PEPS: Positional Encoding Projected Sampling

GUILLAUME PEREZ, AMD, Spain
JANARBEK MATAI, AMD, USA
TAKAHIRO HARADA, AMD, USA

Implicit neural representations (INRs) are increasingly being used as tools to map coordinates to signals, encompassing applications from neural fields to texture compression, shape representations, and beyond. Most INR methods are based on using high-dimensional projections of the initial coordinates through encoders such as grid or positional encoding. Nevertheless, positional encoding is often insufficient and grids, as we show in this paper, require high resolution for being able to learn. In this paper, we demonstrate that positional encoding can be used not only as a high-dimensional embedding but also decomposed as a series of meaningful points. We propose the Positional Encoding Projected Sampling, where we treat the projection of the original coordinate at each frequency as a point of interest. We describe the motion of each point with respect to the frequencies and show that it follows a unique pattern. Finally, we use the unique motion of each point as a basis decomposition for doing learned positional encoding using grids. We prove, using three competitive applications – image representation, texture compression, and signed distance function– that the proposed approach outperforms the current state of the art methods, and often requires 25% less parameters for equivalent reconstruction error or rendering.

CCS Concepts: • **Computing methodologies → Image compression**.

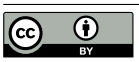
ACM Reference Format:
Guillaume Perez, Janarbek Matai, and Takahiro Harada. 2026. PEPS: Positional Encoding Projected Sampling. *Proc. ACM Comput. Graph. Interact. Tech.* 9, 1, Article 8 (May 2026), 19 pages. https://doi.org/10.1145/3806062

1 Introduction

Implicit neural representations (INRs) have emerged as a powerful tool for modeling multi-dimensional signals. INRs are models that learn coordinate-to-signal functions using neural networks. They are used in various domains, ranging from computer graphics–where they capture two-dimensional textures for texture compression [Farhadzadeh et al. 2024; Vaidyanathan et al. 2023] or three-dimensional signed distance functions for rendering and NeRF [Fujieda and Harada 2024; Mildenhall et al. 2021] to inverse problems and signal denoising [Saragadam et al. 2023]. Most INRs are based on multi-layer perceptrons (MLPs), in which the activation functions, the input embedding, and other components have been dedicated to handling the low-dimensional coordinate input signal. The reason for these dedicated components is the spectral bias of MLPs, which often struggle to represent high-frequency signals from low-dimensional inputs [Rahaman et al. 2019].

The best-known INR method to tackle this bias is to apply absolute positional encoding to the low-dimensional coordinates to project them into a higher-dimensional space [Mildenhall et al. 2021; Tancik et al. 2020]. Analogous to the positional encoding used in large language models (LLMs) [Vaswani et al. 2017], INR positional encoding applies various multiplicative coefficients to the

Authors' Contact Information: Guillaume Perez, AMD, Murcia, Spain, guillaume.perez06@gmail.com; Janarbek Matai, AMD, San Diego, California, USA, Janarbek.Matai@amd.com; Takahiro Harada, AMD, Santa Clara, CA, USA, takahiro.harada@amd.com.



ACM 2577-6193/2026/5-ART8
https://doi.org/10.1145/3806062

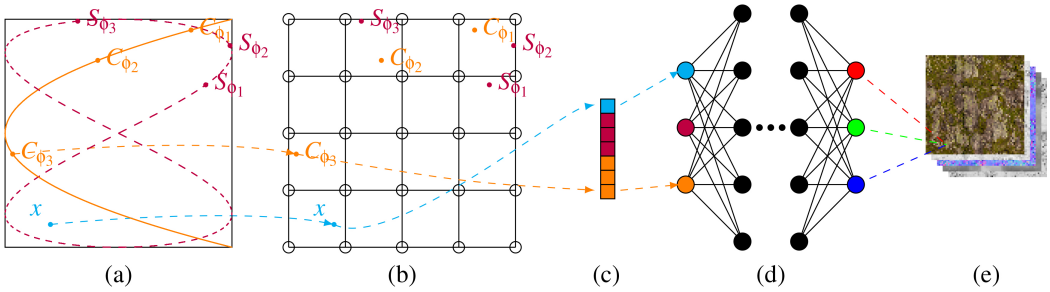


Fig. 1. Grid-PEPS method applied to neural texture compression of a texture set of size k. (a) Given a coordinate point $x \in \mathbb{R}^2$, the method extracts $S_\phi = \sin x\phi$ and $C_\phi = \cos x\phi$ for all the targeted frequencies along the frequency curves. (b) Then for each point in $P = (x, S_{\phi_1}, \dots, C_{\phi_k})$ it samples the grid to get latent values. (c) The result is the concatenation of the latent vector of each point. (d) The resulting vector is fed to the neural network. (e) The output of the neural network $T(x) \in \mathbb{R}^{3k}$ is the values of the textures of the set at coordinate x. The pseudo-code is given by Algorithm 1 without the pink part.

coordinates before applying a sinusoidal function. The greater the number of coefficients, the greater the resulting number of dimensions. This positional encoding, combined with a simple MLP, has been shown to be an efficient way to project low-dimensional coordinates into a high-dimensional space and to learn many coordinate-to-signal functions. Recently, various methods have been developed to increase the representational capabilities of INRs, for example, by replacing ReLU activations in MLPs with periodic functions [Ramasinghe and Lucey 2022; Saragadam et al. 2023; Sitzmann et al. 2020]. Most of these works come at the cost of sensitivity to initial parameters, longer training times, and, most importantly, longer inference times, preventing their use in real-time applications.

