Neural Fields as Distributions: Signal Processing Beyond Euclidean Space

Daniel Rebain[1], Soroosh Yazdani[2], Kwang Moo Yi[1], Andrea Tagliasacchi[3, 4, 5]

[1] University of British Columbia, [2] Google Research, [3] Google DeepMind,

[4] Simon Fraser University, [5] University of Toronto

Abstract

Neural fields have emerged as a powerful and broadly applicable method for representing signals. However, in contrast to classical discrete digital signal processing, the portfolio of tools to process such representations is still severely limited and restricted to Euclidean domains. In this paper, we address this problem by showing how a probabilistic re-interpretation of neural fields can enable their training and inference processes to become "filter-aware". The formulation we propose not only merges training and filtering in an efficient way, but also generalizes beyond the familiar Euclidean coordinate spaces to the more general set of smooth manifolds and convolutions induced by the actions of Lie groups. We demonstrate how this framework can enable novel integrations of signal processing techniques for neural field applications on both Euclidean domains, such as images and audio, as well as non-Euclidean domains, such as rotations and rays. A noteworthy benefit of our method is its applicability. Our method can be summarized as primarily a modification of the loss function, and in most cases does not require changes to the network architecture or the inference process.

1. Introduction

In the field of signal processing, there are few operations more fundamental than applying a filter to a signal, and the specific way we represent signals has a significant impact on what filtering methods are available. For example, neural fields are growing in popularity as a way of representing complex signals [17, 27, 29], but few efficient and general-purpose methods exist for applying filters to them [14, 20, 35]. Our goal in this work is to address this shortcoming, and provide a more general mathematical treatment of neural fields which will unlock better filtering methods for them.

The simplest approach to incorporating signal processing operations in neural field pipelines is to apply classical discrete operations to the data at a stage when it is represented discretely. For example, some filtering operations may be applicable to discrete training data like images or audio before the neural field is constructed. In other cases, it may be possible to apply the desired filter to a discrete *sampling* of the neural field's outputs, *e.g.*, rendered images. While an effective strategy may exist in the cases where it is applicable, there are many possible applications for which this is not a good option due to the dimensionality and/or

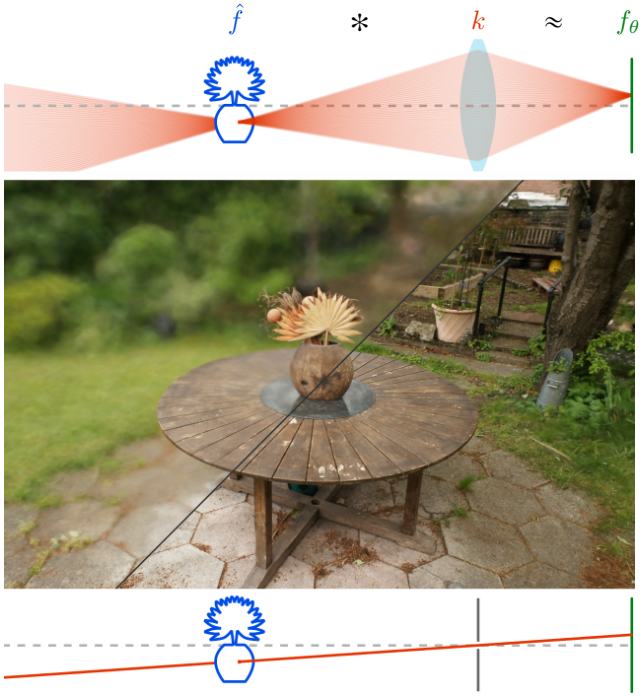

Figure 1. Our method enables a variety of neural field-based methods to be reinterpreted within a common framework of non-Euclidean signal processing. For example, we show how the simulation of lens effects, such as depth of field *(top)*, can be implemented *without extra inference cost* as a linear filter applied at training time. This filtering operation is realized as a group convolution via the action of $SE(3)$ between the light field of the scene $\hat{f}$ and distribution of rays k. Our training process results in the *observed* light field being *directly* approximated by the learned network $f_\theta \approx \hat{f} * k$. In the case where a typical pinhole camera model is desired, our method reduces to convolving with a Dirac delta distribution over the ray manifold *(bottom)*. See our project page at https://ubc-vision.github.io/nfd for more qualitative results.

topology of the signal's domain. An example of such an application, which we explore in this paper, is the task of modelling lens effects in novel view synthesis. Typically, a neural field model used for such applications operates on a four or five-dimensional non-Euclidean manifold of rays, and is supervised by sparse training data. As such, filtering before training is *not feasible* for most filter types, and filtering at inference increases cost significantly due to the extra network evaluations required. We aim to demonstrate that there are better options for this kind of filtering task that require neither.

Another approach would be to *modify* the field representation to allow some kinds of filters to be applied. This is a powerful approach, and has seen substantial success in some areas [1, 3, 14], but is limited in the kinds of filters it can support. Specifically, these methods typically focus on frequency decomposition, *i.e.*, band-limiting filters, and often require the filtered signal as supervision rather than constructing it from the input signal and filter kernel. These methods are useful for separating low and high-frequency components and tasks such as anti-aliasing, but are not suitable for more general tasks.

The approach we champion is to extend the *training* process for neural fields, in a way that is agnostic to both the neural architecture and the filter itself, to incorporate the computation of the filtered signal. This is ideal in many cases as it does not require any "extra" steps in the typical case of pipelines which *already* include neural field training as a required step.

To achieve this, we propose a novel formulation that *combines* training and filtering as a maximum likelihood estimation problem, where the signal itself is viewed as a probability distribution. We allow this by leveraging the theoretical link between the convolution of probability distributions and the transformation of random variables by group actions, which enables an efficient combined loss function.

To summarize, we make the following contributions:

- we derive a novel probability-based training objective for neural fields which produces filtered versions of the training signal *without* requiring filtered ground truth,
- we show how this formulation generalizes beyond fields on Euclidean spaces to convolutions of functions on the homogeneous spaces of arbitrary Lie groups, thereby expanding the set of available filtering operations,
- we demonstrate several potential applications for this method, including image filtering, environment lighting, and simulation of lens effects for novel view synthesis.

2. Related Works

Since their introduction [7, 16, 22], and mass popularization in the wake of the successes of Neural Radiance Fields [17], neural fields have been applied to many different problems across a number of fields [34]. Of particular interest to us are applications which are traditionally associated with, or adjacent to, signal processing. Notable examples are computer vision and graphics, where neural fields were first deployed, and have been used to model images [19], objects [11, 16, 22], environment maps [4, 39], light fields [25, 27, 28], and more. Similar applications have also appeared in adjacent fields like medical imaging [26, 37] and acoustics [10]. The intersections of our contributions with the entire literature on neural fields are far too numerous to list exhaustively, but broadly speaking, any applications which incorporate signal processing operations could have the potential to be made simpler, more efficient, or more general, through integration with our method.

