Neural Fields as Distributions: Signal Processing Beyond Euclidean Space

Supplementary Material

Method	PSNR	Iterations	Inference
Nsampi et al. [21]	24.9	15726	11.2 MP/s
Ours	**25.3**	322262	22.8 MP/s

Table 4. **Video filtering** – This test compares methods on applying a 1D box filter to the time axis of a video. The PSNR is averaged over 3 300-frame videos with 300x300 resolution, and excludes the first and last 12 frames 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. The ground truth filtered frames are computed via averaging sliding windows.

7. Euclidean Filtering Experiments

Details of Euclidean filtering comparisons. For our experiments on image filtering, we use 3 512×512 images ("whale", "einstein", and "train"), and apply Gaussian and box filters to all images at the scales specified. For network architecture, we use a 4-layer, 256-width MLP with swish activations [25] and positional encoding. We train using the Adam optimizer [13], with a constant learning rate of 10^{-5}. For a minimal example of an implementation of our method for image filtering, see Fig. 4.

Further experiments. We also perform comparisons to the baseline of [21] on temporal video filtering, in a similar setup to that which they report performing. As reported in Tab. 4, we find the two methods behave similarly for this task in reconstruction accuracy, as well as in inference speed, as they use a 1-dimensional box filter, which reduces the extra network evaluations required.

8. Full non-Euclidean Derivation

For our generalized derivation, we will again start by constructing a distribution-like ground-truth field from a non-negative signal:

$$\hat{p}(\mathbf{x}) = \frac{\hat{f}(\mathbf{x})\mathcal{Q}(\mathbf{x})}{\int_{\mathbf{M}} \hat{f}(\mathbf{x}')\mathcal{Q}(\mathbf{x}')d\mu(\mathbf{x}')}. \tag{19}$$

We will apply the same assumptions as before to our filter kernels, with the added restriction that they are $\mathbf{H}$-symmetric. From this point, we can again begin our derivation from the expression for expected log-likelihood:

$$\ell(\theta|\hat{p} * k) = \mathbb{E}_{\mathbf{x}\sim\hat{p}*k}[\log(p_\theta(\mathbf{x}))]. \tag{20}$$

Again, we expand the expectation and convolution, this time using the definition from (17):

$$\ell(\theta|\hat{p} * k) = \int_{\mathbf{M}} (\hat{p} * k)(\mathbf{x}) \log(p_\theta(\mathbf{x}))d\mu(\mathbf{x}), \tag{21}$$

$$= \int_{\mathbf{M}}\int_{\mathbf{M}} \hat{p}(\eta(\mathbf{x})\eta(\mathbf{v})^{-1}\mathbf{o})k(\mathbf{v}) \log(p_\theta(\mathbf{x}))d\mu(\mathbf{v})d\mu(\mathbf{x}). \tag{22}$$

```
import jax
import jax.numpy as jnp
import flax.linen as nn
import optax
import skimage.io
import matplotlib.pyplot as plt

class MLP(nn.Module):
  @nn.compact
  def __call__(self, x):
    net = jnp.concatenate([
        jnp.sin(2**i * x) for i in range(8)]
        + [jnp.cos(2**i * x) for i in range(8)]
        + [x], axis=-1)
    for i in range(5):
      net = nn.Dense(256)(net)
      net = nn.relu(net)
    return nn.softplus(nn.Dense(3)(net))

gt_image = skimage.io.imread("512x512.jpg")
gt_image = jnp.array(gt_image / 255.0)

def loss(params, rng1, rng2):
  i = jax.random.randint(rng1, (4096, 2), 0, 512)
  x = i.astype(jnp.float32) / 511.0
  v = 0.01 * jax.random.normal(rng2, (4096, 2))
  rgb_gt = gt_image[i[:, 1], i[:, 0]]
  rgb_pred = MLP().apply(params, x + v) + 1e-6
  loss = jnp.mean(
      -jnp.log(rgb_pred) * rgb_gt) / jnp.mean(rgb_gt)
  loss += jnp.log(jnp.mean(rgb_pred))
  loss += (jnp.mean(rgb_pred) - jnp.mean(rgb_gt))**2
  return loss

rng = jax.random.PRNGKey(42)
params = MLP().init(rng, jnp.zeros((1, 2)))
optimizer = optax.adam(1e-4)
opt_state = optimizer.init(params)

@jax.jit
def step(params, opt_state, rng):
  rng1, rng2, rng_next = jax.random.split(rng, 3)
  loss_fn = lambda params: loss(params, rng1, rng2)
  grad = jax.grad(loss_fn)(params)
  updates, opt_state = optimizer.update(grad, opt_state)
  params = optax.apply_updates(updates, params)
  return params, opt_state, rng_next

for _ in range(50000):
  params, opt_state, rng = step(params, opt_state, rng)

x = jnp.linspace(0.0, 1.0, 512)
X = jnp.stack(jnp.meshgrid(x, x), axis=-1)
pred_image = MLP().apply(params, X)
plt.imshow(pred_image)
```