To overcome this, several applications of INRs, such as texture compression in graphics, use learned positional encoding. In learned positional encoding, a vector of parameters is learned for each coordinate during training. Grid-based models are a good example of learned positional encoding when the coordinates are bounded. They are defined by d-dimensional discrete grids of resolution $r \in \mathbb{N}$, and store a vector of parameters at each discrete location in $[0, r-1]^d$. Grid encoders use the input coordinate to sample parameter vectors from their grid. A range of different methods for representing the grids, sampling the grids, or applying various kernel functions to the resulting parameter vectors has been developed in recent years [Takikawa et al. 2021; Zhao et al. 2024]. A known limitation of simple grids, which we highlight in the experimental section, is that increasing their resolutions is almost the only way to increase their performance. That was the motivation for several works designing multi-resolution grids, hash grids, etc., to capture broader frequency bands of the input coordinates [Müller et al. 2022].

In this paper, we propose a simple method, Positional Encoding Projected Sampling (PEPS). Existing methods consider the result of absolute positional encoding as a high-dimensional vector that is either fed to the network or used as a multiplicative coefficient [Fujieda et al. 2023]. PEPS instead considers it as a *sequence of points* and uses each of those points to sample from an encoder. This formulation enables a basic grid representation to directly learn high-frequency signals, overcoming the need for high-resolution grids. Moreover, in our experiments, PEPS uses fewer parameters than all existing methods while achieving better or similar accuracy.

Furthermore, PEPS comprises several components, such as the frequencies used, the encoders used, and how multiple samples are aggregated. We propose using the power spectra of natural

images to design an efficient aggregation function for PEPS. Power Spectral Density (PSD) characterizes the power distribution of a signal over its constituent frequency components. It is a useful metric for analyzing natural processes and learning capabilities. For example, we show that the PSD of the Kodak image dataset [Kodak 2022] follows an inverse relationship with frequency, and our experimental section shows that our Pink-PEPS is the best method for representing this dataset. We leverage this inverse relationship to design Pink-PEPS, a particular version of PEPS in which the resulting points are aggregated using a latent-dimension allocation inversely proportional to their frequency.

Finally, the experimental section focuses on three distinct applications. The first application is an implicit image representation problem, a well studied problem consisting of storing an image into a neural model. Using the Kodak dataset, we show that the PEPS methods lead to new highs in term of quality, error and also distance to the power spectra of the original images. In addition, we show that PEPS methods decrease performance by only 1% compared to state-of-the-art grid methods while using 25% fewer parameters. The second application is 2D texture-set compression, where INRs are used to compress texture sets. We compare PEPS against state-of-the-art models on texture compression and show that it improves on almost all texture types. Moreover, it reduces model size by more than 25% while maintaining accuracy and rendering on par with the state of the art. Third, for the 3D signed distance function problem, the PEPS counterparts achieve the highest accuracy across all existing methods and strictly improve rendering on the hardest instances. Specific contributions of this paper can be summarized as follows:

- A simple method named Positional Encoding Projected Sampling (PEPS) that applies a learned positional encoding to the output of absolute positional encoding.
- An extension named Pink-PEPS, which is faster and well suited for representing signals with inverse power spectral density.
- Applications of PEPS to image representation and texture compression that strictly improve rendering quality over state-of-the-art models.
- Application of the PEPS method to signed distance function representation that achieves the highest accuracy in IoU and strictly improves performance on the hardest instances over state-of-the-art models.

The rest of the paper is organized as follows. We provide background and motivation for our study in the next section, then present the theoretical framework for the PEPS method. We propose two example implementations: Grid-PEPS for wrapping grid encoders, and Pink-PEPS for noise reduction. Then, for each application, we provide a comparison against state-of-the-art methods and conclude.

2 Positional Encoding and INRs

The transformer architecture introduced by [Vaswani et al. 2017] represented a paradigm shift, mainly due to the self-attention mechanism. However, this architecture was unable to capture the positional information because of its permutation invariance. That was the main motivation for the use of positional encoding (PE). In their version, the one-dimensional position index x is embedded into a high dimensional space by applying sinusoidal functions to the position. Let ϕ_i be the ith frequency targeted by the PE, the definition can be written:

$$\mathrm{PE}(x, 2i) = \sin x\phi_i \tag{1}$$

$$\mathrm{PE}(x, 2i+1) = \cos x\phi_i \tag{2}$$

This definition, known as *absolute positional encoding* (APE), was successfully incorporated in the learning process by addition or multiplication to other embeddings or simply directly fed to neural networks.

Definitions for the ϕ values have been the focus of many research, and various definitions, depending on the application, have been proposed. In the context of LLMs, the very well known definition $\phi_i = \frac{1}{10000^{i/d}}$, with d the number of dimensions of the positional encoding, has been the reference since its definition. In the context of neural fields and INRs, $\phi_i = 2^i\pi$ has established dominance and is used in texture compression, NeRF, and various higher dimensional problems [Mildenhall et al. 2021].

Learned positional encoding. are methods that associate to each possible coordinate a learned vector of parameters. It was stated in [Vaswani et al. 2017] that they used learned positional encoding analogously to [Gehring et al. 2017], and results were identical. Indeed, a drawback of learned positional encoding in the context of LLMs is their inability to represent a position unseen during training.

Yet, in the INRs field, learned positional encoding has proven its efficacy compared to other methods, and grid-based methods became the standard option [Zhao et al. 2024]. Grid-based methods uses grids, multi-grids, multi-resolution, or even multi-resolution-hash grids in their encoders. They are a type of learned positional encoding as each coordinate will results in a learned parameter vector. They are defined by three main components. The first one and most important one is the grid representation. A basic grid of resolution r and dimensions d stores k dimensional feature vectors in each cell and can be represented by a tensor of shape $(r_1, \dots, r_d, k)$. Many improvements of the grid representation have been proposed to reduce the memory consumption by using hash-maps, quantized, or block compressed representation [Müller et al. 2022; Vaidyanathan et al. 2023]. The second component is the grid feature vector extraction, that given a coordinate, extracts a set of feature vectors from the grid. For example, it is usual to extract the neighboring feature vectors of a given input coordinate. Finally, the last component is the aggregation function that uses the input coordinate and the extracted feature vectors to output the final feature vector. This last operation is often done by multi-linear interpolation of the neighboring feature vectors. Recent and state of the art results are made using a combination of those technics, combined with advanced quantization [Laurent and Anis 2025; Shi et al. 2025; Weinreich et al. 2024].