Neural networks and group convolution. A central component of our contribution is the extension of neural field convolution to signals on non-Euclidan domains using *group convolution*. While our application of group theory to this particular problem is novel, we are far from the first to investigate the applications of group convolution in the context of neural networks. In particular, there is a line of research focused on extending the well-known *translation-equivariance* of Convolutional Neural Networks (CNNs) to other forms of equivariance [6, 9, 13]. A concrete example would be a CNN which learns to perform a rotation-equivariant transformation of a signal on the surface of a sphere. Similarly, our method could be used to train a neural field to represent the result of applying a rotationally-invariant filter kernel (or family thereof), to a signal defined on the surface of a sphere. The most important difference is that because we construct our method in terms of neural fields, we need not worry about the specifics of discretization or signal representation on such challenging domains.

Band-limited neural fields. While signal processing methods for neural fields remain under-explored, there has been work towards addressing this. Perhaps the most successful has been Mip-NeRF [1] and its many descendants and alternatives [2, 3, 14, 36]. These works modify the field network architecture, or the methods used to sample from it, with the goal of providing some control over the frequency content of the represented signal. The archetypal use case for this is anti-aliasing, which requires the application of a low-pass filter. Usually, this is achieved by decomposing or biasing the network to output different frequency components of the represented signal depending on the input provided.

Differential operators for filtering neural fields. While frequency-decomposition methods are useful for addressing aliasing, they do not support more general filters, and in some cases only provide rough approximations of the filtered signal. A few works have gone beyond this paradigm and attempted to support performing arbitrary convolutions

on neural fields. INSP [35] is a method which takes already-trained neural fields as inputs, and composes them with a function that applies some filter to their output. This is achieved by leveraging the differentiability of neural networks to predict not only the output of the input field at a point, but also the gradient and higher-order derivatives of the field. These can be mapped to the filtered output by an auxillary network which takes advantage of the information that spatial derivatives encode about the local neighbourhood. This approach has the distinct advantage of not depending on the original neural field architecture and the ability to operate on an already trained network. However, it is constrained in practice by the fidelity of the approximation provided by the finite number of derivatives used, which limits it to the use of simple kernels. By contrast, our method places no restrictions on the complexity or size of the filter kernel.

Neural field filtering by repeated differentiation. The most relevant to us is the recent work of Nsampi et al. [20], that propose a combined method of training and filtering a neural field representation. Differently from our method, they leverage the property that convolving a signal with a filter is equivalent to convolving the signal's anti-derivative and the filter's derivative. If the filter is approximated as a piecewise polynomial which will reduce to a sum of Dirac deltas under repeated differentiation, then the convolved signal may be computed in terms of repeated integrations of the signal. This is effective for filters which can be well approximated as low-order piecewise polynomials, but adds extra complexity by requiring construction of both the repeated integral field and approximated filter. We avoid this complexity, as our method requires only a way to draw samples from the desired filter kernel.

3. Neural fields as distributions

The literature on neural fields has largely converged to nomenclature in which "field" refers to some function over a coordinate space, similar to the usage of the term in physics, and distinct from the notion of a field in mathematics [34]. While some works have restricted their consideration to spatio-temporal domains [34, Def. 1], we will adopt a broader definition that considers functions defined over *smooth manifolds*. In particular, let $\mathbf{M}$ be an infinitely differentiable manifold, and $\hat{f}(\mathbf{x}) : \mathbf{M} \to \mathbb{R}^d$ be a continuous mapping. We will define a neural field as an approximation $\hat{f}(\mathbf{x}) \approx f_\theta(\mathbf{x})$ parameterized as a neural network with weights θ.

Coordinate encoder. It is important to note at this point that typical neural networks, as a consequence of their structure, require their inputs to be elements of $\mathbb{R}^m$. Rather than bake this restriction into our definition, we will assume the existence of a *coordinate encoder*, which continuously maps elements of $\mathbf{M}$ into $\mathbb{R}^m$. This generalization simplifies the mathematical treatment of neural field methods that operate on domains other than $\mathbb{R}^m$, such as rotations or rays. From now on, we will assume the coordinate encoder to exist as an internal module of f_θ.

Euclidean and non-Euclidean spaces. For the sake of accessibility, we will present a derivation of our training formulation in this section that considers only Euclidean spaces, *i.e.*, $\mathbf{M}{=}\mathbb{R}^m$. We will detail how our method extends to other smooth manifolds using a group-theoretic definition of convolution in Sec. 5.

3.1. Regression and maximum likelihood

Broadly speaking, training a neural field can be defined as the problem of finding network weights θ which achieve the approximation $\hat{f}(\mathbf{x}) \approx f_\theta(\mathbf{x})$ over some distribution of coordinate values $\mathbf{x} \sim \mathcal{Q}$ that represents the subdomain where ground truth values are known, and that may be continuous or discrete. This is frequently treated as a regression problem, where the error between ground truth observations and the predictions of the model are directly minimized. A very common implementation of this is least-squares regression, which yields an optimization objective of the form:

$$\mathcal{L} = \mathbb{E}_{\mathbf{x}\sim\mathcal{Q}}\ [||\hat{f}(\mathbf{x}) - f_\theta(\mathbf{x})||_2^2]. \tag{1}$$

Equation (1) can equivalently be interpreted as a maximum likelihood estimation problem where we assume that errors in predicted field values are normally distributed [12, Sec. 9.6]. Such an approach is ideal when the goal is to fit the exact signal represented by the ground-truth data. For our case, however, fitting a *filtered* signal $\hat{f} * k \approx f_\theta$ with access only to the unfiltered signal $\hat{f}$ would require expanding the convolution inside the loss:

$$\mathcal{L} = \mathbb{E}_{\mathbf{x}\sim\mathcal{Q}}\left[\left\|\left(\int_{\mathbb{R}^m} k(\mathbf{v})\hat{f}(\mathbf{x}-\mathbf{v})d\mathbf{v}\right) - f_\theta(\mathbf{x})\right\|_2^2\right]. \tag{2}$$

For filters which can be interpreted as probability distributions, such as Gaussians, the integral can be expressed as an expectation over samples drawn from k:

$$\mathcal{L} = \mathbb{E}_{\mathbf{x}\sim\mathcal{Q}}[||\mathbb{E}_{\mathbf{v}\sim k}[\hat{f}(\mathbf{x}-\mathbf{v})] - f_\theta(\mathbf{x})||_2^2]. \tag{3}$$

However, this approach requires that the ground truth signal value be known for *any* possible value of $\mathbf{x}-\mathbf{v}$ within the support of $\mathcal{Q} * k$. For applications which are trained from real-world measurements, the data is typically sparse and/or incomplete, and thus incompatible with this requirement. The other major drawback is the $O(n)$ complexity of the loss for each network evaluation, which requires n additional samples to approximate the application of the filter [8]. This *Monte Carlo* approach will be effective in any case where the sample mean of the expectation converges

quickly enough that only a small number of samples is required to achieve the desired accuracy. In practice, this limits consideration to kernels which are small relative to the Nyquist rate of the signal [20].

