Neural Gaussian Scale-Space Fields

FELIX MUJKANOVIC, Max-Planck-Institut für Informatik, Germany
NTUMBA ELIE NSAMPI, Max-Planck-Institut für Informatik, Germany
CHRISTIAN THEOBALT, Max-Planck-Institut für Informatik, Germany
HANS-PETER SEIDEL, Max-Planck-Institut für Informatik, Germany
THOMAS LEIMKÜHLER, Max-Planck-Institut für Informatik, Germany

Fig. 1. We learn neural fields that capture continuous, anisotropic Gaussian scale spaces. Given a training signal, such as an image or geometry (left), we learn a neural field representation that allows continuous Gaussian smoothing (center). Crucially, this representation is learned self-supervised, i.e., without ever filtering the training signal. Our scale spaces are continuous in all parameters, including arbitrary covariance matrices that allow anisotropic filtering (right).

Gaussian scale spaces are a cornerstone of signal representation and processing, with applications in filtering, multiscale analysis, anti-aliasing, and many more. However, obtaining such a scale space is costly and cumbersome, in particular for continuous representations such as neural fields. We present an efficient and lightweight method to learn the fully continuous, anisotropic Gaussian scale space of an arbitrary signal. Based on Fourier feature modulation and Lipschitz bounding, our approach is trained self-supervised, i.e., training does not require any manual filtering. Our neural Gaussian scale-space fields faithfully capture multiscale representations across a broad range of modalities, and support a diverse set of applications. These include images, geometry, light-stage data, texture anti-aliasing, and multiscale optimization.

ACM Reference Format:

1 INTRODUCTION

Continuous neural representations, so-called neural fields, are becoming ubiquitous in visual-computing research and applications [Tewari et al. 2022; Xie et al. 2022]. At their core, they map coordinates to signal values using a neural network. The generality, compactness, and malleability of this continuous data structure make them a popular choice for representing a broad variety of modalities. For example, neural fields have been used to represent geometry [Park et al. 2019], images [Stanley 2007], radiance fields [Mildenhall et al. 2020], flow [Park et al. 2021], reflectance [Gargan and Neelamkavil 1998], and much more.

In its basic form, a trained neural field allows querying the original signal value for a given coordinate. Oftentimes, however, this functionality is not sufficient: Many important use cases require low-pass filtered versions of the signal, arising from a custom band-limiting kernel that might be spatially-varying and even anisotropic. For example, a downstream application might require querying the signal at different scales [Marr and Hildreth 1980; Starck et al. 1998].
We evaluate the accuracy and applicability of our approach on a broad variety of tasks and modalities. This includes anisotropic smoothing of images and geometry; (pre-)filtering of textures and light-stage data; spatially varying filtering, and multiscale optimization. We further analyze the interplay between Fourier features and Lipschitz-bounded MLPs to elucidate the combined effect of the central ingredients of our approach.

In summary, our contributions are:

- A novel approach for learning a fully continuous, anisotropic Gaussian scale space in a general-purpose neural field representation.
- An effective and efficient training methodology to achieve this goal self-supervised, i.e., without the requirement to filter the training data.
- The application and careful evaluation of our method on a spectrum of relevant modalities and tasks.

2 RELATED WORK

Here, we review related work on classical and neural multiscale representations (Sec. 2.1), the use of Fourier features in neural fields (Sec. 2.2), and Lipschitz bounds in deep learning (Sec. 2.3). For a comprehensive overview of neural fields in visual computing, we refer to recent surveys [Tewari et al. 2022; Xie et al. 2022].

2.1 Multiscale Signal Representations

Representing a signal at multiple scales has a long history in signal processing and visual computing, with the concept of scale spaces [Iijima 1959; Koenderink 1984; Lindeberg 2013; Witkin 1987] at the center of attention. Many different scale spaces can be constructed from a signal [Dorst and Van den Boomgaard 1994; Florack et al. 1995; Weickert 1998] using a rigorous axiomatic foundation [Lindeberg 1997]. Of special interest is the linear form, i.e., the convolution of the signal with a family of Gaussian kernels, as it exhibits a number of useful and well-studied properties, such as predictable behavior after differentiation [Babaud et al. 1986].

Scale spaces are typically constructed using various flavors of discretization. The convolution of a discrete signal with a discretized Gaussian kernel can be executed using cubature, but comes at high computational cost, especially in higher dimensions. Acceleration strategies involve the discrete Fourier transform [Brigham 1988] or exploiting the separability of Gaussians [Geusebroek et al. 2003]. In low dimensions, pyramidal structures (MIP mapping) [Burt 1981; Williams 1983] are even more efficient. Here, the isotropic scale parameter, i.e., the pyramid level, is also discrete. Anisotropic filtering using pyramids (RIP mapping) exists [Simoncelli and Freeman 1995], but comes at the cost of an additional coarse discretization of filter orientation, which can be hidden using carefully designed steerable filters [Freeman and Adelson 1991]. A different line of work has successfully explored signal representations using a discrete set of multiscale basis functions [Daubechies 1988; Guo et al. 2006; Mallat 1989]. In contrast to all these works, our approach is fully continuous in all dimensions.

Continuous representations impose significant challenges for multiscale techniques, and a dominant strategy for filtering is stochastic multi-sampling [Barron et al. 2023; Hermosilla et al. 2018; Ma...
et al. 2022; Shocher et al. 2020; Wang et al. 2018]. Such Monte Carlo approaches require a high number of samples to avoid objectionable noise. Sample count can be significantly reduced by relying on differentiation and integration properties of convolutions [Nampi et al. 2023]. However, learning the required integral representation is costly, high-quality Gaussian filtering still requires a substantial number of network evaluations, and general anisotropic filtering requires pre-computing many kernel shapes. Our neural fields are easy to train and allow arbitrary anisotropic Gaussian filtering using just a single forward pass. A different strategy relies on approximating continuous filtering using a learned linear combination of derivatives of the signal obtained via automatic differentiation [Xu et al. 2022]. Different from our solution, this approach only supports small filter kernels.

Pre-filtering for anti-aliasing in continuous neural representations has recently received a lot of attention [Barron et al. 2021, 2022; Hu et al. 2023; Nam et al. 2023]. Similar to our approach, these solutions employ carefully crafted inductive biases that help learn a multiscale representation. However, they rely on supervision across scales, such as images of scene objects captured at different distances. In contrast, our method allows to learn a full scale space from a single-scale supervision signal, thereby significantly extending its applicability to a broad range of signals and modalities.