Advantage of APE. The definition of the *absolute positional encoding* offers the advantage of encoding relative 1D distances. Indeed, given a position x and a position $x+k$, the absolute positional encoding of these points is:

$$\mathrm{PE}(x+k, 2i) = \mathrm{PE}(x, 2i)\cos k\phi_i + \mathrm{PE}(x, 2i+1)\sin k\phi_i, \tag{3}$$

$$\mathrm{PE}(x+k, 2i+1) = \mathrm{PE}(x, 2i+1)\cos k\phi_i - \mathrm{PE}(x, 2i)\sin k\phi_i \tag{4}$$

This linear relationship between any two points at similar distance is one the of reason why this absolute definition allows to learn information between tokens independently of their global position. Moreover, this relationship is known to be a rotation using the following the matrix:

$$\begin{bmatrix} \mathrm{PE}(x+k, 2i+1) \\ \mathrm{PE}(x+k, 2i) \end{bmatrix} = \begin{bmatrix} \cos k\phi_i & -\sin k\phi_i \\ \sin k\phi_i & \cos k\phi_i \end{bmatrix} \begin{bmatrix} \mathrm{PE}(x, 2i+1) \\ \mathrm{PE}(x, 2i) \end{bmatrix} \tag{5}$$

This rotation is *axis wise*. This implies that applying APE to higher dimensional coordinates, such as 2D, results in each axis having this rotation relationship as it can be seen on Figure 2.

As stated above, when considering both sinusoidal functions, the transformation is a rotation coordinate wise. This implies that the positional encoding directly loss the link between the coordinates of multi-dimensional points. That is why for 2D and 3D problems such as chemical

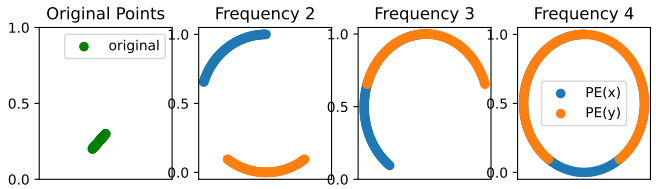


Fig. 2. Rotation of x and y axis at each frequency. On the first plot, the original line of points. Then on each frequency i plot, the coordinates are $PE(x) = (\text{PE}(x, 2i), \text{PE}(x, 2i+1))$ and normalized.

analysis, dedicated version are often designed to incorporate various geometric information [Wang et al. 2023].

Lissajous curves. Even if the rotations of each axis is independent, linear and simple, the multi-dimensional resulting spatial information is more complex. Consider first the sinusoidal 2D case. Given a point (x, y), and a frequency ϕ_i, the generated point is $(\sin \phi_i x, \sin \phi_i y)$. This is a particular case of Lissajous curve. Lissajous curves [Lissajous 1857], from the physicists Jules Antoine Lissajous, are considered a particular case of complex harmonic motions and often used in aerial search and polynomials interpolation [Erb et al. 2016; Steckenrider et al. 2024]. An in depth study of Lissajous curves is out of the scope of this paper. The main known property of those curves is the following.

Proposition 2.1. *For any two points (x, y) and (x', y'), their associated Lissajous curve is the same if and only if $(x, y) = (x', y')$.*

The proof is given in appendix.

3 Positional Encoding Projected Sampling

We propose the Positional Encoding Projected Sampling (PEPS) method. The previous section showed that the motion of points under absolute positional encoding is geometrically meaningful and results in a unique trajectory for each point. Indeed, the 3D curve resulting from positional encoding is unique for each point and lies within the range [-1, 1]. The PEPS method projects the original point multiple times onto the Lissajous curve (these are the points of interest) and samples from an encoder at each of those points. The objective is to use the uniqueness of the Lissajous curve to increase the information extracted by the learned positional encoding.

Consider INRs defined by an encoder (positional encoding, grid, etc.) and a subsequent model (MLP, etc.). PEPS can be seen as a wrapper around the encoder. It is a parametric method that projects the input coordinate into a series of points. Then, each of these new points is fed to the encoder; the results are aggregated and finally fed to the model. These four steps (Project, Encode, Aggregate, and Model) are explicitly given in the following definition.

Definition. Let $x \in \mathbb{R}^d$ be the d-dimensional input coordinate vector. Let $L \in \mathbb{N}$ be the number of frequencies used in the absolute positional encoding. Let the positional encoding coordinate points be defined by:

$$S_i(x) = \frac{1 + \sin x\phi_i}{2} \tag{6}$$

$$C_i(x) = \frac{1 + \cos x\phi_i}{2} \tag{7}$$

for $i = 1, \dots, L$, where ϕ_i denote the frequency coefficients. Let $P^x = (x, S_1, \dots, S_L, C_1, \dots, C_L)$ be the list of *points of interest* of x. S_i is used instead of $S_i(x)$ when there is no ambiguity. The size of P^x is

$2L+1$. Let encoders $E = (E_1, \dots, E_{2L+1})$ be the list of encoders, and M the model. The PEPS method is defined by:

$$M\big(A(E_1(P_1^x), \dots, E_{2L+1}(P_{2L+1}^x)), \delta\big), \tag{8}$$

where A is the aggregation function and δ denotes the other inputs to the model.

Example: Grid-PEPS. Let G be a grid encoder, where each latent vector has dimension k. Consider the case where $E_i = G$ for all i, which implies that the grid is shared by all encoders. Given a coordinate x, the Grid-PEPS method generates the points P^x, samples the grid G at each element P_i^x to obtain the latent vector l_i^x, and aggregates these results. Figure 1 shows the Grid-PEPS method, using concatenation for aggregation, applied to neural texture compression. Pseudocode is given in Algorithm 1. The additional pink component is detailed in the next section.

