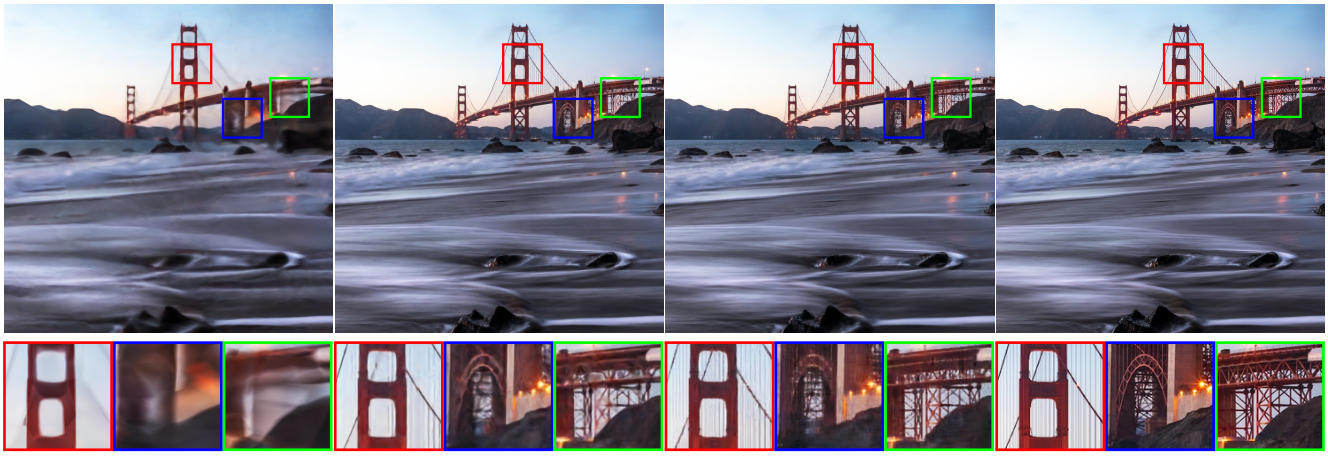
Pacific Graphics (2023)
R. Chaine, Z. Deng, and M. H. Kim (Editors)

Short Paper

Local Positional Encoding for Multi-Layer Perceptrons

S. Fujieda[1] and A. Yoshimura[1] and T. Harada[1]

[1]Advanced Micro Devices, Inc.



PSNR: 25.58 dB, SSIM: 0.7915 (a) Positional Encoding

PSNR: 30.37 dB, SSIM: 0.9133 (b) Grid Encoding

PSNR: 31.40 dB, SSIM: 0.9229 (c) Local Positional Encoding

(d) Reference

Figure 1: *Comparison of* BRIDGE *images reconstructed using an MLP with (a) positional encoding, (b) grid encoding, (c) local positional encoding, and (d) the reference. They all use a* 64×64 *grid with* 16*-dimensional latent vectors and* 4 *frequencies in positional encoding. The MLP has three hidden layers with* 64 *neurons each. The original and reconstructed image resolutions are both* $1,024 \times 1,024$.

Abstract

A multi-layer perceptron (MLP) is a type of neural networks which has a long history of research and has been studied actively recently in computer vision and graphics fields. One of the well-known problems of an MLP is the capability of expressing high-frequency signals from low-dimensional inputs. There are several studies for input encodings to improve the reconstruction quality of an MLP by applying pre-processing against the input data. This paper proposes a novel input encoding method, local positional encoding, which is an extension of positional and grid encodings. Our proposed method combines these two encoding techniques so that a small MLP learns high-frequency signals by using positional encoding with fewer frequencies under the lower resolution of the grid to consider the local position and scale in each grid cell. We demonstrate the effectiveness of our proposed method by applying it to common 2D and 3D regression tasks where it shows higher-quality results compared to positional and grid encodings, and comparable results to hierarchical variants of grid encoding such as multi-resolution grid encoding with equivalent memory footprint.

CCS Concepts

• ***Computing methodologies*** → ***Artificial intelligence; Machine learning algorithms; Image representations;***

1. Introduction

A multi-layer perceptron (MLP) has been used in many applications in computer vision and graphics to find a mapping from a low-dimensional coordinate to other properties at that location. However, MLPs usually suffer from capturing high-frequency signals from such low-dimensional inputs, which is known as *spectral bias* [RBA*19]. One approach to handle this issue of an MLP is to map the input vector to a higher-dimensional space using positional encoding [TSM*20,MST*21]. It applies sinusoidal functions to the

DOI: 10.2312/pg.20231273

input vector before passing it to the MLP. Although positional encoding is a simple and effective approach, it requires a larger network as it increases the input dimension dramatically. The highest reproducible frequency depends on the number of frequencies which is manually decided as a hyperparameter according to the use cases. It has to be many enough to represent the desired high-frequency details. To handle the number of frequencies efficiently, Hertz et al. [HPG*21] introduce a novel learning policy, SAPE, to select proper frequencies according to the local spatial position to better fit the locally varying signals. Mip-NeRF [BMT*21] and its extension, Mip-NeRF 360 [BMV*22], also try to tune the number of frequencies automatically by using features approximating the integral over the positional encoding of all coordinates within a sub-volume. If a particular frequency has a period which is larger than the size of the sub-volume, they penalize the encoding of that frequency as its amplitude gets close to zero. These methods have a better capability of representing high-frequency signals but still require high-dimensional inputs which lead to a large MLP.