Strong architectural inductive biases allow training neural networks with intermediate activations that represent progressively band-limited versions of the learned signal [Fathony et al. 2020; Lindell et al. 2022; Shekarforoush et al. 2022], with applications in coarse-to-fine learning [Karras et al. 2018; Xiangli et al. 2022]. This leads to a discretization of scales and typically only allows isotropic filtering with a sinc-kernel. Extending this scheme to the anisotropic case is possible [Yang et al. 2022], but, similar to RIP mapping, it requires an additional coarse discretization of filter orientation, limiting this approach to low dimensions. The discretization of scales can be combined with neuroexplicit architectures, e.g., via spatial discretization or subdivision [Saragadam et al. 2022; Takikawa et al. 2022], and many corresponding domain-specific solutions exist [Chen et al. 2021, 2022a; Gauthier et al. 2022; Kuznetsov et al. 2021; Paz et al. 2022; Takikawa et al. 2021; Xu et al. 2021; Zhuang et al. 2023]. Yet, none of them allows fully continuous, arbitrary, anisotropic Gaussian filtering.

2.2 Fourier Features in Neural Fields

With early applications in time series analysis and representation learning [Kazemi et al. 2019; Vaswani et al. 2017; Xu et al. 2019], Fourier features [Rahimi and Recht 2007] are now a popular tool for learning neural-field representations of signals [Mildenhall et al. 2020]. Also referred to as positional encoding, their ability to map coordinates to latent features of different frequencies is an effective remedy for the spectral bias of neural networks [Rahaman et al. 2019]. Tancik et al. [2020] have analyzed the properties of such an encoding for neural fields using the neural tangent kernel [Jacot et al. 2018], and propose the use of normally distributed frequency vectors. Many applications rely on carefully dampened Fourier features to increase training stability [Hertz et al. 2021; Lin et al. 2021; Park et al. 2021; Yang et al. 2023], or to obtain a multiscale representation given a multiscale supervision signal [Barron et al. 2021, 2022]. Further, the analytical structure of Fourier features has been exploited for alias-free image synthesis [Karras et al. 2021]. We employ dampened Fourier features with carefully chosen frequency vectors as well, and combine this encoding with a Lipschitz-bounded network to obtain a Gaussian scale space.

Fourier features have been explored in different architectural variants. Examples of this scheme include periodic activation functions [Mehta et al. 2021; Sitzmann et al. 2020], Wavelet-style spatiotemporal encodings [Wu et al. 2023], or the modulation of unstructured representations based on radial basis functions [Chen et al. 2023b]. Different from our approach, the goal of these works is to improve single-scale reconstruction quality of complex signals.

2.3 Lipschitz Networks and Matrix Parameterizations

Neural networks with guaranteed Lipschitz bounds have numerous applications, such as robustness [Cisse et al. 2017; Hein and Andriushchenko 2017], smooth interpolation [Liu et al. 2022], and generative modeling [Arjoysky et al. 2017]. Technically, a desired Lipschitz constant can be enforced on the level of individual weight matrices. Corresponding methods can be divided into two classes.

The first class of methods relies on variants of projected gradient descent, where weight matrices are projected towards the closest feasible solution for each optimization step [Szegedy et al. 2013]. This can be done using spectral normalization [Behrmann et al. 2019; Gouk et al. 2021; Miyato et al. 2021; Yang et al. 2021; Yoshida and Miyato 2017], or using more sophisticated projections [Cisse et al. 2017] relying on orthonormalization [Bjurck and Bowie 1971]. The second class of methods reparameterize the weight matrices such that an unconstrained optimization can be applied [Anil et al. 2019; Liu et al. 2022]. Our method relies on this approach, as we observe that it leads to controllable training dynamics, but the choice of matrix parameterization is crucial for numerical stability.

The singular value decomposition is a convenient tool in this context [Mathiasen et al. 2020; Zhang et al. 2018b]. As it requires a parameterization of orthogonal matrices, ad-hoc parameterizations for special orthogonal matrices have been considered [Arjoysky et al. 2016; Helfrich et al. 2018; Huang et al. 2018; Jing et al. 2017]. More general solutions rely on Householder reflections [Mathiasen et al. 2020; Mhammedi et al. 2017; Zhang et al. 2018b], but they exhibit unfavourable properties when used within an optimization loop. We rely on matrix exponentials [Hyland and Ratsch 2017; Lezcano-Casado and Martinez-Rubio 2019], which have been shown to outperform other approaches [Golinski et al. 2019]. Yet, to the best of our knowledge, we are the first to use this set of techniques in the context of Lipschitz-bounded neural networks.

3 PRELIMINARIES

Here, we introduce concepts relevant to our approach. We first establish our notation for signals and fields, before reviewing technical background on Gaussian scale spaces, Fourier features, and Lipschitz continuity.

Signals and fields. We are concerned with arbitrary continuous signals \( f \in \mathbb{R}^{d_i} \rightarrow \mathbb{R}^{d_o} \), where \( d_i \) and \( d_o \) are typically rather small. This generic formulation captures many modalities in visual
A very practical benefit of this structure is that it provides precise control over the frequency content of a signal. Yet, obtaining a scale-space representation is costly, as it requires executing a continuous $d_i$-dimensional integral for each combination of $\mathbf{x}$ and $\Sigma$ – an operation for which, in general, no closed-form solution exists.

Fourier features. Neural networks exhibit an intrinsic spectral bias towards "simple" solutions [Rahaman et al. 2019], which makes it challenging for basic fully-connected architectures to learn high-frequency content. An established remedy is to first featureize the input coordinate $\mathbf{x}$ using a fixed mapping $\gamma : \mathbb{R}^{d_i} \to \mathbb{R}^{2m}$, based on sinusoids of $m$ different frequencies [Mildenhall et al. 2020; Rahimi and Recht 2007; Tancik et al. 2020]:

$$
\gamma(\mathbf{x}) = \begin{pmatrix}
\lambda_1 \cos (2\pi a_1^T \mathbf{x}) \\
\lambda_1 \sin (2\pi a_1^T \mathbf{x}) \\
\vdots \\
\lambda_m \cos (2\pi a_m^T \mathbf{x}) \\
\lambda_m \sin (2\pi a_m^T \mathbf{x})
\end{pmatrix}.
$$

In this positional encoding, $a_i \in \mathbb{R}^{d_i}$ are frequency vectors and $\lambda_i \in \mathbb{R}$ are weights of the corresponding Fourier feature dimensions. Small offsets in $\mathbf{x}$ lead to rapid changes of $\gamma(x)$ for high frequencies $a_i$. Therefore, feeding $\gamma(\mathbf{x})$ instead of the raw $\mathbf{x}$ into a neural network effectively lifts the burden of creating high frequencies from the network, resulting in higher-quality fits of complex signals.

Lipschitz continuity. A Lipschitz-continuous function is limited in how fast it can change. Formally, for this class of functions, there exists a Lipschitz bound $c \geq 0$ such that

$$
\|f(x_1) - f(x_2)\|_p \leq c\|x_1 - x_2\|_p
$$

for all possible $x_1$ and $x_2$ and an arbitrary choice of $p$. Intuitively, moving a certain distance in the function’s domain is guaranteed to result in a bounded change of function values.