The aggregation function A plays an important role, analogous to techniques for embedding positional encoding into neural networks. We show that the aggregation function can drastically improve results, incorporate task-specific knowledge, and affect runtime. We studied several aggregation functions (concatenation, additive learning, frequency based-additive learning, sub-vector extraction etc.). The goal is to trade off accuracy against time and memory by controlling the number of features given to the MLP, as these methods are often coupled with very small MLPs (few layers with few neurons). Unless otherwise specified, we use basic concatenation. Given that the dimension of each l_i^x is d, the final dimension of the extracted features is $(2L+1)d$. As in absolute positional encoding, the value L controls the total number of dimensions. The next section presents the Pink aggregator, an aggregation function that extracts smaller latent vectors.

Relationship with existing works. If the encoders E_i are identity functions, then the points are directly fed to the MLP, and this method is analogous to MLPs with absolute positional encoding. Hence, the proposed method can be seen as a generalization of APE. In the context of the combination of grids and APE, Fujieda et al. [Fujieda et al. 2023] proposed the Local Positional Encoding (LPE) method. The LPE method extracts the feature vector l_0^x from the grid using the initial coordinate x, and returns the product $LocalPE(x) \odot l_0^x$, where $\odot$ is the Hadamard product, and $LocalPE(x)$ is the positional encoding of x with respect to the local cell of the grid. This differs from the proposed method, which uses the different points resulting from the positional encoding to sample the grid multiple times. In the context of hash grids, the input coordinate is transformed by the hash function to be restricted to the interval $[0, H]$, where H is the hash-table size. The idea of using a transformation of the input coordinate before extracting information is related. However, these are two complementary methods, as the hash-map approach only deals with a single coordinate. Finally, multi-resolution grids use several grids of different resolutions, often in a pyramid setting. They sample each of these grids with the input coordinate and aggregate the results. One possible comparison is that PEPS performs multi-frequency grids (encodings) in the sense that, instead of changing the resolution between grids to capture higher-frequency details, PEPS directly uses the frequency itself. We show in the experimental section that the proposed PEPS extension can be used on par with multi-resolution grids, hash grids, or any encoder, as it can be seen as a wrapper for these, and always leads to improvements.

4 Pink Aggregator for Natural Processes (Pink-PEPS)

The previous section defined the PEPS method. In this section, we show how to design an aggregation function for PEPS to control the number of dimensions and the energy allocated to each frequency.

Pink noise. Fourier analysis has long been the backbone for analyzing signals, decomposing them into their frequency components. One powerful metric based on Fourier analysis is the power

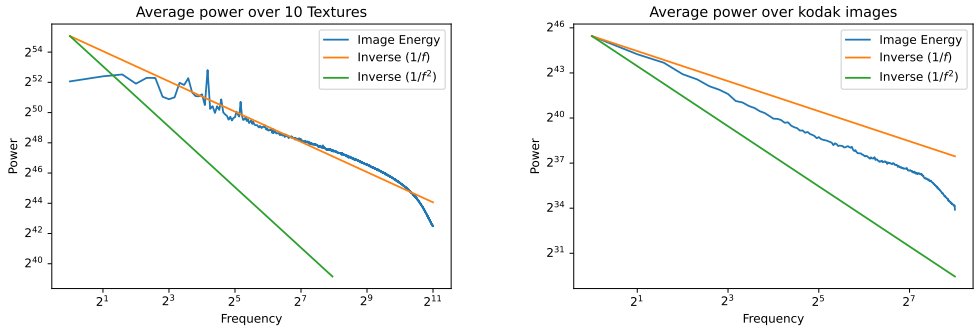


Fig. 3. (left) Power spectra of 10 textures. (right) Power spectra of the Kodak dataset. As we can see, both are following a $\frac{1}{f^\alpha}$ law [Van der Schaaf and van Hateren 1996].

spectral density (PSD). From natural images and the emissions of massive quasars scattered across the universe to the responses of biological systems and the melodies of music, many real-world signals have a PSD that follows an inverse relationship with frequency [Field 1987; Keshner 1982; Szendro et al. 2001]. These laws are usually described as $1/f^\alpha$ laws. When $\alpha = 0$, the power is the same across frequencies; white noise is an example. When $\alpha = 2$, the signal is dominated by low frequencies; Brownian motion is an example of such a process, where the next position is a small random step from the previous one. In between, when $\alpha = 1$, we have pink noise, often simply written $1/f$ noise. It is a distribution of power that is inversely proportional to frequency. Generating $1/f$ noise has been a major topic of interest in fields ranging from music [Pachet et al. 2015; Perez et al. 2018; Voss and Clarke 1975, 1978] to digital image generation [Perlin 1985]. For example, Figure 3 shows the power spectra analysis of our texture set dataset and of the Kodak dataset [Kodak 2022] indicating that they also follow a $1/f$ law. More precisely, the texture set exhibits an initial plateau and then follows a $1/f$ law, whereas the Kodak natural images follow a $1/f^\alpha$ law with $1 \leq \alpha \leq 2$. Given the potential benefits of generating $1/f$ signals, several methods have been proposed [Kasdin 1995; Keshner 1982]. A notably interesting algorithm is the Voss algorithm, which was developed by the physicist Richard Voss and then later published by Martin Gardner [Gardner 1978]. These algorithms are often based on exponentially reducing the probability or rate of modification of higher-frequency components.

For PEPS, aggregating by directly concatenating the latent vectors resulting from each frequency gives the same potential power to each frequency. Therefore, we design an aggregation function to control the power allocated to each frequency sampled by the PEPS method. Intuitively, given two frequencies f and $2f$, the inverse $(1/f)$ law implies that the power at frequency $2f$ should be approximately half of that at f. We propose the Pink aggregator for PEPS (Pink-PEPS).