Conversely, our goal in this work is to achieve $O(1)$ complexity, where each evaluation of the network is supervised with *one* sample from potentially discrete ground truth data, which may not be amenable to continuous sampling. Achieving this goal will unlock signal processing applications in more challenging domains where the model is trained without direct access to the underlying signal, such as the construction of light and radiance fields from images (see Sec. 5.2).

To this end, we re-formulate our problem differently from standard regression. Rather than treating the field values as random variables and estimating their distribution, we instead model the *sample coordinates* as random variables, with field values interpreted as densities of the sample distribution. This transformation enables us to take advantage of the correspondence between convolution and random variable addition, and avoid the limitations of directly regressing the filtered field values by Monte Carlo estimate. We will show in the remainder of this section how this results in an efficient loss function for learning filtered signals.

3.2. Signals as sample distributions

To begin, we will make simplifying assumptions that we will later partially relax. First, we consider only *distribution-like* signals that map to a single *non-negative* real number and *integrate to one* over the coordinate manifold. This enables re-writing integrals as expectations over this distribution. We can construct a distribution-like field $\hat{p}$ from a non-negative ground truth field $\hat{f}$:

$$\hat{p}(\mathbf{x}) = \frac{\hat{f}(\mathbf{x})\mathcal{Q}(\mathbf{x})}{\int_{\mathbb{R}^m} \hat{f}(\mathbf{x}')\mathcal{Q}(\mathbf{x}')\, d\mathbf{x}'}. \tag{4}$$

We will also consider filters k that are distribution-like:

$$k(\mathbf{x}) \geqslant 0 \text{ for all } \mathbf{x} \in \mathbb{R}^m, \quad \int_{\mathbb{R}^m} k(\mathbf{x}) d\mathbf{x} = 1. \tag{5}$$

With these properties in place, we can construct our maximum likelihood estimation problem. Specifically, we wish to find parameters θ for our probability density function $p_\theta(\mathbf{x})$, such that the likelihood given samples from the target distribution is maximized. We also wish to incorporate filtering, so our target distribution will be the convolution of the signal and the filter $\hat{p} \ast k$. The resulting optimization is the maximization of the expected log-likelihood:

$$\ell(\theta | \hat{p} \ast k) = \mathbb{E}_{\mathbf{x} \sim \hat{p} \ast k} \ [\log(p_\theta(\mathbf{x}))]. \tag{6}$$

We can presumably neither draw samples from $\hat{p} \ast k$, nor directly evaluate its PDF, so we will start by rewriting the expectation in a more friendly form. First, we will expand the expectation and convolution as integrals:

$$\begin{aligned} \ell(\theta | \hat{p} \ast k) &= \int_{\mathbb{R}^m} (\hat{p} \ast k)(\mathbf{x}) \log(p_\theta(\mathbf{x})) d\mathbf{x}, \\ &= \int_{\mathbb{R}^m} \int_{\mathbb{R}^m} \hat{p}(\mathbf{x} - \mathbf{v}) k(\mathbf{v}) \log(p_\theta(\mathbf{x})) d\mathbf{v} d\mathbf{x}. \end{aligned} \tag{7}$$

Then we execute a change of variables $\mathbf{x}' = \mathbf{x} - \mathbf{v}$:

$$\ell(\theta | \hat{p} \ast k) = \int_{\mathbb{R}^m} \int_{\mathbb{R}^m} \hat{p}(\mathbf{x}') k(\mathbf{v}) \log(p_\theta(\mathbf{x}' + \mathbf{v})) d\mathbf{v} d\mathbf{x}',$$

so to re-write the double integral as nested expectations:

$$\ell(\theta | \hat{p} \ast k) = \mathbb{E}_{\mathbf{v} \sim k, \mathbf{x} \sim \hat{p}} \ [\log(p_\theta(\mathbf{x} + \mathbf{v})]. \tag{8}$$

This result is a direct consequence of the well-known equivalence between sums of random variables and convolution of distributions [12, Eq. 2.2.1].

At this point, we have reached a form for the optimization objective that can be evaluated. It is reasonable to assume that for both the kernel and the signal we can either draw samples or directly evaluate the PDF, either of which yield straightforward strategies for computing the expected value of the log-likelihood.

Our only remaining problem is that we still assume the learned $p_\theta(\mathbf{x})$ to behave like a PDF, *i.e.*, to be normalized. Because most neural field architectures do not satisfy this property by construction, we will further rewrite this in terms of an *un-normalized* field $f_\theta(\mathbf{x})$. In the case where the distribution $\mathcal{Q}$ of known ground truth values is continuous, this can be done straightforwardly by integrating over the support of $\mathcal{Q}$. Unfortunately, this distribution is often not continuous, as we frequently wish to supervise with discretized representations such as 2D pixel grids. While there may be interpolation strategies to interpret such data as continuous functions, we will keep our derivation general and consider the case where $\mathcal{Q}$ is non-zero only at discrete points, *i.e.*, a mixture of Dirac delta distributions.

As illustrated in Fig. 2, we apply a normalization strategy that handles both discrete and continuous supervision distributions. This is achieved by choosing to define $p_\theta(\mathbf{x})$ as proportional to the un-normalized field $f_\theta(\mathbf{x})$, weighted by the (potentially discrete) PDF $(\mathcal{Q} \ast k)(\mathbf{x})$:

$$p_\theta(\mathbf{x}) = \frac{(\mathcal{Q} \ast k)(\mathbf{x}) f_\theta(\mathbf{x})}{\int_{\mathbb{R}^m} (\mathcal{Q} \ast k)(\mathbf{x}') f_\theta(\mathbf{x}') d\mathbf{x}'}. \tag{9}$$

We use the convolution of $\mathcal{Q}$ and k here, as this will weight the normalization proportionally to the distribution of the random variable $\mathbf{x} + \mathbf{v}$, thereby maintaining balance between the probability and normalization terms. Using (9) in the MLE objective will ensure that the learned field is only constrained in areas where the supervision signal exists, thereby allowing the network outputs to vary freely