If $f$ is implemented using a fully-connected network (MLP) with $l$ layers and 1-Lipschitz activation functions (e.g., ReLU), an upper Lipschitz bound is given by [Gouk et al. 2021]

$$
c = \prod_{k=1}^{l} \|W_k\|_p,
$$

where $W_k$ is the (trainable) weight matrix of the $k$'th network layer. Enforcing bounded weight-matrix norms effectively imposes a fixed, global constraint on how rapidly $f$ can change.

4 METHOD

We seek to learn a neural field $F(\mathbf{x}, \Sigma)$ that captures the full anisotropic Gaussian scale space $f_G(\mathbf{x})$ of a signal $f$, i.e., a family of Gaussian-smoothed signals with arbitary, anisotropic covariance $\Sigma$. We consider, both, coordinates $\mathbf{x}$ and covariance matrix $\Sigma$, continuous parameters, so that the field can be queried at any location using any Gaussian filter. Since computing $f_G$ via Eq. 1 is intractable for all but the simplest $f$, we learn $F(\mathbf{x}, \Sigma)$ self-supervised, i.e., we only rely on the original signal $f$.

To achieve this goal, we make a simple but far-reaching observation: Careful dampening of high-frequency Fourier features produces a low-pass filtered signal of high quality if the neural network representing the signal is Lipschitz-bounded. Based on this observation, we develop a novel paradigm that leverages the combined properties of modulated Fourier features and Lipschitz-continuous networks (Sec. 4.1). Our approach relies on a neural architecture with carefully designed constraints (Sec. 4.2), such that training can be performed using supervision from raw, unfiltered signal samples (Sec. 4.3). The emerging continuous filter parameters are uncalibrated, since we employ an efficient method that does not explicitly execute any Gaussian smoothing during training. Therefore, after training, we perform a lightweight calibration to enable precise filtering (Sec. 4.4).

4.1 Self-supervised Learning of Gaussian-smoothed Neural Fields

We consider an established neural architecture that consists of the composition of a positional encoding using Fourier features $\gamma$ (Eq. 2) with a multi-layer perceptron (MLP) $\Psi_\theta$:

$$
F(x) = \Psi_\theta (\gamma(x)).
$$

Here, $\theta$ represents the trainable network parameters, consisting of weight matrices $W_k$ and bias vectors $b_k$. Based on this setup, our approach fuses two techniques that are well-known in isolation, but, to the best of our knowledge, have not yet been systematically considered in combination: First, we employ Fourier feature modulation, i.e., we dampen high-frequency components of the positional encoding in Eq. 2 using custom, frequency-dependent
weights $\lambda_i$ [Barron et al. 2021; Hertz et al. 2021; Lin et al. 2021; Park et al. 2021]. Second, we enforce an upper Lipschitz bound of $\Psi_0$ [Gouk et al. 2021; Miyato et al. 2018; Szegedy et al. 2013]. We refer to such a bounded network as $\overline{\Psi}$. To understand how this construction can help learn a controllably smooth function from a raw signal, consider four different strategies for learning a signal in Fig. 3.

In Fig. 3a, we illustrate the basic setup of Eq. 5 without any modifications, i.e., with $\lambda_i = 1 \forall i$ and an unbounded MLP $\Psi_0$. Unsurprisingly, we observe that training $F$ on $f$ results in a faithful fit. Yet, we do not have any handle for creating smooth network responses here.

As a potential remedy, consider the setup in Fig. 3b, where we dampen the higher-frequency Fourier features of $\gamma$. Consistent with established findings in the literature [Mildenhall et al. 2020; Müller et al. 2022; Tancik et al. 2020], we observe that fitting quality degrades. Yet, this happens in an unpredictable way, leading to incoherent high-frequency spikes in $F$. This is because $\Psi_0$ – depending on factors such as signal complexity and network capacity – can compensate for missing input frequencies by forging a function with high gradients w.r.t. its inputs. Fourier feature modulation can help learn signals robustly when used progressively [Hertz et al. 2021; Lin et al. 2021], or facilitate learning of a multiscale representation when supervision across scales is available [Barron et al. 2021, 2022]. Yet, on its own, it is not a viable strategy for learning a smooth function from a raw supervision signal.

We now turn to an architecture with a Lipschitz-bounded MLP $\overline{\Psi}_0$, yet with unmodified Fourier features, depicted in Fig. 3c. We observe that the trained $F$ now indeed captures a smoother version of $f$. We seem to have achieved our goal; however, the Lipschitz bound $c$ needs to be fixed for training and is baked into the MLP.

While this is a useful property for robust training [Cisse et al. 2017; Hein and Andriushchenko 2017] or smooth interpolation [Liu et al. 2022], this strategy does not provide any control over the smoothing once the network is trained.

The above considerations motivate us to develop a new approach that combines frequency modulation with Lipschitz bounding, as shown in Fig. 3d. When dampening high-frequency Fourier features in this setup, the Lipschitz-bounded $\overline{\Psi}_0$ cannot compensate for the missing frequency content, since to turn the now low-frequency encoding into a high-frequency output, it would need to produce large gradient magnitudes w.r.t. the positional encoding. Instead, it is forced to learn an $F$ that matches the raw $f$ as closely as possible given frequency and gradient constraints. This form of “parameterized gradient limiting” through modulated Fourier features facilitates controllable smoothing (dashed colored lines Fig. 4).

While we intuitively expect some form of smoothing, the exact reconstruction qualities emerging from our strategy are not obvious. However, examining reconstructions on a broad variety of real-world signals and modalities, we make a surprising, yet crucial empirical observation: The emerging smoothing is a remarkably faithful approximation of Gaussian filtering. We extensively validate this claim in Sec. 5, but consider a rigorous theoretical justification beyond the scope of this work. In Fig. 4, we visually compare results $F$ obtained from our approach against the best-fitting Gaussian-smoothed versions $f_0$ of $f$ (solid grey curves).

The insights developed above suggest an effective and efficient procedure for learning a Gaussian scale space in a self-supervised fashion: First, we build an architecture following Eq. 5 with carefully sampled Fourier frequency vectors and a robustly Lipschitz-bounded neural network (Sec. 4.2). Second, we train the architecture with strategically dampened Fourier features using the original signal $f$.

Fig. 3. Four different strategies to learn a neural field $F$ from a signal $f$. $F$ takes a continuous coordinate $x$ as input, which is fed into a positional encoding $\gamma$ (Eq. 2) that produces a set of Fourier features using cosine (blue curves) and sine (red curves) functions of different frequencies. The resulting features serve as input to an MLP $\Psi_0$ thatregresses $f$. (a) The basic setup learns a faithful reconstruction of $f$ (the curves for $F$ and $f$ overlay completely), but does not allow any smoothing. (b) Modulating the Fourier features using custom weights $\lambda_i$ (yellow bars) tends to remove some high frequencies, but distorts the reconstruction in an unpredictable way (orange rectangles mark incoherent spikes in $F$). (c) Employing a Lipschitz-bounded MLP $\overline{\Psi}_0$ leads to smoothing, but it requires choosing a single fixed bound for training, lacking flexibility. (d) Our approach combines Fourier feature modulation with Lipschitz bounding to enable controllable smoothing.
for supervision to learn an entire continuous anisotropic Gaussian scale space (Sec. 4.3). Finally, we map dampening weights \( \lambda_i \) to Gaussian covariance \( \Sigma \) to enable precise filtering (Sec. 4.4).

