = \sigma^2 I$ with I being the 2×2 identity matrix. We set $\sigma^2 = 1$ for 1x, $\sigma = 2$ for 1/2x, $\sigma = 4$ for 1/4x, and $\sigma = 8$ for 1/8x. For PNF, we used the spherical tiling $\mathcal{T}_{rect}$ with 10 subbands with 2-times overcomplete. So each subband has angular width of $\frac{2}{12}$ as we need to restrict the subband into either total-vertical or total-horizontal area (i.e. $R_\infty(1, +1)$ or $R_\infty(0, +1)$ as definted in Definition A.7). The band limits for each intermediate residual output are set to be the following fraction of the Nyquist rates: $[0, \frac{1}{8}]$, $[\frac{1}{16}, \frac{1}{4}]$, $[\frac{1}{8}, \frac{1}{2}]$, and $[\frac{1}{4}, 1]$. In this experiment, we compare to the following baselines:

1. RIPE: we use intergrated random Fourier features in the following form: $\exp\left(-\frac{1}{2}\omega^T\Sigma\omega\right)\sin(\omega^T\mathbf{x})$. These intergrated random Fourier features will be followed by a ReLUMLP.
2. IPE: we use ReLUMLP on axis-aligned intergrated positional encodings as shown in [4].
3. BACON: replacing BACON's filter with the intergrated random Fourier Features.
4. RIPE-sup: same as RIPE, but supervised with both 1x and 1/4x.
5. IPE-sup: same as IPE, but supervised with both 1x and 1/4x.

All baselines have about 0.3M parameters (which is about the same number of floating points as the 128x128 image). All models are trained on 1x images except for IPE-sup, which is also supervised for 1/4x. The results are shown in Tab. 7. As we can see from the table, PNF is capable of representing 1/2x, 1/4x, and 1/8x resonably well while supervised only on 1x resolution. Comparing to IPE-sup, PNF is able to match the performance at 1/4, eventhough PNF is not supervised at 1/4x. Moreover, PNF used random Fourier features instead of the axis-align basis functions. As shown in the comparison between RIPEs and IPEs, axis-align positional encoding performs better in this setting, which we hypothesize the reason being that during the test-time the Σ's eigenvectors are also axis-align. PNF model choose direction randomly within the angle of the PNF-Controllable set of subbands, and it still achieves very strong performance.

C.4 Licenses

DIV2K [2] dataset is available for academic use only. The multiscale Blender dataset [4] is provided under the creative commons public license. The Stanford 3D scanning repository[8] dataset is available to use for research purposes. Data used for texture transfer is available for academic use or is free to use. Set12 dataset [75] is available under under the creative commons public license. All data does not contain personally identifiable information or offensive content.

C.5 GPU Resources

Image based experiments require a single GPU such as an NVIDIA Titanx/Titanrtx GPU. For Neural fields and 3D Signed Distance Fields, a higher GPU number is required per experiment, and up to 4 NVIDIA Titanrtx GPUs. An internal GPU cluster was used for experiments.

C.6 Negative Societal Impact

On some experimental settings, our framework requires a large GPU and memory requirement, which may have an adverse environmental impact. Neural fields have been used in a verity of generative settings, our framework may be used in potential negative use of such generative models. For instance, for neural radiance fields, creating realistic novel views of fake scenes, could be used maliciously. Such cases could be better handled by developing tools to detect fake scenes, or any other content generated using neural fields such as our PNF.

[8] http://graphics.stanford.edu/data/3Dscanrep/