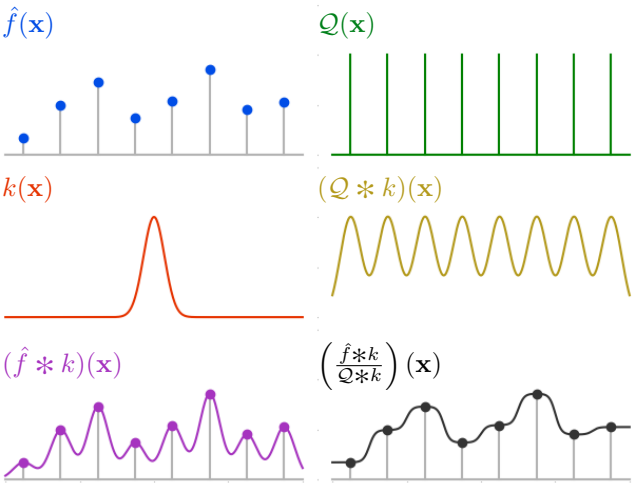

Figure 2. Traditional neural field training objectives which are supervised by a discretely sampled signal $\hat{f}(\mathbf{x})$ can be written as an expectation over the support distribution $\mathcal{Q}(\mathbf{x})$ of the data, thus allowing the natural interpolating property of the network to determine the value in unsupervised areas. Our method, however, must handle the case where a continuous kernel $k(\mathbf{x})$, of any shape or size, is applied to discrete data. If we normalize the predicted field uniformly over the coordinate space, the unsupervised areas are effectively interpreted as having ground truth value zero, resulting in the network learning the function $(\hat{f} * k)(\mathbf{x})$. Conversely, if the normalization constant is weighted by $(\mathcal{Q} * k)(\mathbf{x})$, this issue is avoided, and a reasonble interpolation can be achieved.

elsewhere, as is typical in neural field training. The resulting expression for the expected log-likelihood is:

$$\ell(\theta|\hat{p} * k) = \mathbb{E}_{\mathbf{v}\sim k,\mathbf{x}\sim\hat{p}}\left[\log\left(\frac{(\mathcal{Q} * k)(\mathbf{x}+\mathbf{v}) f_\theta(\mathbf{x}+\mathbf{v})}{\int_{\mathbb{R}^m}(\mathcal{Q} * k)(\mathbf{x}') f_\theta(\mathbf{x}') d\mathbf{x}'}\right)\right]. \quad (10)$$

Again, we need to eliminate the convolution operations which we cannot directly evaluate. First, we split the factors inside the log to separate terms:

$$\begin{aligned}\ell(\theta|\hat{p} * k) = \mathbb{E}_{\mathbf{v}\sim k,\mathbf{x}\sim\hat{p}}[\log((\mathcal{Q} * k)(\mathbf{x}+\mathbf{v}))] \\ + \mathbb{E}_{\mathbf{v}\sim k,\mathbf{x}\sim\hat{p}}[\log(f_\theta(\mathbf{x}+\mathbf{v}))] \\ - \log\left(\int_{\mathbb{R}^m}(\mathcal{Q} * k)(\mathbf{x}) f_\theta(\mathbf{x}) d\mathbf{x}\right). \quad (11)\end{aligned}$$

The first term does not depend on θ, and so will be irrelevant to the training optimization. The integral in the third term is of the same form as the one in (7), and can be transformed to an expectation in the same way:

$$\begin{aligned}\ell(\theta|\hat{p} * k) = \mathbb{E}_{\mathbf{v}\sim k,\mathbf{x}\sim\hat{p}}[\log((\mathcal{Q} * k)(\mathbf{x}+\mathbf{v}))] \\ + \mathbb{E}_{\mathbf{v}\sim k,\mathbf{x}\sim\hat{p}}[\log(f_\theta(\mathbf{x}+\mathbf{v}))] \\ - \log(\mathbb{E}_{\mathbf{v}\sim k,\mathbf{x}\sim\mathcal{Q}}[f_\theta(\mathbf{x}+\mathbf{v})]). \quad (12)\end{aligned}$$

Finally, we can write the maximum-likelihood estimation objective as a loss function which can be minimized by typical gradient-based optimizers:

$$\begin{aligned}\mathcal{L}_{\text{MLE}} = \mathbb{E}_{\mathbf{v}\sim k,\mathbf{x}\sim\hat{p}}[-\log(f_\theta(\mathbf{x}+\mathbf{v}))] \\ + \log(\mathbb{E}_{\mathbf{v}\sim k,\mathbf{x}\sim\mathcal{Q}}[f_\theta(\mathbf{x}+\mathbf{v})]). \quad (13)\end{aligned}$$

This objective is similar to that used in training Energy-Based Models [30], with the difference that the normalization term is evaluated over the data support $\mathcal{Q}$, rather than the modeled probability.

3.3. Practical considerations

Scale ambiguity of the solution. The loss function in (13) is sufficient only to constrain the value of $f_\theta(\mathbf{x})$ *up to a global scale*. Because the derivation incorporates the normalization constant term explicitly, the resulting objective has a null space corresponding to scalar multiplication of the network output. Multiple options exist for addressing this. If reconstructing the signal up to an arbitrary scale is sufficient, then only a simple regularizer of the mean output is needed. Alternatively, if the goal is to exactly match the scale of the input signal, including the gain of the filter k if it is not one, an additional loss term of the form $||\mathbb{E}_{\mathbf{x}\sim\mathcal{Q}}[\hat{f}(\mathbf{x})] - \mathbb{E}_{\mathbf{x}\sim\mathcal{Q}}[f_\theta(\mathbf{x})]||_2^2$ can be used to match the expected value of the predicted and ground truth signals.

Choice of sampling distributions. While directly approximating the expectations in (13) using the sample mean is a viable option, we found empirically that this is not the optimal strategy. In particular, this direct approach has two shortcomings: first, it requires two sets of samples at which the network must be evaluated, doubling the computational cost, and second, the disjoint locations of positive and negative supervision results in sub-optimal convergence for a given batch size. To address this, we simply rewrite the first term as an expectation over $\mathcal{Q}$:

$$\begin{aligned}\mathcal{L}_{\text{Balanced}} &= \mathbb{E}_{\mathbf{v}\sim k,\mathbf{x}\sim\mathcal{Q}}\left[-\log(f_\theta(\mathbf{x}+\mathbf{v}))\frac{\hat{p}(\mathbf{x})}{\mathcal{Q}(\mathbf{x})}\right] \\ &\quad + \log(\mathbb{E}_{\mathbf{v}\sim k,\mathbf{x}\sim\mathcal{Q}}[f_\theta(\mathbf{x}+\mathbf{v})]), \\ &= \mathbb{E}_{\mathbf{v}\sim k,\mathbf{x}\sim\mathcal{Q}}\left[-\log(f_\theta(\mathbf{x}+\mathbf{v}))\frac{\hat{f}(\mathbf{x})}{a}\right] \\ &\quad + \log(\mathbb{E}_{\mathbf{v}\sim k,\mathbf{x}\sim\mathcal{Q}}[f_\theta(\mathbf{x}+\mathbf{v})]), \quad (14)\end{aligned}$$

where $a = \int_{\mathbb{R}^m}\hat{f}(\mathbf{x})\mathcal{Q}(\mathbf{x})d\mathbf{x}$ is the normalization constant from (4). In this form, both terms can use the same set of samples, thereby mitigating the issues mentioned above.