### 4.2 Architecture
Following the reasoning developed in the previous section, we design our neural field as

\[
F(\mathbf{x}, \hat{\Sigma}) = \overline{\mathbf{W}_\theta} \left( \gamma(\mathbf{x}, \hat{\Sigma}) \right),
\]

which implements two modifications to the basic setup of Eq. 5. First, we extend the positional encoding \( \gamma \) to incorporate a pseudo-covariance matrix \( \hat{\Sigma} \) as an additional parameter, as detailed in Sec. 4.2.1. Second, we use a Lipschitz-bounded MLP \( \mathbf{W}_\theta \), the construction of which is explained in Sec. 4.2.2.

We emphasize that our formulation in Eq. 6 naturally supports spatially varying filtering, as \( \mathbf{x} \) and \( \hat{\Sigma} \) are independent inputs to the field.

#### 4.2.1 Fourier Features for Filtering
We are concerned with designing a variant of the positional encoding in Eq. 2 that facilitates high-quality filtering through feature dampening.

We observe that the distribution of frequencies \( a_i \) plays a crucial role in the process. Several strategies have been explored in the literature on neural field design. A popular approach relies on axis-aligned frequencies [Barron et al. 2021, 2022; Mildenhal et al. 2020] (Fig. 5a), but the lack of angular coverage does not allow arbitrary anisotropies. Tancik et al. [2020] propose to introduce clusters and holes. We find this uneven coverage problematic for highly selective dampening and opt for a strategy that involves stratification [Niederreiter 1992]. Specifically, we use a Sobol [1967] sequence and map it to the hyperball using the method of Griepentrog et al. [2008] (Fig. 5c). We then radially warp the samples such that radially averaged sample density follows a zero-mean Gaussian distribution with variance \( \sigma^2_R \) (Fig. 5d). We find this shifting of sample budget towards the low frequencies a good trade-off between high-quality filtering with small-scale and large-scale kernels.

Our positional encoding needs to support anisotropically modulated Fourier features. To this end, we use a positive semi-definite pseudo-covariance matrix \( \hat{\Sigma} \in \mathbb{R}^{d_t \times d_t} \) to obtain frequency-dependent dampening factors \( \lambda_i \) of the individual components in Eq. 2 (Fig. 5e):

\[
\lambda_i(\hat{\Sigma}) = \exp \left( -\sqrt{\mathbf{a}^T \hat{\Sigma} \mathbf{a}} \right).
\]

Notice how \( \hat{\Sigma} \) is used without inversion here, in contrast to \( \Sigma \) in Eq. 1. This is a direct consequence of the reciprocal relationship of covariance in the primal and the Fourier domain [Brigham 1988]. Consider filtering a 2D signal with stronger horizontal than vertical smoothing. The convolution in the primal domain requires a kernel with higher variance in the horizontal direction, while the corresponding multiplication in the Fourier domain needs to dampen horizontal frequencies more strongly, leading to vertically elongated covariance.

Different from very similar existing techniques for supervised anti-aliasing based on axis-aligned frequencies [Barron et al. 2021, 2022], our dampening operates on Fourier frequencies that occupy the entire \( d_t \)-dimensional space, enabling arbitrary anisotropic filtering. In Sec. 4.4, we describe how to obtain filtering results with Gaussian covariance \( \Sigma \) from pseudo-covariance \( \hat{\Sigma} \).

#### 4.2.2 Robust Lipschitz Bounding
We require the MLP \( \overline{\mathbf{W}_\theta} \) in Eq. 6 to be Lipschitz-bounded. Eq. 3 gives us the freedom to use any \( p \)-norm, but we choose \( p = 2 \), because only this choice retains spatially invariant bounding after adding the positional encoding. To understand this connection, consider a setting with a one-dimensional input coordinate \( \mathbf{x} \). Now consider a coordinate pair \( (x_1, x_2) \) and a shifted version of it \( (x_3, x_4) \) (Fig. 6, top). If we applied \( \overline{\mathbf{W}_\theta} \) directly to these coordinates, the right-hand side of Eq. 3 tells us that because \( x_1 \) and \( x_2 \) have the same distance as \( x_3 \) and \( x_4 \), a fixed Lipschitz bound \( c \) results in the same smoothing, i.e., the bounding is spatially invariant. However, our situation is different: \( \overline{\mathbf{W}_\theta} \) does not operate on raw coordinates, but on their positional encoding \( \gamma(\mathbf{x}) \), where each \( \mathbf{x} \) is mapped to a location on a circle (Fig. 6, top). Only if equidistant coordinates remain equidistant after positional encoding, formally

\[
\|\gamma(x_1) - \gamma(x_2)\|_p = \|\gamma(x_3) - \gamma(x_4)\|_p,
\]

the property that a specific \( c \) has the same effect across the entire domain is maintained. Eq. 8 is only fulfilled by the isotropic 2-norm.

Following Eq. 4, choosing \( p = 2 \) translates into an MLP \( \overline{\mathbf{W}_\theta} \) whose weight matrices have bounded spectral norms \( \|\mathbf{W}_k\|_2 \). We pick a Lipschitz bound of \( c = 1 \), which can be satisfied by individually constraining the spectral norm of each weight matrix to at most 1.

We parameterize each (arbitrarily-shaped) weight matrix using the singular value decomposition (SVD) \( \mathbf{W}_k = \mathbf{U}_k \mathbf{S}_k \mathbf{V}_k^T \), where \( \mathbf{U}_k \) and \( \mathbf{V}_k \) are orthogonal matrices, and \( \mathbf{S}_k \) is a diagonal matrix containing the non-negative singular values of \( \mathbf{W}_k \). Using this decomposition, \( \|\mathbf{W}_k\|_2 \leq 1 \) can be achieved by constraining the trainable parameters on the diagonal of \( \mathbf{S}_k \) using a sigmoid function. To parameterize \( \mathbf{U}_k \) and \( \mathbf{V}_k \), we capitalize on the fact that the matrix
We train our neural Gaussian scale-space fields using the loss $\mathcal{L} = \mathbb{E}_{x, \hat{\Sigma}} \left\| F(x, \hat{\Sigma}) - f(x) \right\|^2_2$. (10)

It merely requires stochastically sampling coordinates $x$ and pseudo-covariances $\hat{\Sigma}$ to produce a filtered network output and comparing it against original signal samples $f(x)$. We emphasize that our training does not require $f_\Sigma$, thereby completely avoiding costly manual filtering per Eq. 1 or any approximation thereof.