The other approach to resolve the same problem of an MLP is to apply grid encoding to the inputs [STH*19, CAPM20, HCZ21, KMX*21, MGB*21, MLL*21, MESK22, TET*22, WZK*23]. The core idea of grid encoding is to prepare one or multiple grids overlaying the input domain and store latent vectors in each grid cell. We can classify this as another input encoding method, but how it uses an MLP is different from positional encoding. Grid encoding learns features on grid cells and uses an MLP as a decoder, which makes its network smaller. This is a reason for the faster training compared to positional encoding. Another nature of grid encoding is that it trades off the network complexity for the storage space. Therefore, it requires a higher-resolution grid or a higher-dimensional latent vector for each cell to resolve higher-frequency signals, which need larger memory space. Higher-dimensional inputs also require a higher-dimensional grid. Even with three-dimensional inputs such as a position in the 3D space, the memory overhead is quite significant. This nature makes it memory-intensive to use grid encoding for a problem with high-frequency information in the high-dimensional domain. There are some studies that try to overcome this by using a sparse voxel octree holding features [TLY*21], introducing a fixed-size hash table in a pyramid of grids [MESK22], or storing indices into the feature codebook in each grid cell [TET*22]. They lower the memory pressure but do not solve the fundamental problem of grid encoding which requires higher-resolution grids to achieve good quality.

Additionally, Zip-NeRF [BMV*23] tries to combine these two input encodings to take advantage of both benefits: positional encoding offers the simple representation of high-frequency signals and grid encoding allows us to use a small MLP. It integrates a pyramid of grids with hash tables [MESK22] into Mip-NeRF 360's framework [BMV*22]. However, it is still memory-intensive to adopt a set of grids and requires a large MLP. Also, Karnewar et al. [KRWM22] explore the potential of grid encoding from a different perspective. They introduce ReLU Fields where a simple ReLU function is applied to the interpolated grid values without any neural network as a decoder, which results in faster training and evaluation. However, it cannot model more than one discontinuity on signals per grid cell, so other signals such as natural images cannot be represented.

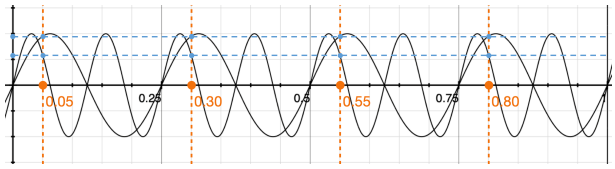


Figure 2: *Problem of using only higher frequencies in positional encoding. There are multiple points mapped to the same vector. In this one-dimensional example, $x = 0.05, 0.30, 0.55, 0.80$ are mapped to exactly the same vector from which a neural network cannot handle the difference.*

In this work, we propose local positional encoding for an MLP which is a hybrid of positional encoding and grid encoding. Local positional encoding can resolve high-frequency signals without using as many frequencies as positional encoding requires, and without preparing a high-resolution grid as grid encoding requires. Our method also stores latent coefficients in each grid cell which are combined with the signal of the local coordinate in the cell mapped by positional encoding with a few frequencies. Therefore, considering its local position and scale in each grid cell, our proposed method can resolve higher-frequency information than positional encoding with the same number of frequencies and grid encoding with the same resolution size as shown in Fig. 1. We evaluate local positional encoding in two applications, 2D image reconstruction and 3D signed distance functions, to present the advantage of the proposed method.

2. Method

2.1. Positional Encoding

Positional encoding transforms the input vector x to a higher-dimensional vector by the following equation:

$$PE(x) = [\cos(2^0\pi x), \sin(2^0\pi x), \cdots, \cos(2^{n-1}\pi x), \sin(2^{n-1}\pi x)], \quad (1)$$

where n is the number of frequencies used to encode the signal. If we use positional encoding with a limited number of frequencies (e.g. 4 frequencies), it simply fails to encode high-frequency signals as shown in Fig. 1a. Larger n could be able to capture higher-frequency details of the input signal but increases the input dimension of an MLP. An offset to the frequencies (i.e. $\sin(2^{o+i}\pi x)$ where o is the offset which we usually start from 0) can decrease the input dimension; however, this approach lets multiple locations in the domain mapped to the same vector, as illustrated in Fig. 2. It leads to training failures since the network cannot distinguish such inputs. In other words, the uniqueness of each feature vector is lost with an offset to the frequencies. Another approach to capturing high-frequency information with fewer frequencies would be to subdivide the domain of the signal into cells and assign a network with positional encoding to each cell. However, in this case, the total memory consumption increases proportionally to the number of cells. Instead, to reduce the size of the entire network, we combine the ideas of positional and grid encodings to provide hints to the network to identify the cells.

2.2. Local Positional Encoding

Our proposed method, local positional encoding, uses a uniform grid to define cells as other grid encoding methods. In order to capture high-frequency information with low memory consumption, we use a single global network for all the cells and introduce a weight for each element of positional encoding stored in each cell to utilize the local information in the cell. We call the weights latent coefficients and train them along with other network weights. Our approach is also inspired by the Short-time Fourier transform (STFT) which provides the frequency information localized in a short term when the frequency of a signal varies over time by applying a window function which spans only for a short period of time to the Fourier transform. In our method, the latent coefficients control the amplitudes of sinusoidal encodings in each cell so that the local frequency information in the cell is captured for the spatially varying signals.