Relaxing the distribution-like constraint. Relaxing the assumption of scalar-valued fields to fit vector-valued signals such as color can be achieved straightforwardly by applying the loss on each channel separately. Doing so with

the naive form of the loss function in (13) would require separate evaluations of the network for each channel, but this too can be avoided by using the form in (14).

Signals that are not strictly positive, but which have a finite minimum value may also be reconstructed by adding an offset to their values such that their range becomes positive. Thanks to the linearity of the convolution operator, this offset can then simply be subtracted from the final result.

The final strategy we can apply is to allow non-positive filter kernels. Because we require compact kernels which integrate to a constant over the whole domain, we can not apply the constant offset trick to this problem. Instead, we can construct a second filter $j(\mathbf{x})$, such that $k(\mathbf{x})+j(\mathbf{x}) \geqslant 0$ for all $\mathbf{x}$. Then, by learning both $\hat{f} \ast (k+j)$ and $\hat{f} \ast j$, we can construct $\hat{f} \ast k$ as $\hat{f} \ast (k+j) - \hat{f} \ast j$.

4. Experiments with Euclidean Signals

We evaluate the our proposed method in comparison to Nsampi et al. [20], as well as the naive Monte Carlo baseline described in Sec. 3, on neural field filtering for signals defined over low dimensional, Euclidean spaces.

We experiment with filtering images, the results of which are reported in Tab. 1, as well as videos, which can be found in the Supplementary Material. We find that, as expected, the Monte Carlo baseline performs very well for small filter sizes, but becomes worse than our method at larger scales, despite the fact that we scale up the sample count proportional to the filter size. This problem only becomes worse as the number of samples required grows exponentially with the dimensionality of the signal.

Our method performs similarly to that of Nsampi et al. [20] for small kernels, but outperforms it for large ones. The multiple network invocations required for each pixel for their method also causes its inference speed, which we report in megapixels per second (MP/s), to be much lower, depending on the complexity of the filter.

As Nsampi et al. [20] is a zero-shot method, in that it is capable of inference on different filters without retraining, we also perform an experiment comparing this ability to our method using a learned *space* of filters, i.e. a network trained with randomly sampled filter scales and the network modified to accept the desired scale as an input. As shown in Tab. 2, our method gives high quality results in such a case without extra training time or network capacity.

Because the authors' code was not yet available at the time we performed experiments, we re-implemented the method described in [20] as part of the same JAX-based [5] codebase as our method. All experiments are performed with exactly the same network architecture (detailed in the Supplementary Material), data pipeline, and software+hardware environment, with only the minimum changes between runs necessary to implement the various loss functions. To ensure we do not unfairly bias the test

σ	Method	PSNR	SSIM	LPIPS	Iterations	Inference Speed
0.02	Monte Carlo	**40.9**	**0.982**	**0.730**	185373	639.5 MP/s
	Nsampi et al. [20]	37.1	0.919	0.767	25972	100.7 MP/s
	Ours	38	0.97	0.734	329720	**674.5 MP/s**
0.04	Monte Carlo	**45.4**	**0.992**	0.710	48904	676.1 MP/s
	Nsampi et al. [20]	42.5	0.987	**0.708**	25983	103.6 MP/s
	Ours	44.2	**0.992**	0.710	328550	**700.0 MP/s**
0.08	Monte Carlo	**49.5**	0.997	**0.586**	14994	658.1 MP/s
	Nsampi et al. [20]	45.8	**0.998**	0.596	25995	103.9 MP/s
	Ours	48	**0.998**	0.589	329270	**684.6 MP/s**
0.16	Monte Carlo	46.6	0.998	**0.499**	5561	**680.9 MP/s**
	Nsampi et al. [20]	46.1	**0.999**	0.507	25992	106.5 MP/s
	Ours	**50.8**	**0.999**	0.502	329060	663.5 MP/s

Table 1. **Image filtering** – Here we apply filters to images at various scales (σ). Several filters are used, with the reported metrics averaged over all image/filter pairs (see the Supplementary Material for a list), and reconstruction metrics only evaluated within the valid region of the convolution to avoid artifacts due to different handling of boundary conditions. Training for all jobs is limited to five minutes, and evaluation is done at whatever iteration it reaches in that time. Ground truth filtered images are computed via discrete Fourier transform.

Method	PSNR	SSIM	LPIPS	Iterations	Inference
Nsampi et al. [20]	**43.5**	0.982	0.647	51913	105.0 MP/s
Ours	42.6	**0.984**	**0.634**	628219	685.4 MP/s

Table 2. **Filter spaces** – A single network is trained to apply filters with width varying from 2% to 16% of the image width, with the reported metrics averaged over a log-uniform sampling of scales, where reconstruction metrics are only evaluated within the valid region of the convolution to avoid artifacts due to different handling of boundary conditions. Training for all jobs is limited to ten minutes, and evaluation is done at whatever iteration it reaches in that time. Ground truth filtered images are computed via discrete Fourier transform.

in favor of our method, we tune all hyperparameters using the loss function from [20], and re-use these values for runs with our method without further tuning.

5. Extending beyond Euclidean space

5.1. Filtering neural fields on smooth manifolds

In this section, we consider the more general case where the input manifold $\mathbf{M}$ is not necessarily Euclidean, *i.e.*, $\mathbf{M} \neq \mathbb{R}^m$. To do this, we will need to identify the properties and requirements of, and requirements for, convolution of functions on non-Euclidean domains. With these definitions in place, we can repeat the derivation of our loss function without the Euclidean assumption.

Convolution on homogeneous spaces. In this more general case, we must be careful with the coordinates $\mathbf{x} \in \mathbf{M}$, as they can not be assumed to admit familiar operations such as addition or multiplication. At the minimum, we require a notion of convolution between functions on $\mathbf{M}$, so we will begin by endowing $\mathbf{M}$ with a convolution operator.

We first start by assuming the existence of a Lie group $\mathbf{G}$

of which $\mathbf{M}$ is a homogeneous space. This means that $\mathbf{G}$ acts transitively on $\mathbf{M}$, or in other words, for any $\mathbf{a}, \mathbf{b} \in \mathbf{M}$, there exists some $\mathsf{g} \in \mathbf{G}$ such that $\mathsf{g}\mathbf{a} = \mathbf{b}$, where $\mathsf{g}\mathbf{a}$ denotes the action of g on a. A concrete example would be $\mathbf{G} = \mathrm{SO}(3)$ (3D rotations) and $\mathbf{M} = S^2$ (a 3D sphere): in this case, any point on the 2-sphere can be transformed to any other by the application of a 3D rotation.