Figure 4. This example code provides a complete, minimal example of an implementation for our method. Specifically, it trains a 2D neural field to approximate a 512×512 image with a Gaussian blur applied. The code for our method is available at https://ubc-vision.github.io/nfd.

Figure 5. We implement environment map filtering using our method (Left), which allows lighting an object with a high-resolution environment map using a single network invocation, while Monte Carlo filtering (Right) requires thousands of samples to achieve high quality.

We then write the change of variables as $\mathbf{x}' = \eta(\mathbf{x})\eta(\mathbf{v})^{-1}\mathbf{o}$, using the group action instead of subtraction:

$$\ell(\theta|\hat{p} \circledast k) = \int_{\mathbf{M}}\int_{\mathbf{M}} \hat{p}(\mathbf{x}')k(\mathbf{v}) \log(p_\theta(\eta(\mathbf{x}')\mathbf{v}))d\mu(\mathbf{v})d\mu(\mathbf{x}'), \quad (23)$$

$$= \mathbb{E}_{\mathbf{v}\sim k,\mathbf{x}\sim\hat{p}}[\log(p_\theta(\eta(\mathbf{x})\mathbf{v}))]. \quad (24)$$

From here the derivation proceeds exactly as before, again using the normalization strategy in (9):

$$\ell(\theta|\hat{p} \circledast k) = \mathbb{E}_{\mathbf{v}\sim k,\mathbf{x}\sim\hat{p}}\left[\log\left(\frac{(\mathcal{Q} \circledast k)(\eta(\mathbf{x})\mathbf{v})f_\theta(\eta(\mathbf{x})\mathbf{v})}{\int_{\mathbf{M}}(\mathcal{Q} \circledast k)(\mathbf{x}')f_\theta(\mathbf{x}')d\mu(\mathbf{x}')}\right)\right], \quad (25)$$

$$= \mathbb{E}_{\mathbf{v}\sim k,\mathbf{x}\sim\hat{p}}[\log((\mathcal{Q} \circledast k)(\eta(\mathbf{x})\mathbf{v}))] + \mathbb{E}_{\mathbf{v}\sim k,\mathbf{x}\sim\hat{p}}[\log(f_\theta(\eta(\mathbf{x})\mathbf{v}))] - \log(\int_{\mathbf{M}}(\mathcal{Q} \circledast k)(\mathbf{x})f_\theta(\mathbf{x})d\mu(\mathbf{x})), \quad (26)$$

$$= \mathbb{E}_{\mathbf{v}\sim k,\mathbf{x}\sim\hat{p}}[\log((\mathcal{Q} \circledast k)(\eta(\mathbf{x})\mathbf{v}))] + \mathbb{E}_{\mathbf{v}\sim k,\mathbf{x}\sim\hat{p}}[\log(f_\theta(\eta(\mathbf{x})\mathbf{v}))] - \log(\mathbb{E}_{\mathbf{v}\sim k,\mathbf{x}\sim\mathcal{Q}}[f_\theta(\eta(\mathbf{x})\mathbf{v})]). \quad (27)$$