Pink-PEPS. The Pink aggregator for PEPS is a resource-allocation strategy for the sampled latent vectors that takes into account the power spectra of natural processes such as images, reduces the method's dimensionality, and ensures uniform gradient flow across the grid. More precisely, it is an aggregation function that extracts a shifted subvector from each latent vector produced by the encoders used by PEPS and concatenates them.

Given a vector $v \in \mathbb{R}^d$, we denote by $v_{i:j}$ the circular sub-vector of v containing elements $(v_{k_i}, v_{k_{i+1}}, \ldots, v_{k_j})$ with $k_i = i \mod d$. Let the sequence $a_0 = 0$, $a_n = \max(1, \left\lfloor \frac{d}{f_n} \right\rfloor)$ for any strictly positive integer n, denote the allocation of latent dimensions for each frequency n. For example,

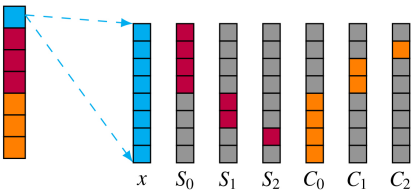


Fig. 4. Pink aggregation applied to a latent dimension (d) of 8, and using the Fourier encoding ($\phi_i = 2^i\pi$). The simple concatenation would directly concatenate the 8*7=56 values (i.e., the blue, purple, orange, and gray ones), while the Pink-PEPS only considers 8+2*4+2*2+2*1=22 values that are inversely spread given their frequency (i.e., the blue, purple, and orange ones).

using the Fourier encoding $\phi_i = 2^i\pi$, the frequency if given by $f_i = \frac{\phi_i}{2\pi} = 2^i$. This implies that $a_n = \max(1, \lfloor \frac{d}{2^n} \rfloor)$.

Let the series $G_n = \sum_{i=0}^{n} a_i$ denotes the sum of already allocated dimensions for frequencies 1 to n. The Pink aggregator extracts from each vector l^{C_i} (resp. l^{S_i}) the sub-vector $l^{C_i}_{G_{i-1}:G_i}$ (resp. $l^{S_i}_{-G_i:-G_{i-1}}$). An example is given in Figure 4, and a possible pseudo-code is given by Algorithm 1.

Intuitively, for each higher frequency, the Pink-PEPS method restricts the dimensionality of the sampled latent vector so that it is inversely proportional to the frequency. For example, using the Fourier positional encoding, the latent dimension is divided by two each time and shifted. The intuition behind the shift, using the series G_n, is to ensure that, when sharing a grid or any learned PE, the whole latent vector receives gradients, rather than only the first few dimensions. Note that the complete vectors (e.g., l^{C_i}) do not need to be computed, as only subparts are required. This is the particular point that makes our implementation of Pink-PEPS faster. Note that an α term can be added to the definition of $\frac{d}{f_n^\alpha}$ for a_n. When $\alpha = 0$, this is the usual PEPS; when $\alpha = 1$, this is Pink-PEPS; and when $\alpha = 2$, this gives Brownian-PEPS. This definition generalizes the PEPS method.

Algorithm 1 Full Grid-Pink-PEPS algorithm

Require: $x \in \mathbb{R}^d$, E : grid encoder, $L \in \mathbb{N}$
  $g \leftarrow E(x)$
  for $i = 1, \dots, L$ **do**
    $l^{S_i} = E(\sin x\phi_i)$
    $l^{C_i} = E(\cos x\phi_i)$
    g.append($l^{S_i}_{-G_i:-G_{i-1}}$)
    g.append($l^{C_i}_{G_{i-1}:G_i}$)
  end for
  return g

5 Application: Implicit Image Representation

We start the experimental section with the implicit image representation problem. In implicit image representation, given a coordinate $x \in [0,1]^2$, the function $P(x)$ returns the RGB values $(r,g,b) \in \mathbb{R}^3$ at location (x). Implicit neural representation is the set of methods using neural

networks for representing function P. This section is split in two parts. First we propose to analyze the learning capability of methods based on grids, and shows how the PEPS method is able to improve their overall capabilities. Then, we compare using the Kodak images dataset [Kodak 2022] to prove the efficacy of the proposed method in the usual settings.

Current bottlenecks for grid methods. Before going into the large comparison against the state of the art methods, we propose to first analyze the parameters impact of the grid methods and PEPS. We use bilinear Interpolation grids (BI grid) which are grids using the bilinear interpolation of the neighbors latent vectors. Local Positional Encoding (LPE) is a wrapper around a BI Grid and multiplies the resulting latent vector by the positional encoding of the point with respect to the cell [Fujieda et al. 2023]. Grid-PEPS is a PEPS wrapper around the same BI Grid which is shared across frequencies and using three frequencies (ϕ_1, ϕ_2, ϕ_3). The $\phi_i = 2^i\pi$ definition is used in all experimental sections. The models are composed of the encoder followed by an MLP of three hidden layers of 64 neurons for both experiments. The learning rate is 0.01. More details are given in the technical appendix. The first experiment is on learning 4K images (roughly 10^8 values) with models having between 10^4 to 10^6 parameters (grid resolutions between 16 to 128, feature vector dimensions between 8 to 64). For each image, a model is trained using the same number of steps, and the results are with respect to the average of all the runs of all images. All implementations have been made in pytorch [Paszke et al. 2017] and are shared between methods to avoid artifact due to different initialization, etc. The loss is the L_1 loss. The metric used in this first study is the Peak signal-to-noise ratio (PSNR). PSNR being based on the log of the mean square error, it is a good learning capability metric for neural networks. Positional encoding is kept out of these results as it does not have a parameter other than L, and is strictly dominated by several order of magnitude when using small neural networks. Using larger neural network lightly improves the results, yet not as much as grid-based methods and drastically increase the running time. In such setting of