As all $\hat{\Sigma}$ need to be positive semi-definite, we sample them using the eigendecomposition $\hat{\Sigma} = Q\Lambda \Lambda^{-1}$, where $Q, \Lambda \in \mathbb{R}^{d_x \times d_x}$. Specifically, we uniformly sample an orthonormal set of eigenvectors $Q$, and log-uniformly sample corresponding non-negative eigenvalues, arranged into the diagonal matrix $\Lambda$.

The network parameters $\theta$ are optimized using Adam [Kingma and Ba 2015] with default parameters.

4.4 Variance Calibration

Once trained, our neural field $F(x, \hat{\Sigma})$ in Eq. 6 captures a scale space, where the degree of smoothing is steered by modulation of Fourier features via pseudo-covariance $\hat{\Sigma}$. However, there is no guarantee at all that a particular choice of $\hat{\Sigma}$ results in a Gaussian-filtered function with covariance $\Sigma = \hat{\Sigma}$.

Importantly, we find that the relationship between $\Sigma$ and $\hat{\Sigma}$ depends on the signal $f$ itself. Consider two sine waves of the same frequency but with different amplitudes (Fig. 8). Gaussian filtering of these signals per Eq. 1 gives two identical smoothed signals up to the original difference in amplitude. Our approach does not exhibit this kind of invariance. The original low-amplitude signal has a lower Lipschitz bound (slope of the black lines in Fig. 8) and, thus, requires more aggressive bounding to achieve the same degree of smoothing as the high-amplitude signal. Therefore, the same $\hat{\Sigma}$ in Eq. 7 will have different effects depending on the signal $f$ itself.

Since our ultimate goal is to produce filtering results with control over covariance that is as precise as possible, we seek to find a signal-specific calibration function $h(\Sigma) = \hat{\Sigma}$ that allows us to obtain our final Gaussian scale-space field:

$$F(x, \Sigma) = \mathbb{E}_\theta (y(x, h(\Sigma))).$$

Notice that $h$ is injected into the pipeline after training.

![Fig. 5. Distribution of Fourier frequencies](image_url)  
![Fig. 7. Our parameterization for a Lipschitz-bounded $W_k \in \mathbb{R}^{3\times 3}$, containing nine trainable parameters $\{b_1, \ldots, b_6\}$ that can be freely optimized.](image_url)  
![Fig. 8. Lipschitz bounds](image_url)
Calibration. We design a lightweight calibration scheme for determining \( h \) that is applicable to any signal modality. First, we empirically observe that the discrepancy between \( \Sigma \) and \( \hat{\Sigma} \) stems from a difference in isotropic scale, while anisotropies are captured faithfully. Consequently, our calibration only considers matrices of the form \( \Sigma = \sigma^2 I \) and \( \hat{\Sigma} = \hat{\sigma}^2 I \), where \( I \) is the \( d \times d \) identity matrix, and \( \sigma^2, \hat{\sigma}^2 \in \mathbb{R}_{\geq 0} \) are variance and pseudo-variance, respectively.

We rely on computing a small number of Monte Carlo estimates of Gaussian smoothing that serve as ground truth and can be matched against our trained field. Specifically, we consider a set of \( n_x = 64 \) random pilot coordinates \( x_i \), and a set of \( n_{\sigma^2} = 16 \) log-uniformly spaced variances \( \sigma^2_j \). For each combination of \( x_i \) and \( \sigma^2_j \), we compute a Monte Carlo estimate of Gaussian smoothing based on \( N = 2000 \) samples from \( F(x, 0) \), i.e., our trained field without any feature dampening (Fig. 9a):

\[
F_{i,j} = \frac{1}{N} \sum_{\tau \sim N(0, \sigma^2)} F(x_i - \tau, 0). \tag{12}
\]

In addition, we consider a set of \( n_{\hat{\sigma}^2} = 256 \) log-uniformly spaced pseudo-variances \( \hat{\sigma}^2_k \) and compute, for each \( x_i \) (Fig. 9b),

\[
\hat{F}_{i,k} = F(x_i, \hat{\sigma}^2_k I). \tag{13}
\]

For each variance \( \sigma^2_j \), we now find the pseudo-variance \( \hat{\sigma}^2_{k_j} \) that results in the lowest error across pilot coordinates \( x_i \):

\[
k_j = \arg\min_k \sum_{i=1}^{n_x} \| F_{i,j} - \hat{F}_{i,k} \|^2_2. \tag{14}
\]

Our final task is to regress the transformation \( h \) that maps variances \( \sigma^2_j \) to their corresponding pseudo-variances \( \hat{\sigma}^2_{k_j} \). We observe a strong linear relationship (Fig. 9c), so we choose \( \hat{\sigma}^2 = h(\sigma^2) = \mu \sigma^2 \), where \( \mu \in \mathbb{R} \). Taking the logarithmic spacing of our samples into account, we regress

\[
\mu = \left( \prod_{j=1}^{n_{\sigma^2}} \hat{\sigma}^2_{k_j} \right)^{\frac{1}{n_{\sigma^2}}}. \tag{15}
\]

Application to the full-covariance setting gives our final calibration:

\[
\hat{\Sigma} = h(\Sigma) = \mu \Sigma. \tag{16}
\]

Fig. 9. Our variance calibration is based on pilot coordinates \( x_i \) (red points). (a) We use Monte Carlo samples (blue points) to estimate ground-truth smoothing based on our field when no feature dampening is applied. (b) We compute a sequence of differently smoothed network responses to be matched against the ground-truth from a. (c) The obtained variance–pseudo-variance pairs (green points) exhibit a linear relationship.

Discussion. The one-time estimations in Eq. 12 and Eq. 13 require a total of \( n_x \times (n_{\sigma^2} \times n_{\hat{\sigma}^2}) \approx 2M \) forward passes through our trained network, the total computation of which is instantaneous. Therefore, the entire calibration procedure imposes negligible cost compared to network training.

5 EVALUATION

We demonstrate the performance of our approach on different modalities (Sec. 5.1) and applications (Sec. 5.2), before analyzing individual components of our pipeline (Sec. 5.3). Our source code and supplementary materials are available on our project page at https://neural-gaussian-scale-space-fields.mpi-inf.mpg.de.

Implementation Details. All our signals are scaled to cover the unit domain \([-1, 1]^d\). Our positional encoding \( y \) uses 1024 Fourier features \((m = 512)\). Networks \( \Psi \) consist of four layers with 1024 features each, and are trained with a learning rate of 5e-4 (1e-4 for light stage data) until convergence. The variances for radial Fourier feature warping are \( \sigma^2 = 2000 \) for images, \( \sigma^2 = 100 \) for SDFs, \( \sigma^2 = 500 \) for light stage data, and \( \sigma^2 = 50 \) for optimization. Eigenvalues for \( \hat{\Sigma} \) during training are log-uniformly sampled in \([10^{-12}, 10^2]\). We have implemented our method in PyTorch [Paszke et al. 2017].