Finally, we can again reduce this to a loss function:

$$\mathcal{L}_{\text{MLE}} = \mathbb{E}_{\mathbf{v}\sim k,\mathbf{x}\sim\hat{p}}[-\log(f_\theta(\eta(\mathbf{x})\mathbf{v}))] + \log(\mathbb{E}_{\mathbf{v}\sim k,\mathbf{x}\sim\mathcal{Q}}[f_\theta(\eta(\mathbf{x})\mathbf{v})]). \quad (28)$$

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 for (13) are also applicable here.

9. Filtering Experiments on S^2

Filtering spherical data. We can apply our filtering method to a variety of neural field-based tasks. For example, some prior works have used neural fields to model *environment maps* [4, 41]. Environment maps are effectively image data, but defined over the manifold S^2 rather than the more familiar $\mathbb{R}^2$. It is also common to require these maps to be *pre-filtered* for efficient use in rendering algorithms [41]. As such, correct filtering of environment maps requires formulating the filter and convolution operation as group convolution, specifically via the action of $\text{SO}(3)$. We show the results of such a pre-filtering operation implemented via our method in Fig. 5.

10. Light Field Filtering

Lens modelling. We model lens effects as a filter applied under the action of $\text{SE}(3)$. As the models we apply these filters to are defined over rays, the filtering operation can be understood, and implemented, as a process of perturbing rays from an original ray, interpreted as representing the optical axis of the lens system, to rays which contribute to a particular region on the virtual "sensor". We use a simplified model, based approximately on a thin lens, in which the ray intersections are assumed to be distributed according to an isotropic 2-dimensional Gaussian distribution on the plane of the lens, as well as a similar Gaussian distribution on the focal plane, both centered about the optical axis. This forms a *Gaussian beam* distribution which converges towards the focal plane, and diverges away from it.

To demonstrate practical implementation, we apply this filter to two different styles of light field model: NeRF and light field networks. For the NeRF experiments, we modify the implementation of Mip-NeRF 360 [2], specifically by manipulating the integrated positional encoding to condition the network on the specific filter to be applied. As standard Mip-NeRF already models a conical distribution of rays, we need only modify the encoding of this distribution to match the ray "width" at any given depth to match our Gaussian beam distribution described above. We then use the loss function from (14) to train, while drawing random perturbations from the distribution of lens parameters we wish the model to learn.

For light field networks, the process is very similar, except that there is no integrated positional encoding. Instead, we condition the MLP directly on the parameters of the distribution as an extra input. We construct the network as an 8-layer, 1536-width, relu-activated MLP, with a learnable Fourier feature basis as positional encoding. We implement this network within the same codebase as our NeRF implementation, and all other details are the same as Mip-NeRF 360 training.

Motion blur – lifting rays to SE(3). Some distributions, which are not rotation-invariant, require lifting the field from the manifold of rays to SE(3), an example being filters representing motion blur. In practical implementations, this can be achieved in a few ways. First, in cases where the underlying model uses volume rendering and integrated positional encoding, it is possible to convolve the sampling Gaussians with a Gaussian distribution representing the linearization of the action of the SE(3) transformation at that point in space, which will be accurate as long as rotation angles are not large. We implement this approach in our modified Mip-NeRF 360 codebase, see our project page: https://ubc-vision.github.io/nfd. It is also possible to directly condition a light field network on this extra degree of freedom as a network input.

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] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *ICLR*, 2015. 1

[14] 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

[15] 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

[16] 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

[17] 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

[18] 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

[19] 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

[20] 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

[21] 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

[22] 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

[23] 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

[24] 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

[25] Prajit Ramachandran, Barret Zoph, and Quoc V Le. Searching for activation functions. *ICLR*, 2018. 1

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

[27] 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

[28] 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

[29] 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

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

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

[32] 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

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

[34] 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

[35] 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

[36] 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

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

[38] 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

[39] 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

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

[41] 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