Given this structure, the convolution of two real-valued functions f and k on $\mathbf{M}$ is a mapping $\mathcal{X}(\mathbf{M}) \times \mathcal{X}(\mathbf{M}) \rightarrow \mathcal{X}(\mathbf{G})$[1] and is defined as:

$$(f * k)(\mathsf{g}) = \int_{\mathbf{M}} f(\mathsf{g}\eta(\mathbf{v})^{-1}\mathbf{o})k(\mathbf{v})d\mu(\mathbf{v}), \qquad (15)$$

where $\mathbf{o}$ is an "origin" element[2] in $\mathbf{M}$, $\eta : \mathbf{M} \rightarrow \mathbf{G}$ is a lifting function such that $\eta(\mathbf{x})\mathbf{o} = \mathbf{x}$, and μ is the Haar measure on $\mathbf{M}$. This definition has the inconvenient property that the function resulting from the convolution of two signals on $\mathbf{M}$ is defined over $\mathbf{G}$. As such, in cases where $\mathbf{M} \ncong \mathbf{G}$ we will need to either lift the domain of our field to $\mathbf{G}$, or restrict our consideration to functions whose convolution will be interpretable as functions on $\mathbf{M}$.

More specifically, let $\mathbf{H}$ be a subgroup of $\mathbf{G}$ such that the quotient space $\mathbf{G}/\mathbf{H} \cong \mathbf{M}$. We say that a function ϕ on $\mathbf{G}$ is $\mathbf{H}$-invariant if $\phi(\mathsf{g}) = \phi(\mathsf{gh})$ for all $\mathsf{h} \in \mathbf{H}$ and all $\mathbf{G}$. What we want is for the convolution $f * k$ to be $\mathbf{H}$-invariant. When this holds, the actions $\mathsf{g} \in \mathbf{G}$ which satisfy $\mathsf{g}\mathbf{a} = \mathbf{b}$ for any $\mathbf{a}, \mathbf{b} \in \mathbf{M}$ will all have the same value of $(f * k)(\mathsf{g})$. This can be achieved by restricting consideration to functions k which are $\mathbf{H}$-symmetric about $\mathbf{o}$, *i.e.*, k s.t.:

$$k(\mathbf{x}) = k(\mathsf{g}\mathbf{x}) \;\; \forall \mathsf{g} \in \mathbf{G} \quad \text{where } \mathsf{g}\mathbf{o} = \mathbf{o}. \qquad (16)$$

To again use $\mathbf{G} = \mathrm{SO}(3)$, $\mathbf{M} = S^2$ as an example, spherical convolution requires rotationally symmetric kernels, because $\mathrm{SO}(3)/\mathrm{SO}(2) \cong S^2$. For the case where $\mathbf{M} \cong \mathbf{G}$, $\mathbf{H}$ will be the trivial group, and the $\mathbf{H}$-symmetry of k will always be trivially satisfied. The resulting expression for $\mathbf{H}$-invariant convolution expressed as a function on $\mathbf{M}$ is:

$$(f * k)(\mathbf{x}) = \int_{\mathbf{M}} f(\eta(\mathbf{x})\eta(\mathbf{v})^{-1}\mathbf{o})k(\mathbf{v})d\mu(\mathbf{v}). \qquad (17)$$

We refer the reader to [24], [13], and [6], which provide more in-depth introductions to this topic.

Loss function for non-Euclidean signals. When generalized to non-Euclidean manifolds, our loss function is:

$$\mathcal{L}_{\text{MLE}} = \mathbb{E}_{\mathbf{v}\sim k, \mathbf{x}\sim\hat{p}}[-\log(f_\theta(\eta(\mathbf{x})\mathbf{v}))] + \log(\mathbb{E}_{\mathbf{v}\sim k, \mathbf{x}\sim\mathcal{Q}}[f_\theta(\eta(\mathbf{x})\mathbf{v})]). \qquad (18)$$

This form is equivalent to (13) in the case where $\mathbf{M}{=}\mathbb{R}^m$ and the group action is vector addition. The same re-weighting and generalization strategies described in Sec. 3.3 are also applicable here. For the full derivation of this form, please refer to the Supplementary Material.

[1] $\mathcal{X}(A)$ here denotes a mapping $A \rightarrow \mathbb{R}$.

[2] An arbitrary, but consistent, point of reference on the manifold.

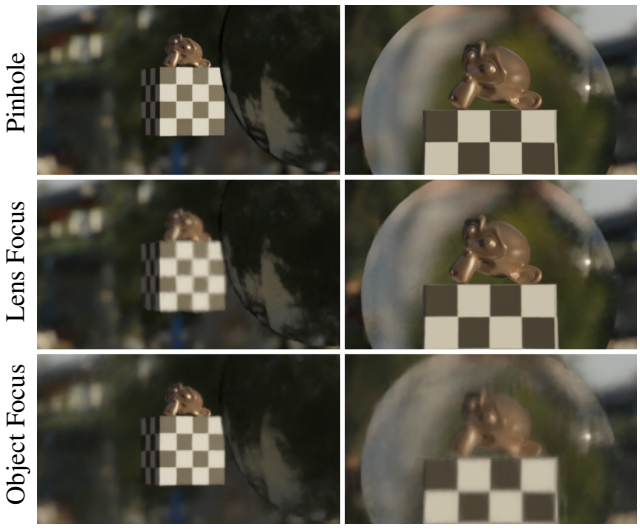

Figure 3. **Light field network** – We show how our filtering method is able to train a depth of field-aware light field network from pinhole ground truth images such that complex optical effects, such as refraction in this scene with an object behind a lens, are correctly modelled. In this figure, all images are rendered from the same network, with aperture and focus settings varied between the rows to focus on the whole scene (pinhole), the object through the lens, and the object observed directly.

5.2. Experiments on non-Euclidean signals

In this section, we demonstrate the application of our method to non-Euclidean filtering problems.

Light field filtering. Despite the name, this section deals with both Light Field Networks and Neural Radiance fields. Light field networks are a direct neural parameterization of the *Plenoptic function* – the function which defines the intensity of observed light at every point, and in every direction in space. "Radiance fields", by contrast, are a parameterization of the light *leaving* every point in space in every direction, as well as a density value, which describes how material at that point in space blocks light from passing through it. Through the application of volume rendering, this representation can also be interpreted as a light field, as both are mappings from ray parameters to light or color.