Baselines. We quantitatively and qualitatively compare our filtering results against several baselines, while a converged Monte Carlo estimate of Eq. 1 serves as ground truth.

BACON [Lindell et al. 2022], MINER [Saragadam et al. 2022], and PNF [Yang et al. 2022] learn neural multiscale representations, where intermediate network outputs constitute a discrete set of low-pass filtered versions of the original signal. Following Nsampi et al. [2023], we linearly combine these intermediate outputs using coefficients that we optimize per signal to best match the filtered ground truth. Only PNF supports anisotropic filtering using a discretization of orientation. MINER requires prefiltered input during training.

We further consider INS [Xu et al. 2022], which performs signal processing of a trained neural field via a dedicated filtering network. Each filter kernel requires training a separate filtering network, while our approach supports a continuous family of filter kernels.

Finally, we compare against NFC [Nsampi et al. 2023], which allows filtering based on a learned integral field that needs to be queried hundreds of times per output coordinate. While this method supports continuous axis-aligned scaling of filter kernels, general anisotropic kernels require individual optimizations, leading to a discretization of kernels in the anisotropic setting. In contrast, our approach handles arbitrary anisotropic Gaussian kernels and produces filtered results using a single forward pass. We obtain best results for NFC when using piecewise linear models for 2D isotropic filtering, and piecewise constant models in all other cases.

All methods differ in their (implicit) treatment of signal boundaries. To facilitate a meaningful quantitative comparison, we crop all results such that the boundary does not influence the evaluation. Qualitative results always show uncropped signals.

Regardless of whether we evaluate isotropic or anisotropic filtering capabilities, our scale-space fields are always trained using the complete anisotropic pipeline as described in Sec. 4.
Table 1. **Image** quality of filtering with different **isotropic** kernels (columns) for different methods (rows). "x-cont." and "\( \sigma^2 \)-cont." indicate whether the method is continuous in the spatial and the kernel domain, respectively. Bold and underlined numbers denote the best and second-best method, respectively.

<table>
<thead>
<tr>
<th>Method</th>
<th>x-cont.</th>
<th>( \sigma^2 )-cont.</th>
<th>( \sigma^2 = 0 )</th>
<th>( \sigma^2 = 10^{-1} )</th>
<th>( \sigma^2 = 10^{-3} )</th>
<th>( \sigma^2 = 10^{-2} )</th>
<th>( \sigma^2 = 10^{-1} )</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>PSNR↑</td>
<td>LPIPS↓</td>
<td>SSIM↑</td>
<td>PSNR↑</td>
<td>LPIPS↓</td>
<td>SSIM↑</td>
</tr>
<tr>
<td>BACON</td>
<td>✓</td>
<td>✓</td>
<td>32.89</td>
<td>0.308</td>
<td>0.823</td>
<td>38.95</td>
<td>0.235</td>
</tr>
<tr>
<td>MINER</td>
<td>✓</td>
<td>x</td>
<td>41.19</td>
<td>0.088</td>
<td>0.963</td>
<td>37.38</td>
<td>0.259</td>
</tr>
<tr>
<td>INSP</td>
<td>✓</td>
<td>x</td>
<td>30.57</td>
<td>0.454</td>
<td>0.770</td>
<td>30.14</td>
<td>0.420</td>
</tr>
<tr>
<td>NFC</td>
<td>✓ ✓</td>
<td>✓</td>
<td>20.75</td>
<td>0.703</td>
<td>0.533</td>
<td>26.49</td>
<td>0.224</td>
</tr>
<tr>
<td>Ours</td>
<td>✓ ✓</td>
<td>✓</td>
<td>33.85</td>
<td>0.305</td>
<td>0.854</td>
<td>35.05</td>
<td>0.207</td>
</tr>
</tbody>
</table>

Fig. 10. Qualitative results for isotropic image filtering. We show results (upper left triangles) next to error visualizations (lower right triangles). Our supplementary materials contain more visual comparisons.
We extensively evaluate our method on two signal modalities relevant for visual computing: images and signed distance fields.

5.1 Modalities

We extensively evaluate our method on two signal modalities relevant for visual computing: images and signed distance fields.

5.1.1 Images. We consider a corpus of 100 RGB images \((d_i = 2, d_o = 3)\) at a resolution of \(2048 \times 2048\) pixels, randomly selected from the Adobe FiveK dataset [Bychkovsky et al. 2011] and treated as continuous signals using bilinear interpolation. In a first step, we investigate isotropic filtering on a set of five Gaussian kernels with variances \(\sigma^2 \in \{0, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}\}\), where the first configuration measures fitting quality of the original signal without any filtering. In Tab. 1, we evaluate results using the image quality metrics PSNR, LPIPS [Zhang et al. 2018a], and SSIM [Wang et al. 2004]. Fig. 10 provides a corresponding qualitative comparison. We see that our approach is highly competitive across all filter sizes. While NFC provides the best results across all filter kernels of significant size, it introduces severe boundary artifacts, the effect of which we purposefully exclude in our numerical evaluations.

To evaluate performance for anisotropic filtering, we sample 100 full covariance matrices \(\Sigma\) using the scheme described in Sec. 4.3, where each \(\Sigma\) is evaluated on all 100 test images. We report corresponding results in Tab. 2 and Fig. 11 and observe that our approach outperforms all baselines on this task.

In Fig. 12, we show spatially varying filtering for foveated rendering. Here, the size of the filter kernel is modulated by the distance to a fixation point in the image. Our approach naturally supports such spatially varying kernels, since evaluation location \(x\) and filter covariance matrix \(\Sigma\) are independent inputs to our model.

5.1.2 Signed Distance Fields. Encoding surfaces as the zero-level-set of an SDF \((d_i = 3, d_o = 1)\) is a popular way to represent geometry [Park et al. 2019]. Our evaluation is based on four 3D models, following a similar protocol as for images. In Tab. 3 and Tab. 4, we list quantitative evaluations for isotropic and anisotropic filtering, respectively, using MSE and intersection over union (IoU) across the SDF, as well as the Chamfer distance of the reconstructed surfaces. Fig. 13 and Fig. 14 show corresponding qualitative results. While results appear mostly inconclusive for the isotropic case, we outperform the only other baseline that can handle anisotropic filtering in this domain – NFC – by a large margin.

5.2 Applications

Here, we present three applications that utilize the properties of neural Gaussian scale-space fields. First, we demonstrate anti-aliasing with texture fields, before filtering a 4D light-stage capture and showing a proof-of-concept application in the domain of continuous multiscale optimization.

5.2.1 Texture Anti-aliasing. Texturing a 3D mesh is a fundamental building block in many rendering and reconstruction pipelines. It requires re-sampling of a texture into screen space, which must account for spatially-varying, anisotropic minification and magnification to avoid aliasing [Heckbert 1986]. Our method enables this functionality for textures that are represented as continuous neural fields.