More specifically, local positional encoding starts with a transformation of the global coordinate of the input point to the local coordinate:

$$x_l = x_g \cdot N - z, \tag{2}$$

$$z = \lfloor x_g \cdot N \rfloor, \tag{3}$$

where $x_l, x_g \in \mathbb{R}^d$ are local and global coordinates, N is the grid resolution and z is the cell index. The cell index z is also used to look up the latent coefficients for the cell:

$$A_{PE}(z) = [a_0^c(z), a_0^s(z), \cdots, a_{n-1}^c(z), a_{n-1}^s(z)], \tag{4}$$

where the subscripts c and s denote the coefficients for cosine and sine functions, respectively. Then we apply positional encoding to the local coordinate x_l to generate a feature vector which is multiplied by the latent coefficients $A_{PE}(z)$. We can see that this operation limits the effect of the sinusoidal functions for each cell to some range where trainable latent coefficients work as a window function for STFT. Thus, the input to the MLP is computed with the element-wise multiplication of $PE(x_l)$ and $A_{PE}(z)$:

$$\begin{aligned} PE(x_l) \odot A_{PE}(z) = [&a_0^c(z)\cos(2^0\pi x_l), a_0^s(z)\sin(2^0\pi x_l), \cdots, \\ &a_{n-1}^c(z)\cos(2^{n-1}\pi x_l), a_{n-1}^s(z)\sin(2^{n-1}\pi x_l)]. \end{aligned} \tag{5}$$

However, using this feature vector computed with Equation 5 causes visual discontinuities at the cell boundaries. This is because the cosine function with the lowest frequency (i.e. $\cos(2^0\pi x)$) has discontinuities at the edges of the range $[0,1]$. Therefore, instead of using the sinusoidal encodings with the lowest frequency, $a_0^c(z)\cos(2^0\pi x_l)$ and $a_0^s(z)\sin(2^0\pi x_l)$, we use the two-dimensional latent coefficient $A_G(z) = [a_0^g(z), a_1^g(z)]$ stored in the grid cell for the input of the MLP. $A_G(z)$ works as the feature vector of grid encoding without being multiplied by sinusoidal functions. As a result, local positional encoding transforms the input vector x_g as the following equation:

$$\begin{aligned} LPE(x_g) = [&A_G(z), a_1^c(z)\cos(2^1\pi x_l), a_1^s(z)\sin(2^1\pi x_l), \cdots, \\ &a_{n-1}^c(z)\cos(2^{n-1}\pi x_l), a_{n-1}^s(z)\sin(2^{n-1}\pi x_l)]. \end{aligned} \tag{6}$$

And the trainable latent coefficients stored in each cell are:

$$A(z) = [A_G(z), a_1^c(z), a_1^s(z), \cdots, a_{n-1}^c(z), a_{n-1}^s(z)]. \tag{7}$$

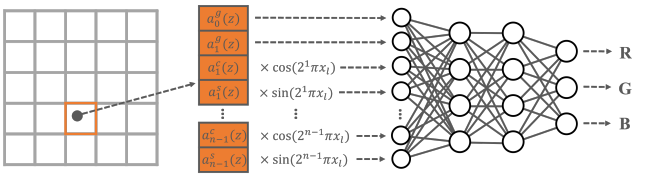


Figure 3: *Illustration of local positional encoding in a two-dimensional image reconstruction problem.*

Additionally, assigning one set of the latent coefficients in a cell causes discontinuities at the edges of the cell. Thus, we store the latent coefficients at each corner vertex of the cell and they are linearly interpolated based on the local coordinate x_l to avoid discontinuities. Fig. 3 illustrates the example of local positional encoding in a 2D image reconstruction problem.

2.3. Network

Our network is a small MLP with three hidden layers with 64 neurons each. We can use any activation in the MLP, but we experimentally choose a leaky ReLU activation function with $\alpha = 0.01$ in all the examples in this paper, except for the output layer to which we apply a sigmoid activation function for the image reconstruction and none for signed distance functions. The dimension of the input layer is given by $2 \cdot n \cdot d$ where d is the dimension of the input vector. When we use 4 frequencies in local positional encoding, for example, each cell in the grid stores an 8-dimensional feature vector for each dimension, following Equation 7. Therefore, the input to the network is also an 8-dimensional vector computed with Equation 6 in a one-dimensional problem.

The weights of the neural network are initialized with Xavier initialization procedure [GB10]. The latent coefficients in grid cells are initialized with the uniform distribution $\mathcal{U}(-10^{-4}, 10^{-4})$.

2.4. Optimization

Local positional encoding does not add any trainable parameter to the network. However, we introduce the latent coefficients, $A(z)$, in the grid which are trainable parameters to be updated during the training using the following equations:

$$\frac{\partial \mathcal{L}}{\partial a_i^c} = \frac{\partial \mathcal{L}}{\partial n_i} \cdot \frac{\partial n_i}{\partial a_i^c} = \cos(2^i \pi x_l) \cdot \frac{\partial \mathcal{L}}{\partial n_i}, \tag{8}$$

$$\frac{\partial \mathcal{L}}{\partial a_i^s} = \frac{\partial \mathcal{L}}{\partial n_i} \cdot \frac{\partial n_i}{\partial a_i^s} = \sin(2^i \pi x_l) \cdot \frac{\partial \mathcal{L}}{\partial n_i}, \tag{9}$$

where $\mathcal{L}$ is a reconstruction loss and n_i is the corresponding sinusoidal encoding of the input vector to the MLP in Equation 6. We jointly optimize the network and the grid using gradient descent with the Adam optimizer [KB15], where we set $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\varepsilon = 10^{-15}$. For the image reconstruction task, we use the $\mathcal{L}^2$ loss function with a learning rate of 0.02. And for the signed distance functions, we use the mean absolute percentage error (MAPE) similar to [MESK22] with a learning rate of 10^{-4}. These hyperparameters are chosen experimentally.