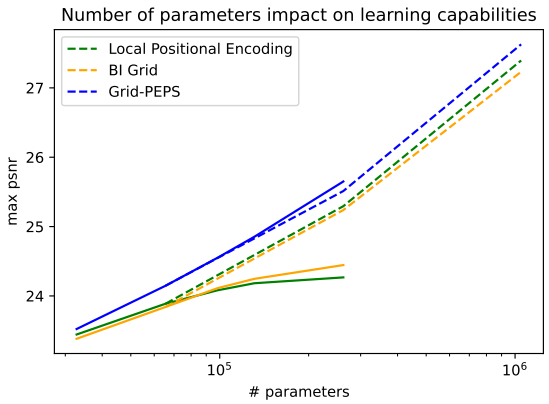


Fig. 5. Impact of the number of parameters on the image learning capability (PSNR↑). Dashed lines are grid resolution modifications. Regular lines are feature dimension modifications.

extreme compression, a hypothesis would be that additional parameters should always improve the model. Yet, Figure 5 shows that increasing the parameters doesn't always help the learning model. Even in this setting, methods such as grid and LPE fail to improve when given higher dimensional latent vectors. In contrary, the Grid-PEPS method, which is based on the BI grid, is able to linearly use the additional parameters to improve its fitting. It is able to reach new highs in PSNR with low

resolution grids. In addition, the recent paper on NTBC [Laurent and Anis 2025] explain that they had to shift by half a texel the odd textures, etc. This kind of issue is by design corrected by the PEPS methods. Finally, PEPS remains the best method, whatever the parameters configuration.

Table 1. Kodak dataset of *low* resolution (768,512) images. All methods have the same number of parameters, except LPE having 6 percent less parameters, the bottom row having 25 percent less parameters, and PE which use a larger MLP.

Method	PSNR (↑)	FLIP (↓)	LPIPS (↓)	LSD (↓)	SSIM (↑)
PE	39.91	3.71e-02	1.47e-02	4.46e-02	0.960
LPE -6%	45.06	*1.84e-02*	1.62e-03	7.49e-03	0.992
NTC_N	44.87	1.96e-02	2.23e-03	9.74e-03	0.992
Grid	45.30	**1.83e-02**	1.37e-03	1.24e-02	0.992
G-PEPS	47.72	2.08e-02	1.30e-03	*4.19e-03*	*0.993*
G-P-PEPS	47.83	2.24e-02	**1.05e-03**	4.41e-03	*0.993*
NTC_{PEPS}	*48.02*	2.07e-02	1.43e-03	**4.07e-03**	**0.994**
$NTC_{PinkPEPS}$	**48.07**	2.14e-02	*1.20e-03*	4.22e-03	**0.994**
G-P-PEPS -25%	44.89	2.81e-02	3.12e-03	8.45e-03	0.987

Kodak dataset. The final MLP having the same size for any final resolution might results in discrepancies depending on the resolution. We propose to focus on a smaller resolution set, with smaller encoders but using the same MLP size as before (three hidden of 64 neurons) to avoid its saturation and highlight the actual impact of the encoder. The dataset is the Kodak dataset having 24 images of resolution (768,512) [Kodak 2022]. The NTC methods have two grids of respective size (192,128) and (96,64) and respective feature dimensions of 12 and 20 following the architecture from the neural texture compression paper [Vaidyanathan et al. 2023]. Grid and G-PEPS and G-P-PEPS (Grid Pink PEPS) have the same grid of (196,128) and 17 features. Those settings represents a factor of around 2.7 of compression. The PE (positional encoding) used 10 Fourier frequencies and a larger MLP of 3 layers of 300 neurons. Four additional metrics are reported here. The SSIM metric is a perception quality metric comparing the structural information of the images [Wang et al. 2004]. The FLIP error is the average of the FLIP error of an image [Andersson et al. 2020]. The LPIPS error is a perception metric defined using an Alexnet [Zhang et al. 2018]. The LSD metric, also known as log-spectral distortion, is simply the distance between two log-spectra applied to the 2D Fourier representation of the image [Gray and Markel 2003; Rabiner and Juang 1993]. This measure is heavily used in signal processing, and mostly in speech applications. More discussion about those metrics can be found in appendix. We use this spectral metric here to characterize the capability of a neural network to output an image having the same spectral property as the original one. As we can see in Table 1 the PEPS method (G-PEPS and NTC_{PEPS}) are drastically improving their non-PEPS counterparts. Then, the PinkPEPS methods (G-P-PEPS, etc.) are clearly better in all metrics including reconstruction error and perceptual quality and spectral distance. Our current explanation is that in all settings, the additional features brought by PEPS methods are informative, but the small neural networks required for real-time rendering might struggle to use it. The PinkPEPS methods returning less values, especially less high frequency values, reduces the input noise and the total number of parameters. Dual comparison scatters are provided in Figure 7. As we can see, not only the PEPS method is better in average, but it completely dominates its grid counterpart. Moreover, the version having 25% less parameters seems comparable to the existing SOTA in PSNR, LSD, and SSIM. The same result conclusion can be seen for the NTC versions too. Moreover, even with a L2 loss, adaptive dual learning rates and different activation (full description in appendix), the PEPS remains dominant as shown by the fourth plot.

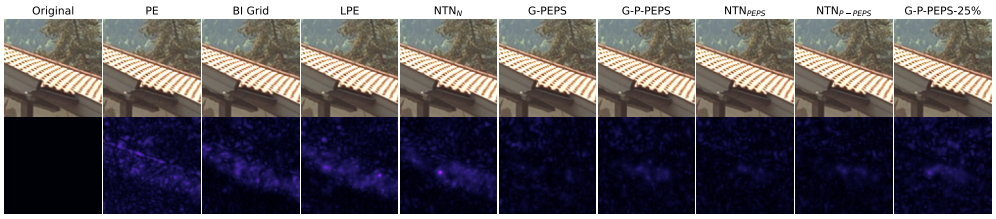


Fig. 6. (Image) 100 pixels samples from the Kodak dataset. Bottom images are Flip error [Andersson et al. 2020].