In Fig. 15, we show a result using a scale-space field for texturing an object. We first learn the scale space of the texture in uv-coordinates, and determine the optimal anisotropic Gaussian kernel for a given camera view that results in alias-free re-sampling [Heckbert 1989]. We see that our approach is successful in removing aliasing artifacts from the rendering. The supplementary materials contain a video that demonstrates view-coherent texturing.

5.2.2 Light Stage. A light stage allows to capture an object using different controlled illumination conditions. A structured capture, e.g., using one light at a time, thus enables high-quality relighting using arbitrary environment maps in a post-process.

We apply our method to the 4D product space of 2D pixel coordinates and 2D spherical light directions. As demonstrated in Fig. 16, sampling a single light direction from our model at the finest scale produces hard shadows, while moving to coarser scales in light direction introduces soft shadows.

5.2.3 Multiscale Optimization. As a final application, we show that our scale-space fields can be used for continuous multiscale optimization. In our proof-of-concept setup, we assume that we have...
Table 3. Quality of filtered SDFs with different isotropic kernels. Refer to the caption Tab. 1 for details on individual columns and highlighting.

<table>
<thead>
<tr>
<th>Method</th>
<th>x-cont.</th>
<th>(\sigma^2)-cont.</th>
<th>(\sigma^2 = 0)</th>
<th>(\sigma^2 = 10^{-4})</th>
<th>(\sigma^2 = 10^{-3})</th>
<th>(\sigma^2 = 10^{-2})</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>MSE↓ Cham↓ IoU↑</td>
<td>MSE↓ Cham↓ IoU↑</td>
<td>MSE↓ Cham↓ IoU↑</td>
<td>MSE↓ Cham↓ IoU↑</td>
<td>MSE↓ Cham↓ IoU↑</td>
</tr>
<tr>
<td>BACON</td>
<td>✓</td>
<td>✓</td>
<td>2.5e-3 1.3e-3 0.99</td>
<td>4.0e-3 2.2e-3 0.97</td>
<td>8.3e-2 1.5e-2 0.84</td>
<td>2.6e-4 4.9e-2 0.53</td>
</tr>
<tr>
<td>MINER</td>
<td>✓</td>
<td>×</td>
<td>1.6e-7 1.1e-3 0.98</td>
<td>3.3e-7 1.4e-3 0.98</td>
<td>4.1e-6 8.0e-3 0.92</td>
<td>1.8e-4 6.1e-2 0.52</td>
</tr>
<tr>
<td>INSP</td>
<td>✓</td>
<td>×</td>
<td>1.2e-1 1.3e-3 0.99</td>
<td>4.3e-2 4.4e-3 0.95</td>
<td>3.6e-2 1.1e-2 0.88</td>
<td>3.1e-2 3.7e-2 0.64</td>
</tr>
<tr>
<td>NFC</td>
<td>✓</td>
<td>✓</td>
<td>3.7e-3 5.7e-3 0.89</td>
<td>2.5e-5 4.8e-3 0.92</td>
<td>1.4e-5 2.2e-3 0.97</td>
<td>1.0e-5 2.3e-2 0.77</td>
</tr>
<tr>
<td>Ours</td>
<td>✓ ✓</td>
<td>✓</td>
<td>8.3e-5 3.9e-3 0.94</td>
<td>6.0e-5 5.5e-3 0.92</td>
<td>6.5e-4 1.6e-2 0.83</td>
<td>1.1e-2 1.3e-1 0.32</td>
</tr>
</tbody>
</table>

Fig. 13. Qualitative results for isotropic SDF filtering on a kernel with \(\sigma^2 = 10^{-3}\). Our supplementary materials contain more visual comparisons.

Table 4. SDF quality for methods that support anisotropic kernels. Refer to the caption of Tab. 1 for details on individual columns and highlighting.

<table>
<thead>
<tr>
<th>Method</th>
<th>x-cont.</th>
<th>(\Sigma)-cont.</th>
<th>MSE↓ Cham↓ IoU↑</th>
</tr>
</thead>
<tbody>
<tr>
<td>NFC</td>
<td>✓</td>
<td>×</td>
<td>7.1e-2 4.6e-1 0.08</td>
</tr>
<tr>
<td>Ours</td>
<td>✓ ✓</td>
<td>✓</td>
<td>2.8e-3 1.2e-1 0.42</td>
</tr>
</tbody>
</table>

Fig. 14. (a) An original SDF. (b) Our smoothing of (a) with an isotropic kernel, suppressing detail. (c) Our smoothing of (a) with an anisotropic kernel, where the strength of the smoothing is weaker in the vertical direction, but strong in all other directions, allowing anisotropic structure suppression. (d) Strong smoothing is only applied in the vertical direction.

Fig. 15. (a) Rendering a textured mesh is prone to aliasing artifacts. (b) Our method learns the continuous scale space of a texture and allows spatially varying, anisotropic pre-filtering, eliminating aliasing. Please consult our supplementary materials for video results.

Fig. 16. Filtering of 4D light stage-data leads to a smoothing of the illumination condition. Notice how the hard shadows in the original signal (left) are smoothed out in the filtered version (right).
Table 5. Fitting and evaluation time as well as model size on disk for different methods. "Pref. Fit Time" measures fitting a prefiltered image/SDF whose highest frequencies were removed by a Gaussian filter with $\sigma^2 = 10^{-4}$. "#Evaluations" denotes the number of model evaluations necessary to obtain a filtered output.

<table>
<thead>
<tr>
<th>Method</th>
<th>Disk Size</th>
<th>Image Fit Time</th>
<th>Pref. Fit Time</th>
<th>Eval. Time</th>
<th>#Evaluations</th>
<th>SDF Fit Time</th>
<th>Pref. Fit Time</th>
<th>Eval. Time</th>
<th>#Evaluations</th>
</tr>
</thead>
<tbody>
<tr>
<td>BACON</td>
<td>5 MB</td>
<td>89 s</td>
<td>62 s</td>
<td>1.6 s</td>
<td>1</td>
<td>2021 s</td>
<td>1538 s</td>
<td>6.2 s</td>
<td>1</td>
</tr>
<tr>
<td>PNF</td>
<td>7 MB</td>
<td>2491 s</td>
<td>1338 s</td>
<td>5.7 s</td>
<td>1</td>
<td>——</td>
<td>——</td>
<td>——</td>
<td>——</td>
</tr>
<tr>
<td>MINER</td>
<td>17 MB</td>
<td>3 s</td>
<td>2 s</td>
<td>0.1 s</td>
<td>1</td>
<td>18 s</td>
<td>15 s</td>
<td>0.1 s</td>
<td>1</td>
</tr>
<tr>
<td>INSPI</td>
<td>4 MB</td>
<td>83 s</td>
<td>11 s</td>
<td>189.5 s</td>
<td>32</td>
<td>270 s</td>
<td>255 s</td>
<td>575.7 s</td>
<td>22</td>
</tr>
<tr>
<td>NFC</td>
<td>1 MB</td>
<td>—— b</td>
<td>—— b</td>
<td>63.9 s</td>
<td>145-169</td>
<td>—— b</td>
<td>—— b</td>
<td>505.9 s</td>
<td>216-343</td>
</tr>
<tr>
<td>MLP</td>
<td>24 MB</td>
<td>15 s</td>
<td>12 s</td>
<td>1.8 s</td>
<td>1</td>
<td>107.6 s</td>
<td>75 s</td>
<td>7.3 s</td>
<td>1</td>
</tr>
<tr>
<td>Ours</td>
<td>24 MB</td>
<td>74 s</td>
<td>36 s</td>
<td>1.8 s</td>
<td>1</td>
<td>1294.1 s</td>
<td>413 s</td>
<td>7.3 s</td>
<td>1</td>
</tr>
</tbody>
</table>

*a* Includes evaluations of derivative networks obtained using automatic differentiation.