Table 1: *Comparisons of PSNR and SSIM for the images shown in Fig. 1 and Fig. 6. "PE" is short for "Positional Encoding", "GE" is for "Grid Encoding" and "LPE" is for "Local Positional Encoding". The bold numbers show the best results for each image.*

	BRIDGE			KEYBOARD			FISH MARKET			TREES		
	PE	GE	LPE	PE	GE	LPE	PE	GE	LPE	PE	GE	LPE
PSNR [dB]	25.58	30.37	**31.40**	27.98	35.26	**35.98**	18.80	23.12	**24.03**	20.56	24.20	**24.54**
SSIM	0.7915	0.9133	**0.9229**	0.8375	0.9359	**0.9367**	0.4608	0.7065	**0.7071**	0.4629	0.7090	**0.7167**

(a) PE (b) GE (c) LPE (d) Ref.

Figure 4: *Detailed comparison of* KEYBOARD *images. (a) Positional encoding, (b) grid encoding, (c) local positional encoding, and (d) the reference.*

PSNR: 24.03, SSIM: 0.7071
Different coefficients

PSNR: 23.28, SSIM: 0.6776
Same coefficients

Figure 5: *Comparison of* FISH MARKET *images reconstructed using the different coefficients (Left), and the same coefficients which halve the number of parameters (Right), for sinusoidal functions.*

3. Results

We implemented an MLP with different encodings using C++ and HIP for GPU programming language [AMD21]. All the evaluations are done by executing the codes on AMD Radeon™ RX 7900 XT or AMD Radeon™ RX 7900 XTX GPU on Windows machines. All the training weights are stored in 32-bit floating point values. We evaluate local positional encoding against other input encodings, such as positional encoding and grid encoding, in the image reconstruction task (Sec. 3.1) and in the task of representing signed distance functions (Sec. 3.2). Also, we compare our encoding with multi-resolution grids for signed distance functions (Sec. 3.3).