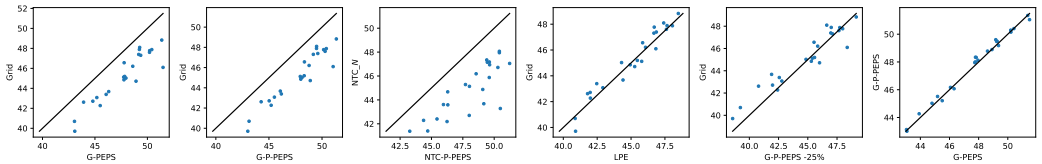


Fig. 7. Dual scatters of PSNR results for the Kodak dataset. Each axis is representing the PSNR of a method. If a point is on the side of a method, its result is better for this method.

In addition, the result that the basic grid is better than the engineered method NTC$_N$ can be directly explained by Figure 5. In this settings, trading the features of higher resolution grids to make high-dimensionality latent features in lower-resolution grids is not working, and the NTC$_N$ lacks efficiency. That is the bottleneck we highlighted. In contrary, as shown in Figure 5, the PEPS method is able to use the low-resolution grids with high-dimensional latent feature vectors, which explains the good results of NTC$_{PEPS}$. That is one of the main advantages of the PEPS method, the ability to use parameters almost equivalently from resolution of latent dimensional size. Samples from the generated images are shown in Figure 6, and in the additional files. As we can see, the PEPS methods exhibit less error and are closer to the color palette from the original image. The results of this section aimed at highlighting the impact of the PEPS method. We also tested other loss functions (L1, L2), activation functions (Leaky Relu, Gelu, Silu), and adaptive dual learning rates. None of those modification led to change in the capability of the PEPS counterpart to perform better in term of PSNR and SSIM. Some of results are shown in appendix.

6 Application: Neural Texture Compression

Textures and materials are the backbone of modern 3D engines and physically based rendering that are used in applications such as games. In those engines, many 4K textures must be loaded at the same time, creating a memory bottleneck. That is the reason why neural texture compression (NTC) has gained interest. Compared to traditional image compression technics, the goal is to have random access to the values of all the textures of the set at some given pixel locations. Indeed, the rendering mechanism usually sample the information of the texture locations hit by rays. Recently, an emphasis has been put on texture representation and compression, and existing works has shown that INRs can represents textures sets at different MIP level efficiently. [Farhadzadeh et al. 2024; Vaidyanathan et al. 2023]. Preliminary research works have focused on the capability of positional encoding and grid-like representation to represent and efficiently compress textures and materials. Recent works focused more on making the grid-like methods faster by quantization-like technics either ad-hoc or using available tools such as block compression and cooperative vectors [Laurent

and Anis 2025; Shi et al. 2025; Weinreich et al. 2024]. These methods can be described using their multi-grids definition, fine tuned frequencies for positional encoding, and quantization strategy. From a learning capability point of view, even if their implementation is different, they are all bounded by the learning capability of the grid itself. Compared to image representation, in texture representation, given a coordinate $x \in [0,1]^2$, the function $T(x)$ returns the texture set information $t \in \mathbb{R}^{3k}$ at this location with k the number of textures of the texture set. The goal for a given texture set S associated with function T_S, is to train a network N_S such that $N_S(x) = T_S(x), \quad \forall x \in [0,1]^2$. Additional goals are compression and real-time capabilities.

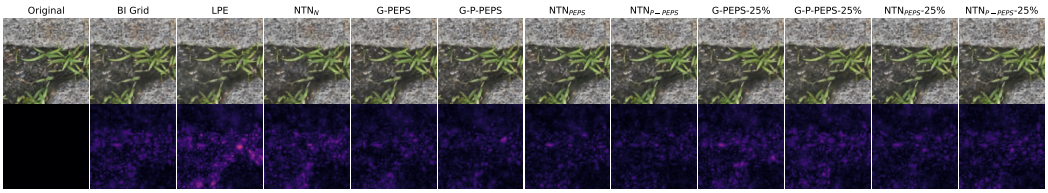


Fig. 8. 100 pixels samples from the 4K Paving Stone texture set. Bottom images are Flip error [Andersson et al. 2020].

Texture set compression. For a fair comparison, we propose to align all the methods with the configuration given in [Vaidyanathan et al. 2023] named NTC$_N$. For 4K texture sets, the BI Grid, Grid-PEPS, and Grid-Pink-PEPS methods will have a single grid of resolution 1024 and feature vectors of dimension 17. The LPE method will have a single grid of resolution 1024 and feature vectors of dimension 16. The NTC methods have two grids of respective size 1024 and 512 and respective feature dimensions of 12 and 20. In addition, the NTC methods have the *Tiled positional encoding* which corresponds here to the last three frequencies of the positional encoding with respect to image size. The median number of textures in texture sets is 5, so the average compression factor is 14 (ratio is 93%). The "-25%" versions have the same grid sizes and respective latent-feature dimensions of 9 and 15 for the NTCs and 13 for the Grids, We experimentally saw that the number of frequencies have a light impact on the PSNR, and almost none on the SSIM, yet three or four frequencies seems to be a sweet spot for the methods. In this section, all PEPS methods uses four frequencies. so the average compression factor is 18 (ratio is 95%). No quantization is involved as our goal is to analyze the impact of the PEPS methods, and not the various quantization strategies. For that reason, methods from [Laurent and Anis 2025; Shi et al. 2025; Weinreich et al. 2024] as they rely on quantization technics of grids are omitted. The rest of the configuration is the same as the previous experiment. The model is learning the filtered textures directly using bilinear interpolation of the true pixels during the training. The texture sets used in this paper are based on the available sets from [Fujieda and Harada 2024; Weinreich et al. 2024] and additional sets, all available from https://polyhaven.com/. Except when clearly stated, the total number of texture sets is 18. Full description of the dataset is given in appendix.