The manifold over which a light field is defined can be either be $\mathbb{R}^3 \times S^2$ in the case of the full 5D Plenoptic function, or $\mathrm{Gr}(2, 4)$ (lines in projective 3D space) in the 4D case where the content of the scene is observed from outside its convex hull. In either case, this manifold of rays is non-Euclidean, and any filtering operations performed on it must be formulated accordingly.

To demonstrate the capability of our proposed method to handle such non-trivial manifolds, we use it to implement

Method	Test Set Reconstruction (PSNR / SSIM / LPIPS)							
	Garden	Bicycle	Stump	Counter	Room	Bonsai	Kitchen	Average
MipNeRF360 [2]	**27.0 / 0.813 / 0.170**	**24.4 / 0.685 / 0.301**	**26.4 / 0.744 / 0.261**	**29.6 / 0.894 / 0.204**	31.6 / **0.913 / 0.211**	**33.5 / 0.941 / 0.176**	**32.2 / 0.920 / 0.127**	**29.2 / 0.844 / 0.207**
Ours	26.4 / 0.762 / 0.249	24.0 / 0.582 / 0.442	26.0 / 0.691 / 0.383	28.9 / 0.847 / 0.330	**31.7** / 0.896 / 0.302	32.7 / 0.917 / 0.265	31.6 / 0.900 / 0.189	28.8 / 0.806 / 0.309

Table 3. **Mip-NeRF 360** – We evaluate how well our models, trained for thin lens depth of field modelling, perform in reconstructing the test images when evaluated with pinhole ray distributions. Aside from the loss function and learning rate, all settings are kept the same as in Mip-NeRF 360. Despite our method learning to render from an *entire distribution* of lens models (see the Supplementary Material for qualitative video), it *only* loses on average 0.4 dB in PSNR for pinhole rendering.

lens modelling for light fields. This works by defining the filter kernel as distribution of rays contributing to a pixel in an image formed by a thin lens. The group action which defines this convolution is $\mathrm{SE}(3)$ (3D rigid transformations), and the operation can be understood as an average of color over individual rays being perturbed by rigid transformations of space. For lens models which are radially symmetric about the optical axis, this distribution satisfies the $\mathbf{H}$-invariance property described in Sec. 5.1. Other distributions, such as those modelling non-symmetric lenses or motion blur, fail this test and thus require the light field to be lifted to $\mathrm{SE}(3)$ by adding an extra input to the network which controls the "roll" axis of the camera, which is otherwise ignored. We implement both the symmetric and lifted versions; see the Supplementary Material for more details.

We build our implementation on two different architectures: for NeRF, we use the "multinerf" [18] implementation of Mip-NeRF 360 [2], and for light fields, we use a simple Multilayer Perceptron conditioned on the Plücker coordinates of input rays, based on that proposed by Sitzmann et al. [27] (full details in the Supplementary Material).

In our NeRF experiments, we replace the standard photometric reconstruction loss with the formulation in (14), where the kernels k are drawn from a distribution of thin lens models with varying apertures and focal distances. We leverage the network's integrated positional encoding [1] to condition the frequency bias of the field, and change the standard conical model of ray shape to a converging Gaussian beam model. This strategy makes it easy to supervise a *space* of different filters ranging from pinhole to wide aperture, as the ray distribution can be controlled in the same way that Mip-NeRF provides controllable anti-aliasing. We compare the test reconstruction accuracy of these models to Mip-NeRF 360 in Tab. 3.

A few prior works have proposed to explicitly model lens effects in NeRF, typically by some form of numerical or Monte Carlo approximation at inference [15, 23, 33]. The strategy of leveraging integrated positional encoding for depth of field has also been explored [31]. We forego comparison to these methods, as our goal is to establish a more general framework for filtering that does not increase inference cost, and is not specific to a particular model or application.

To show that our method generalizes beyond just what can be achieved with modifications of volume rendering, we train a light field network on a scene containing a refractive lens. Standard NeRF models are unable to faithfully represent such optical phenomena [21, 32, 38], but pure light fields do not suffer this restriction. As we show in Fig. 3, we are able to construct a light field from pinhole-rendered ground truth images that not only allows controllable depth of field rendering, but correctly models the ability of the virtual lens to focus *through* the lens in the scene on an object that would otherwise be out of focus.

6. Conclusions

We have presented a novel method for incorporating efficient, non-Euclidean filtering operations into the training of neural field models. The loss function we propose can be applied to a number of problems across various areas where neural fields are used, and can enable new filtering applications for non-Euclidean neural fields including NeRF, light fields, and other ray-based methods. It also has the potential to provide fresh perspectives on problems where complex manifolds like $\mathrm{SE}(3)$ are used as input to a network, for example by making field evaluations uncertainty-aware by convolving with the measurement error distribution.

Limitations and future work. While our proposed method is applicable to a wide range of filtering applications, there are cases for which it is not well-suited. In particular, applications which require arbitrary filters *not known at training* to be applied at inference would be better suited to a zero-shot approach like the method of Nsampi et al. [20]. The ability of our method to learn spaces of filters in a single model is also limited by the amenability of the network to conditioning on the filter parameters, and more complex filter families will degrade reconstruction accuracy as more network capacity is consumed by learning the more complex space. The investigation of neural field architectures which are less susceptible to this constraint would be a promising avenue of future work.

Acknowledgements. This work was supported in part by the Natural Sciences and Engineering Research Council of Canada (NSERC) Discovery Grant [2023-05617], NSERC Collaborative Research and Development Grant, the SFU Visual Computing Research Chair, Google, Digital Research Alliance of Canada, and Advanced Research Computing at the University of British Columbia. We thank Sneha Sambandam for her helpful comments and feedback.

References

[1] 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 *ICCV*, pages 5855–5864, 2021. 2, 8

[2] Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. Mip-NeRF 360: Unbounded anti-aliased neural radiance fields. *CVPR*, 2022. 2, 8

[3] Jonathan T Barron, Ben Mildenhall, Dor Verbin, Pratul P Srinivasan, and Peter Hedman. Zip-nerf: Anti-aliased grid-based neural radiance fields. In *ICCV*, pages 19697–19705, 2023. 2

[4] Mark Boss, Raphael Braun, Varun Jampani, Jonathan T. Barron, Ce Liu, and Hendrik Lensch. NeRD: Neural reflectance decomposition from image collections. In *ICCV*, pages 12684–12694, 2021. 2

[5] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018. 6

[6] Michael Bronstein, Joan Bruna, Taco Cohen, and Petar Veličković. Geometric deep learning. *African Master in Machine Intelligence*, 2022. 2, 7

[7] Zhiqin Chen and Hao Zhang. Learning implicit fields for generative shape modeling. In *CVPR*, 2019. 2