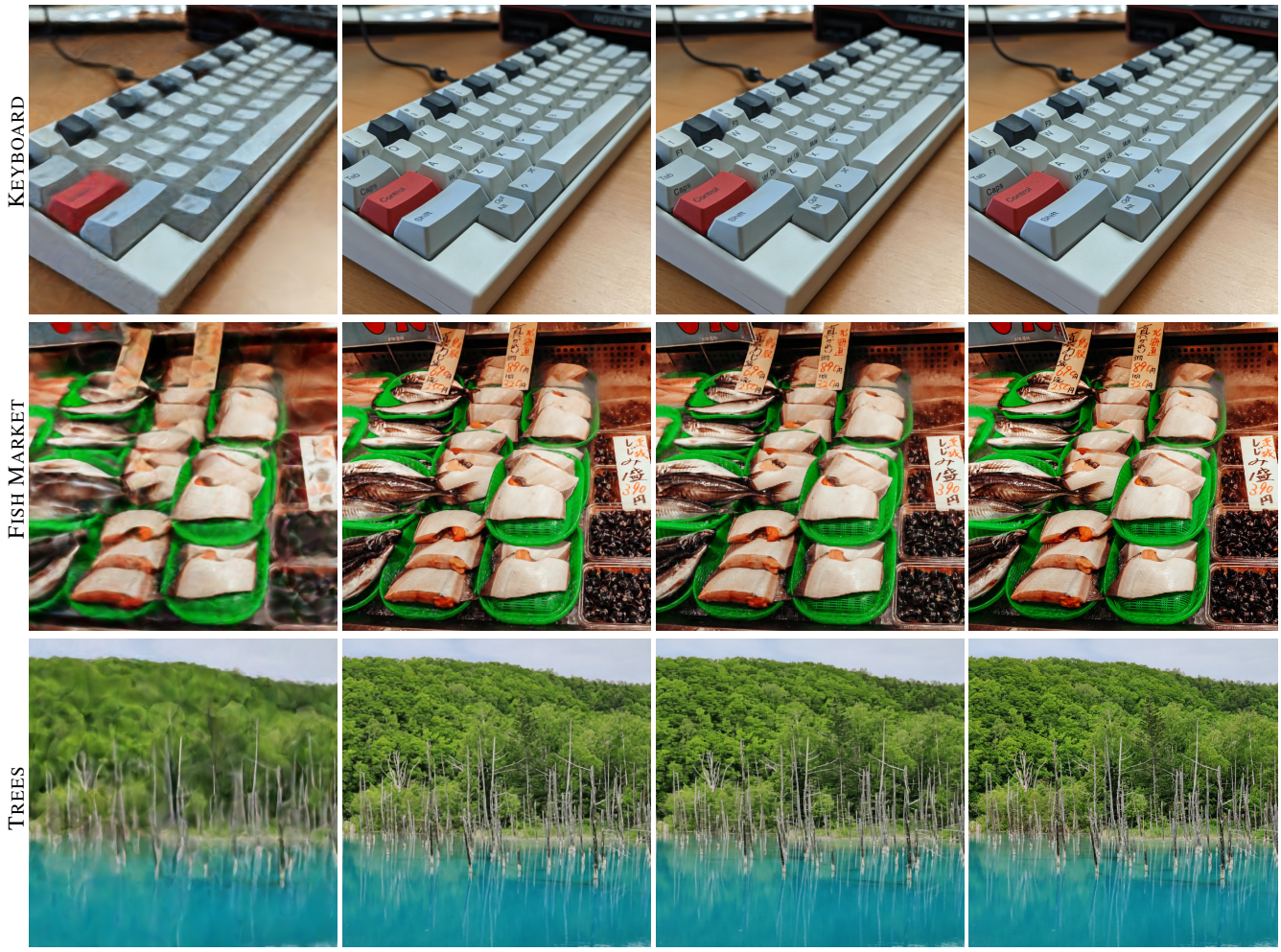
3.1. Image Reconstruction

In this task, we train the network to map a two-dimensional input coordinate to the RGB color of an image at that location. Fig. 1 and Fig. 6 compare the images reconstructed using the MLP with different input encodings, which are trained for 1k iterations. To make the comparison fair, we used the same condition for all the encodings. Local positional encoding uses a grid with $N = 64$ and positional encoding with $n = 4$. We choose these parameters experimentally. A parameter study can be found in the supplemental document. For the equivalent memory consumption, grid encoding also uses a grid with $N = 64$ which stores a 16-dimensional latent vector in each cell, and positional encoding uses 4 frequencies. All these encodings result in a 16-dimensional input vector to the MLP. All the images we used for training are $1,024 \times 1,024$ resolution and the network reconstructs images with the same resolution. As illustrated in Fig. 1 and Fig. 6, we can see that local positional encoding produces visually finer results for all images compared to other encodings while we can clearly observe that the results from grid encoding and positional encoding cannot capture high-frequency details. More detailed comparisons with close-up images can be found in Fig. 1 and Fig. 4. Additionally, Table 1 shows the quantitative comparisons with PSNR and SSIM [WBSS04] where a higher value indicates better quality. Local positional encoding achieves the highest values in both PSNR and SSIM for all images.

Note that we also tried to use the same coefficient for sinusoidal functions with the same frequency (i.e. $a_i^c = a_i^s = a_i$) in order to decrease memory consumption for the grid of local positional encoding. However, this trades off the image quality for memory consumption. As shown in Fig. 5, using the same coefficients decreases memory usage by roughly half, but it leads to worse results. In this paper, we recommend that the different coefficients are used for each sinusoidal function as discussed in Sec. 2.2, but the optimal selection of approaches depends on the specific use case.

3.2. Signed Distance Functions

Signed Distance Functions (SDFs) define the closest distance to surfaces for any spatial location to represent a shape. SDFs can be defined by discretized grid representation; however, they require a lot of memory for dense grids. We evaluate our method by letting the network learn three-dimensional SDFs and comparing the accuracy of the reconstructed SDFs with a small grid resolution. Local positional encoding uses a grid with $N = 32$ and positional encoding with $n = 3$ which results in an 18-dimensional input vector to the MLP. As with the image reconstruction task, a parameter study for SDFs also can be found in the supplemental document. For a fair comparison, grid encoding also uses a grid with $N = 32$ which stores an 18-dimensional latent vector in each cell to make the memory consumption equivalent, and positional encoding uses 3 frequencies to align the input vector dimension. As



(a) Positional Encoding (b) Grid Encoding (c) Local Positional Encoding (d) Reference

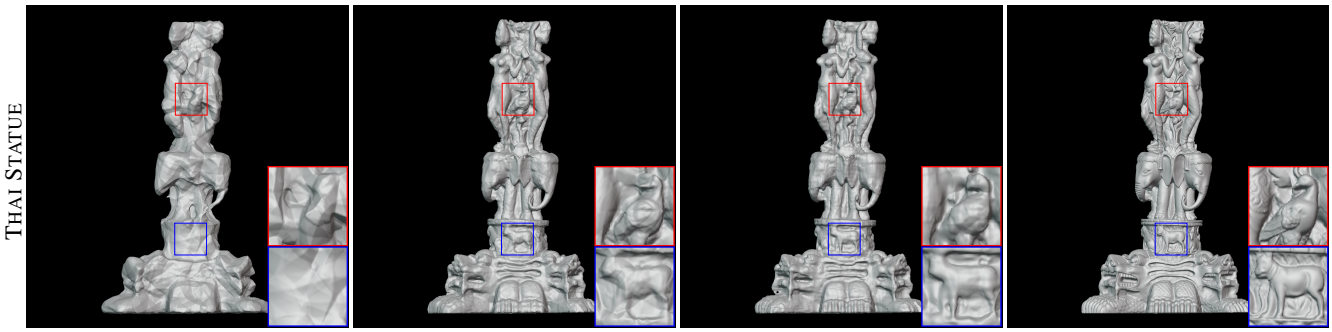
Figure 6: *Qualitative comparison among different encodings for an MLP. They all use a* 64×64 *grid with* 16*-dimensional latent vectors and* 4 *frequencies in positional encoding. The MLP has three hidden layers with* 64 *neurons each. The resolutions of all images are* $1,024 \times 1,024$.

a qualitative evaluation, we render a shape by SDFs with shading to emphasize high-frequency differences of the surface among the encodings. We use the Lit Sphere [SMGG01] for shading for consistent and reproducible results. Fig. 7 shows rendered images with the THAI STATUE model with different encodings along with the intersection-over-union (IoU) metric for a quantitative comparison, which is a ratio between the intersection and union of two volumes. We calculate IoUs by taking a sign of 128 million sampling points around the bounding box of the shape on the reference SDFs. Comparisons with more geometries can be found in the supplemental document. The close-up views of our method (Fig. 7c) show high-frequency details well while grid encoding (Fig. 7b) fails to capture them. However, it is not reflected in IoU values in Fig. 7 where grid encoding gave the highest IoUs. Additionally, local positional encoding has another benefit of faster convergence in the early training phase. Fig. 8 compares grid encoding and local positional encoding with three different training iterations. Our method converges faster and captures the fine details even in the earlier training stages, such as 64 and $1,024$ training iterations, compared to grid encoding.