*b* The method did not reach the PSNR/Chamfer distance threshold.

Fig. 17. Our approach applied to a continuous optimization problem involving an energy landscape with multiple local minima. (a) Performing gradient descent starting from random initializations (white dots) is prone to converging to the closest local minimum (red dots). (b) Our scale-space field allows almost all initializations to converge to the global minimum.

5.3 Ablations

In this section, we analyze individual components of our method using ablation studies. We use the full anisotropic image filtering setting (Sec. 5.1.1) for this investigation and report filtering quality for different configurations in Tab. 6.

First, we are concerned with our Fourier feature dampening. We consider uncorrelated random sampling of frequencies (w/o Sobol) and removal of the frequency warping (w/o Freq. Warping). Second, we look into the Lipschitz-related components. Specifically, we consider the setup in Fig. 3a, where we learn a neural field and simply dampen the Fourier features after training (Freq. Scaling Only), before investigating configurations in which we plainly remove the Lipschitz bounding (w/o Lipschtiz), use a looser bound (10-Lipschtiz), or spectral normalization instead of our reparameterization scheme (Spectral Norm.). We also train our fields using an $\ell_1$-loss instead of using the $\ell_2$-norm in Eq. 10 ($\ell_1$-Loss).

We observe that our full method outperforms all alternative configurations.

Table 6. Ablations.

<table>
<thead>
<tr>
<th>Configuration</th>
<th>PSNR↑</th>
<th>LPIPS↓</th>
<th>SSIM↑</th>
</tr>
</thead>
<tbody>
<tr>
<td>w/o Sobol</td>
<td>34.07</td>
<td>0.072</td>
<td>0.930</td>
</tr>
<tr>
<td>w/o Freq. Warping</td>
<td>33.37</td>
<td>0.077</td>
<td>0.916</td>
</tr>
<tr>
<td>Freq. Scaling Only</td>
<td>20.66</td>
<td>0.541</td>
<td>0.563</td>
</tr>
<tr>
<td>w/o Lipschtiz</td>
<td>21.37</td>
<td>0.216</td>
<td>0.634</td>
</tr>
<tr>
<td>10-Lipschtiz</td>
<td>32.73</td>
<td>0.076</td>
<td>0.915</td>
</tr>
<tr>
<td>Spectral Norm.</td>
<td>29.08</td>
<td>0.127</td>
<td>0.868</td>
</tr>
<tr>
<td>$\ell_1$-Loss</td>
<td>29.73</td>
<td>0.081</td>
<td>0.884</td>
</tr>
<tr>
<td>Ours</td>
<td>34.82</td>
<td>0.069</td>
<td>0.940</td>
</tr>
</tbody>
</table>

5.4 Timings and Model Size

In Tab. 5, we list performance statistics across different methods. We report training times needed to achieve 30 PSNR and 0.004 Chamfer distance on unfiltered images or SDFs, respectively. We additionally measure the time and number of network evaluations required to produce a filtered output. Finally, we report model sizes when stored to disk. All experiments utilize a single NVIDIA A40 GPU.
We observe that our method is generally on par with or faster than BACON, PNF, INSP, and also NFC, which requires orders-of-magnitude more network evaluations than our approach. While MINER is faster, it is supervised on prefiltered data. A vanilla multilayer perceptron in the form of Eq. 5 (MLP) is also faster, but does not produce a scale space.

5.5 Discussion
While a Gaussian filter directly dampens amplitudes of output frequencies, our method dampens amplitudes of encoding frequencies and then relies on the Lipschitz bound to carry this dampening through to the output. We demonstrate consequences of this difference in Fig. 18. In the top row, observe that our method reduces the amplitude of the original sine wave like a Gaussian filter. However, the peaks of our wave are sharper, approaching a sawtooth wave that one would obtain from just limiting the slope of the original wave. In the spectrum, this manifests in the emergence of harmonic frequencies. Fortunately, this effect is barely visible in more complex signals as seen in the bottom row.

Neural networks exhibit an inductive bias against learning a high-frequency output when only low Fourier encoding frequencies are present [Rahaman et al. 2019]. The Lipschitz bound turns this bias into a hard constraint. Thus, it becomes even more important to include sufficiently high encoding frequencies, or else the unfiltered reconstruction is inadvertently bandlimited. In addition, the Lipschitz bound restricts the freedom of the neural network to learn arbitrary functions. We find increasing the network width to be an effective countermeasure.

Neural Radiance Fields [Mildenhall et al. 2020] are a popular application of neural fields, and combining them with our method could enable cheap anti-aliasing. Unfortunately, they exhibit a very high dynamic range in volumetric density, which poses a significant challenge for a Lipschitz-bounded network to fit. Our preliminary experiments indicate that more work is needed to accommodate this specific modality.

Many of the baseline methods we consider do not generate a continuous scale space. Instead, they output a discrete set of filtered signals, which can be linearly combined to approximate all scales that lie in between. Our method similarly combines a finite set of Fourier frequencies. In contrast to these baselines, however, our combination is performed by a highly non-linear MLP, which we observe to eliminate all traces of discretization.

6 CONCLUSION
We have introduced neural Gaussian scale-space fields, a novel paradigm that allows to learn a scale space from raw data. Crucially, we have shown that a faithful approximation of a continuous, anisotropic scale space can be obtained without computing convolutions of a signal with Gaussian kernels. Our idea relies on a careful fusion of strategically dampened Fourier features in a positional encoding and a Lipschitz-bounded neural network. The approach is lightweight, efficient, and versatile, which we have demonstrated on a range of modalities and applications.

We see plenty opportunity for future work. From a theoretical perspective, it would be interesting (and ultimately necessary) to obtain a deeper understanding of why dampened Fourier features fed into a Lipschitz-bounded network result in a good approximation of Gaussian filtering. In terms of applications, we think that our approach can potentially be a useful tool for ill-posed inverse problems in neuro-explicit frameworks, such as inverse rendering or surface reconstruction.

In light of recent efforts in continuous modeling of the real world, we hope that our neural Gaussian scale-space fields contribute a useful component to the toolbox of researchers and practitioners.

REFERENCES