[8] Boyang Deng, John P. Lewis, Timothy Jeruzalski, Gerard Pons-Moll, Geoffrey Hinton, Mohammad Norouzi, and Andrea Tagliasacchi. NASA: neural articulated shape approximation. In *ECCV*, pages 612–628. Springer, 2020. 3

[9] Marc Finzi, Samuel Stanton, Pavel Izmailov, and Andrew Gordon Wilson. Generalizing convolutional neural networks for equivariance to lie groups on arbitrary continuous data. In *ICML*, pages 3165–3176, 2020. 2

[10] Ruohan Gao, Yen-Yu Chang, Shivani Mall, Li Fei-Fei, and Jiajun Wu. ObjectFolder: A dataset of objects with implicit visual, auditory, and tactile representations. *arXiv preprint arXiv:2109.07991*, 2021. 2

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

[12] Robert V. Hogg, Joseph W. McKean, and Allen T. Craig. *Introduction to Mathematical Statistics (6th Edition)*. Prentice Hall, 2004. 3, 4

[13] Risi Kondor and Shubhendu Trivedi. On the generalization of equivariance and convolution in neural networks to the action of compact groups. In *ICML*, pages 2747–2755, 2018. 2, 7

[14] David B. Lindell, Dave Van Veen, Jeong Joon Park, and Gordon Wetzstein. BACON: Band-limited coordinate networks for multiscale scene representation. In *CVPR*, pages 16252–16262, 2022. 1, 2

[15] Li Ma, Xiaoyu Li, Jing Liao, Qi Zhang, Xuan Wang, Jue Wang, and Pedro V Sander. Deblur-NeRF: Neural radiance fields from blurry images. In *CVPR*, pages 12861–12870, 2022. 8

[16] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *CVPR*, pages 4460–4470, 2019. 2

[17] 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. *Communications of the ACM*, 65(1):99–106, 2021. 1, 2

[18] Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, Peter Hedman, Ricardo Martin-Brualla, and Jonathan T. Barron. MultiNeRF: A Code Release for Mip-NeRF 360, Ref-NeRF, and RawNeRF, 2022. 8

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

[20] Ntumba Elie Nsampi, Adarsh Djeacoumar, Hans-Peter Seidel, Tobias Ritschel, and Thomas Leimkühler. Neural field convolutions by repeated differentiation. *ACM TOG*, 2023. 1, 3, 4, 6, 8

[21] Jen-I Pan, Jheng-Wei Su, Kai-Wen Hsiao, Ting-Yu Yen, and Hung-Kuo Chu. Sampling neural radiance fields for refractive objects. In *SIGGRAPH Asia 2022 Technical Communications*, pages 1–4, 2022. 8

[22] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. DeepSDF: Learning continuous signed distance functions for shape representation. In *CVPR*, pages 165–174, 2019. 2

[23] Stanislav Pidhorskyi, Timur Bagautdinov, Shugao Ma, Jason Saragih, Gabriel Schwartz, Yaser Sheikh, and Tomas Simon. Depth of field aware differentiable rendering. *ACM TOG*, 41 (6):1–18, 2022. 8

[24] François Rouvière et al. *Symmetric spaces and the Kashiwara-Vergne method*. Springer, 2014. 7

[25] Mehdi S.M. Sajjadi, Henning Meyer, Etienne Pot, Urs Bergmann, Klaus Greff, Noha Radwan, Suhani Vora, Mario Lučić, Daniel Duckworth, Alexey Dosovitskiy, et al. Scene representation transformer: Geometry-free novel view synthesis through set-latent scene representations. In *CVPR*, pages 6229–6238, 2022. 2

[26] Liyue Shen, John Pauly, and Lei Xing. NeRP: implicit neural representation learning with prior embedding for sparsely sampled image reconstruction. *IEEE Transactions on Neural Networks and Learning Systems*, 2022. 2

[27] Vincent Sitzmann, Semon Rezchikov, Bill Freeman, Josh Tenenbaum, and Fredo Durand. Light field networks: Neural scene representations with single-evaluation rendering. *NeurIPS*, 34:19313–19325, 2021. 1, 2, 8

[28] Mohammed Suhail, Carlos Esteves, Leonid Sigal, and Ameesh Makadia. Light field neural rendering. In *CVPR*, pages 8269–8279, 2022. 2

[29] Peder Bergebakken Sundt and Theoharis Theoharis. Marf: The medial atom ray field object representation. *Computers & Graphics*, 115:122–136, 2023. 1

[30] Yee Whye Teh, Max Welling, Simon Osindero, and Geoffrey E. Hinton. Energy-based models for sparse overcomplete representations. *Journal of Machine Learning Research*, 4(Dec):1235–1260, 2003. 5

[31] Yinhuai Wang, Shuzhou Yang, Yujie Hu, and Jian Zhang. NeRFocus: Neural radiance field for 3d synthetic defocus. *arXiv preprint arXiv:2203.05189*, 2022. 8

[32] Ziyu Wang, Wei Yang, Junming Cao, Qiang Hu, Lan Xu, Junqing Yu, and Jingyi Yu. NeReF: Neural refractive field for fluid surface reconstruction and rendering. In *2023 IEEE International Conference on Computational Photography (ICCP)*, pages 1–11. IEEE, 2023. 8

[33] Zijin Wu, Xingyi Li, Juewen Peng, Hao Lu, Zhiguo Cao, and Weicai Zhong. DoF-NeRF: Depth-of-field meets neural radiance fields. In *ACM MM*, pages 1718–1729, 2022. 8

[34] 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 *Comput. Graph. Forum*, pages 641–676. Wiley Online Library, 2022. 2, 3

[35] Dejia Xu, Peihao Wang, Yifan Jiang, Zhiwen Fan, and Zhangyang Wang. Signal processing for implicit neural representations. In *NeurIPS*, 2022. 1, 3

[36] Guandao Yang, Sagie Benaim, Varun Jampani, Kyle Genova, Jonathan Barron, Thomas Funkhouser, Bharath Hariharan, and Serge Belongie. Polynomial neural fields for subband decomposition and manipulation. *NeurIPS*, 35:4401–4415, 2022. 2

[37] Guangming Zang, Ramzi Idoughi, Rui Li, Peter Wonka, and Wolfgang Heidrich. IntraTomo: self-supervised learning-based tomography via sinogram synthesis and prediction. In *ICCV*, pages 1960–1970, 2021. 2

[38] Yifan Zhan, Shohei Nobuhara, Ko Nishino, and Yinqiang Zheng. NeRFrac: Neural radiance fields through refractive surface. In *ICCV*, pages 18402–18412, 2023. 8

[39] Kai Zhang, Fujun Luan, Qianqian Wang, Kavita Bala, and Noah Snavely. PhySG: Inverse rendering with spherical gaussians for physics-based material editing and relighting. In *CVPR*, pages 5453–5462, 2021. 2