3.3. Comparison with Multi-resolution Grid

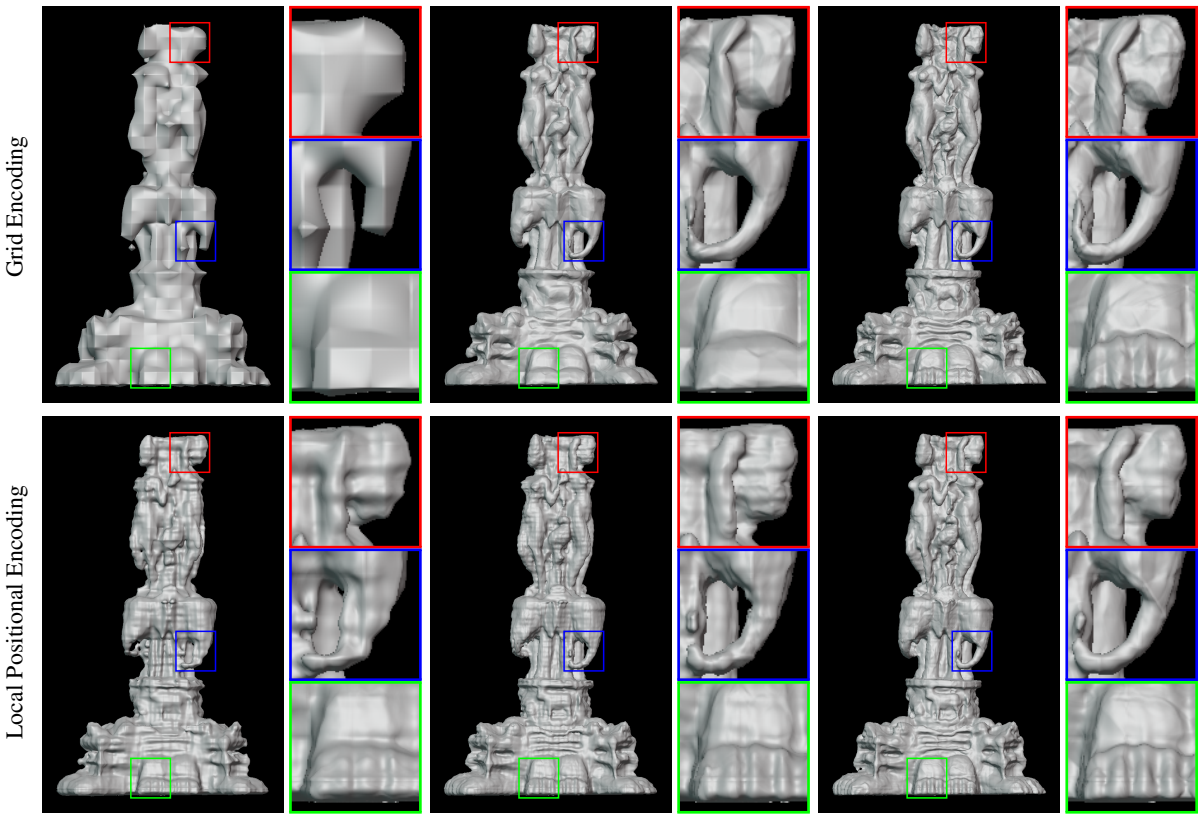
We compared our local positional encoding only with the single low-resolution grid in Sec. 3.1 and Sec. 3.2. However, multi-resolution grids containing higher-resolution grids usually achieve better quality than a single-level grid. Thus, in this section, we evaluate our encoding against multi-resolution grid encoding and multi-resolution hash encoding [MESK22], especially for SDFs.

We use almost the same number of trainable parameters in grids for each encoding for a fair comparison. We choose 3-level grids ($N = 16, 32, 64$) with 2-dimensional latent vectors as multi-resolution grid encoding, and 4-level grids ($N = 16, 32, 64, 128$) with 2-dimensional latent vectors using 2^{17} hash table size for multi-resolution hash encoding. Note that multi-resolution hash encoding has hash collisions on the higher-level grids due to the



(a) Positional Encoding (b) Grid Encoding (c) Local Positional Encoding (d) Reference

Figure 7: *Qualitative and quantitative comparison with positional encoding, grid encoding, and local positional encoding with SDFs geometry rendering. The shading is applied to show surface details by the Lit Sphere [SMGG01] based on its geometric normal. They all use a* $32 \times 32 \times 32$ *grid with* 18*-dimensional latent vectors and* 3 *frequencies in positional encoding. The MLP has three hidden layers with* 64 *neurons each. Each close-up view enclosed by red and blue squares represents how the fine details are captured by the encoding. IoU metrics are shown for each encoding. The bold number represents the best in the encodings for the geometry.*

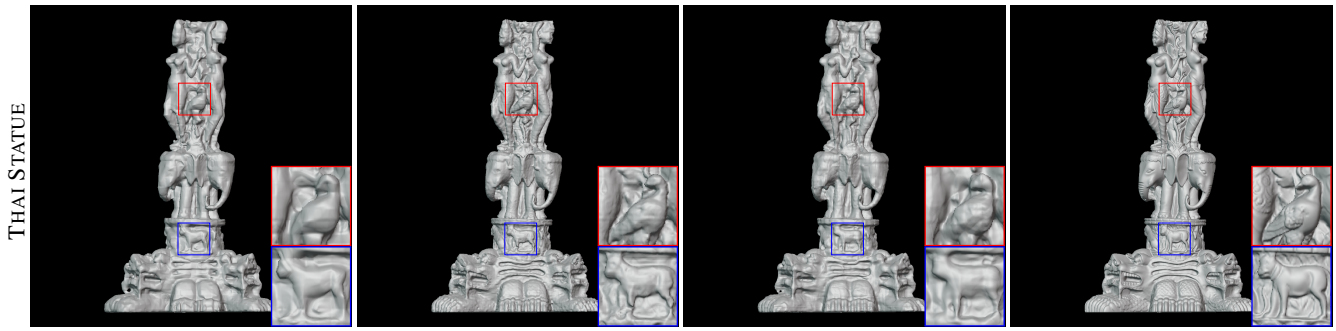


(a) 64 training iterations (b) 1,024 training iterations (c) 16,384 training iterations

Figure 8: *Convergence comparisons with different training iterations between grid encoding and local positional encoding with the* THAI STATUE *model. Our method shows finer geometric details in the earlier training stage.*

Table 2: *Comparisons of IoU with multi-resolution grids for SDFs geometry rendering. "Multi" stands for multi-resolution grid encoding, "Hash" for multi-resolution hash encoding, and "LPE" for local positional encoding. The bold numbers show the best results for each model.*

	Armadillo			Lucy			Thai Statue		
	Multi	Hash	LPE	Multi	Hash	LPE	Multi	Hash	LPE
IoU	0.9904	**0.9941**	0.9920	0.9702	**0.9856**	0.9723	0.9712	**0.9848**	0.9736



(a) Multi-resolution Grid Encoding (b) Multi-resolution Hash Encoding (c) Local Positional Encoding (d) Reference

Figure 9: *Qualitative and quantitative comparison with multi-resolution grid encoding, multi-resolution hash encoding [MESK22], and local positional encoding with SDFs geometry rendering. The parameters of all the encodings are configured for approximately the same amount of latent vectors for a fair comparison. The bold number represents the best in the encodings for the geometry.*

fixed-size hash table while multi-resolution grid encoding does not. The configuration of local positional encoding in this comparison is the same as used in Sec. 3.2. These settings result in roughly 590k parameters for all the encodings in the grids. Fig. 9 shows the rendered images with these three encodings for the Thai Statue model. The rendered images for more geometries can be found in the supplemental document. We can see that our encoding achieved visually better results than multi-resolution grid encoding and comparable results to multi-resolution hash encoding only using a single-level grid. Also, taking a closer look, multi-resolution hash encoding produces micro-structured artifacts on the smooth surface due to hash collisions, which cannot be seen with our encoding, though higher-resolution grids accepting hash collisions can capture high-frequency details well. On the other hand, in Table 2 showing the quantitative comparisons, multi-resolution hash encoding gave the highest score in the IoU metric while despite using only a single-level grid, local positional encoding achieved a higher IoU value than multi-resolution grid encoding. Note that though our encoding uses only a single-level grid, it can be extended to multi-resolution grids, with which we expect to improve our encoding further. It is conceivable to adopt multi-resolution grids in local positional encoding in future work.

4. Conclusions

In this paper, we introduced a novel input encoding method for an MLP, local positional encoding, to allow a small MLP to learn high-frequency signals with a less memory footprint. Our proposed method uses a grid to store the weights (i.e. the latent coefficients) for each sinusoidal function of positional encoding and optimizes them along with the network weights through stochastic gradient descent. Learning the latent coefficients in each grid cell locally controls the amplitudes of the encodings, so it well adopts the spatially varying signals. We demonstrated the effectiveness of local positional encoding against positional and grid encodings for the 2D image reconstruction and 3D signed distance functions. In both tasks, our method can represent better-quality results using a small MLP. Additionally, our encoding also shows comparable results even in comparisons with multi-resolution grid encodings in the SDFs reconstruction problem.

Although local positional encoding captures high-frequency information well only with a small single-level grid, it produces axis-aligned artifacts. We believe they are the inheritance of positional encoding which also suffers from such artifacts. We leave it to future work to investigate an efficient way of alleviating the artifacts. In addition, applying multi-resolution grids to local positional encoding is a straightforward extension to improve it further. Also, our method requires storing the latent coefficients for each dimensional input. Therefore, our encoding in higher dimensional tasks results in more memory intensive. Reducing the total number of latent coefficients in each grid cell is our interesting future work.

Acknowledgments

We thank colleagues in Advanced Rendering Research (ARR) Group for discussion, neural network implementation, and proofreading. The Bridge and Fish Market images are from Anand Dandekar and Sofía Rabassa, respectively. We would also like to thank the Stanford Computer Graphics Laboratory for Armadillo, Lucy, and Thai Statue models.

References

[AMD21] AMD: Hip programming guide v4.5. https://rocmdocs.amd.com/en/latest/Programming_Guides/HIP-GUIDE.html, 2021. 4

[BMT*21] BARRON J. T., MILDENHALL B., TANCIK M., HEDMAN P., MARTIN-BRUALLA R., SRINIVASAN P. P.: Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. In 2021 IEEE/CVF International Conference on Computer Vision (ICCV) (2021), pp. 5835–5844. doi:10.1109/ICCV48922.2021.00580. 2

[BMV*22] BARRON J. T., MILDENHALL B., VERBIN D., SRINIVASAN P. P., HEDMAN P.: Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2022), pp. 5460–5469. doi:10.1109/CVPR52688.2022.00539. 2

[BMV*23] BARRON J. T., MILDENHALL B., VERBIN D., SRINIVASAN P. P., HEDMAN P.: Zip-nerf: Anti-aliased grid-based neural radiance fields. arXiv (2023). 2

[CAPM20] CHIBANE J., ALLDIECK T., PONS-MOLL G.: Implicit functions in feature space for 3d shape reconstruction and completion. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (jun 2020), IEEE. 2

[GB10] GLOROT X., BENGIO Y.: Understanding the difficulty of training deep feedforward neural networks. In International Conference on Artificial Intelligence and Statistics (2010). 3

[HCZ21] HADADAN S., CHEN S., ZWICKER M.: Neural radiosity. ACM Trans. Graph. 40, 6 (dec 2021). URL: https://doi.org/10.1145/3478513.3480569, doi:10.1145/3478513.3480569. 2

[HPG*21] HERTZ A., PEREL O., GIRYES R., SORKINE-HORNUNG O., COHEN-OR D.: Sape: Spatially-adaptive progressive encoding for neural optimization. In Advances in Neural Information Processing Systems (2021), Ranzato M., Beygelzimer A., Dauphin Y., Liang P., Vaughan J. W., (Eds.), vol. 34, Curran Associates, Inc., pp. 8820–8832. URL: https://proceedings.neurips.cc/paper_files/paper/2021/file/4a06d868d044c50af0cf9bc82d2fc19f-Paper.pdf. 2

[KB15] KINGMA D. P., BA J.: Adam: A method for stochastic optimization. In 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings (2015), Bengio Y., LeCun Y., (Eds.). 3

[KMX*21] KUZNETSOV A., MULLIA K., XU Z., HAŠAN M., RAMAMOORTHI R.: Neumip: Multi-resolution neural materials. ACM Trans. Graph. 40, 4 (jul 2021). URL: https://doi.org/10.1145/3450626.3459795, doi:10.1145/3450626.3459795. 2

[KRWM22] KARNEWAR A., RITSCHEL T., WANG O., MITRA N.: Relu fields: The little non-linearity that could. In ACM SIGGRAPH 2022 Conference Proceedings (New York, NY, USA, 2022), SIGGRAPH '22, Association for Computing Machinery. URL: https://doi.org/10.1145/3528233.3530707, doi:10.1145/3528233.3530707. 2

[MESK22] MÜLLER T., EVANS A., SCHIED C., KELLER A.: Instant neural graphics primitives with a multiresolution hash encoding. ACM Trans. Graph. 41, 4 (jul 2022). URL: https://doi.org/10.1145/3528223.3530127, doi:10.1145/3528223.3530127. 2, 3, 5, 7

[MGB*21] MEHTA I., GHARBI M., BARNES C., SHECHTMAN E., RAMAMOORTHI R., CHANDRAKER M.: Modulated periodic activations for generalizable local functional representations. In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) (October 2021), pp. 14214–14223. 2

[MLL*21] MARTEL J. N., LINDELL D. B., LIN C. Z., CHAN E. R., MONTEIRO M., WETZSTEIN G.: Acorn: Adaptive coordinate networks for neural representation. ACM Trans. Graph. (SIGGRAPH) (2021). 2

[MST*21] MILDENHALL B., SRINIVASAN P. P., TANCIK M., BARRON J. T., RAMAMOORTHI R., NG R.: Nerf: Representing scenes as neural radiance fields for view synthesis. Commun. ACM 65, 1 (dec 2021), 99–106. URL: https://doi.org/10.1145/3503250, doi:10.1145/3503250. 1

[RBA*19] RAHAMAN N., BARATIN A., ARPIT D., DRAXLER F., LIN M., HAMPRECHT F., BENGIO Y., COURVILLE A.: On the spectral bias of neural networks. In Proceedings of the 36th International Conference on Machine Learning (09–15 Jun 2019), Chaudhuri K., Salakhutdinov R., (Eds.), vol. 97 of Proceedings of Machine Learning Research, PMLR, pp. 5301–5310. URL: https://proceedings.mlr.press/v97/rahaman19a.html. 1

[SMGG01] SLOAN P.-P. J., MARTIN W., GOOCH A., GOOCH B.: The lit sphere: A model for capturing npr shading from art. In Proceedings of Graphics Interface 2001 (CAN, 2001), GI '01, Canadian Information Processing Society, pp. 143–150. 5, 6

[STH*19] SITZMANN V., THIES J., HEIDE F., NIESSNER M., WETZSTEIN G., ZOLLHÖFER M.: Deepvoxels: Learning persistent 3d feature embeddings. In Proc. Computer Vision and Pattern Recognition (CVPR), IEEE (2019). 2

[TET*22] TAKIKAWA T., EVANS A., TREMBLAY J., MÜLLER T., MCGUIRE M., JACOBSON A., FIDLER S.: Variable bitrate neural fields. In ACM SIGGRAPH 2022 Conference Proceedings (New York, NY, USA, 2022), SIGGRAPH '22, Association for Computing Machinery. URL: https://doi.org/10.1145/3528233.3530727, doi:10.1145/3528233.3530727. 2

[TLY*21] TAKIKAWA T., LITALIEN J., YIN K., KREIS K., LOOP C. T., NOWROUZEZAHRAI D., JACOBSON A., MCGUIRE M., FIDLER S.: Neural geometric level of detail: Real-time rendering with implicit 3d shapes. 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2021), 11353–11362. 2

[TSM*20] TANCIK M., SRINIVASAN P. P., MILDENHALL B., FRIDOVICH-KEIL S., RAGHAVAN N., SINGHAL U., RAMAMOORTHI R., BARRON J. T., NG R.: Fourier features let networks learn high frequency functions in low dimensional domains. NeurIPS (2020). 1

[WBSS04] WANG Z., BOVIK A., SHEIKH H., SIMONCELLI E.: Image quality assessment: from error visibility to structural similarity. IEEE Transactions on Image Processing 13, 4 (2004), 600–612. doi:10.1109/TIP.2003.819861. 4

[WZK*23] WEIER P., ZIRR T., KAPLANYAN A., YAN L.-Q., SLUSALLEK P.: Neural prefiltering for correlation-aware levels of detail. In Proceedings of SIGGRAPH 2023 (2023). 2