Table 2 contains the averaged results over our dataset. As we can see, the Grid Pink PEPS method has better quality metrics than both the BI grid and the NTC$_N$ methods containing several grids and positional encoding. This is a clear result that shows the proposed method improves the learning capabilities of grid-based methods. For texture compression, the PinkPEPS improvement of gridPEPS is smaller, yet still present. A reason could be that the PSD of the texture set starts with a plateau as shown in Figure 3. The NTC$_{PEPS}$ and NTC$_{PinkPEPS}$ methods that use the architecture of NTC$_N$, and Grid-PEPS or PinkPEPS instead of the original grids, show the best performances

Table 2. For each texture type (AO, etc.) the value is the average PSNR over the dateset. PSNR and SSIM columns are the average over all textures of all texture sets of the dataset. NTC_N is the method from [Vaidyanathan et al. 2023] without quantization. LPE is the method from [Fujieda et al. 2023]. The three first line are SOTA methods. Then, the PEPS method is the middle are wrapped around the BI grid or using the same multi-resolution grid architecture as [Vaidyanathan et al. 2023]. Finally, at the bottom, all those methods have have 25% fewer parameters that the rest of the table that have the same overall number of parameters.

Method	PSNR	SSIM	AO	ARM	DIFF	Disp.	metal	normals	rough	specular
LPE	40.21	0.95	42.16	41.18	36.38	50.34	46.83	34.99	39.05	44.76
NTC_N	40.20	0.95	42.59	41.29	36.78	48.99	46.07	34.99	39.18	46.13
BI grid	41.25	0.95	43.59	42.24	37.42	**51.18**	47.35	36.03	39.98	46.07
Grid-PEPS4F	41.23	0.95	43.42	42.11	37.09	*50.81*	49.90	35.90	39.67	47.74
Grid-PinkPEPS4F	41.44	0.95	43.55	42.48	37.48	50.64	49.43	36.24	40.10	47.73
NTC_{PEPS}	*41.79*	0.95	*44.17*	*42.73*	*37.91*	50.28	*50.47*	*36.64*	*40.23*	**48.56**
$NTC_{PinkPEPS}$	**41.89**	0.95	**44.34**	**42.79**	**38.09**	50.42	**50.10**	**36.71**	**40.47**	*48.24*
Grid-PEPS4F-25%	39.86	0.93	42.19	40.83	35.86	49.15	48.63	34.58	38.50	45.41
Grid-PinkPEPS4F-25%	40.03	0.94	42.32	41.26	36.20	48.83	48.07	34.81	38.83	45.31
NTC_{PEPS}-25%	40.59	0.94	43.17	41.60	36.70	49.65	49.23	35.27	39.10	46.15
$NTC_{PinkPEPS}$-25%	40.56	0.94	43.18	41.65	36.74	49.61	47.90	35.30	39.19	46.12

overall. In almost all categories or texture type, they are strictly better. This confirms that both from a direct use (i.e., grid pink PEPS), and used inside high-level architecture, the proposed PEPS methods exhibit better learning capabilities. Finally, the appendix contains samples of the resulting textures. They show that the PEPS counterparts almost always have better FLIP error. Figure 8 is one of them. In conclusion, in all three compression settings, extreme with bottleneck, low with Kodak dataset, and regular compressions here, using the PEPS alternatives always improve the reconstruction error.

Runtime Analysis. We evaluated the performance overhead of PEPS by implementing neural texture decompression using HIP [AMD 2016] and running the code on a Radeon ™RX 9070 XT GPU. The implementation uses Wave Matrix Multiply Accumulate intrinsics for the MLP [AMD 2025]. The encoder and the MLP were fused into a single kernel. We compared the inference time of BI grid and Grid-PEPS, and Grid-PinkPEPS with the same grid configuration: a 1024×1024 grid, a 16 dimensional feature vector and three frequencies for both Grid-PEPS, and Grid-PinkPEPS. Generating a single three-channel 1024^2 texture took 5.47ms, for Grid-PEPS, 4.32ms for BI grid, and 4.86ms for Grid-PinkPEPS. Grid-PEPS was 26.6% slower than BI grid, a penalty that stems from the additional arithmetic operations and memory access required for the the additional grid sampling. Because Grid-PinkPEPS creates smaller input feature vector, it performs fewer arithmetic operation and memory accesses than Grid-PEPS, its runtime of 4.86ms corresponds to a 12.5 % overhead relative to the BI grid. We also tested Grid-PinkPEPS with four frequencies which results in 4.99ms (15.5% overhead), which is still faster than Grid-PEPS. When the grid size is smaller which makes it more cache friendly, we expect the overhead to be smaller.

7 Application: Signed Distance Function

Signed Distance Functions (SDFs) are used to represent 3D shapes during rendering [Osher and Fedkiw 2003; Vicini et al. 2022]. SDFs give the closest distance to surfaces for any spatial location to represent a shape. In addition to distance, SDFs give the information of the location of the point. The SDF will be positive when the point of interest is in the exterior region, and negative when in the interior region. Many works have been defined for representing SDF as the dense representation

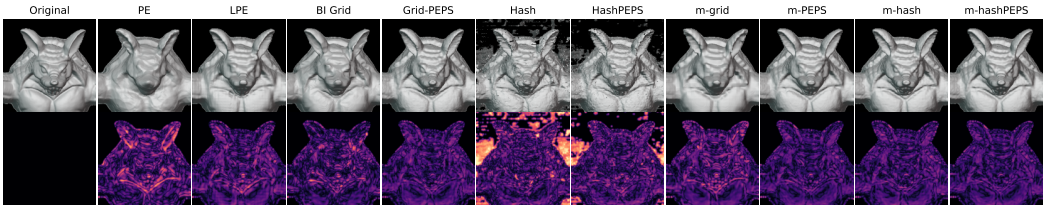


Fig. 9. (SDF) 100 pixels samples from the Armadillo model. Bottom images are Flip error [Andersson et al. 2020].

requires a high-resolution grid that usually consumes a large amount of memory [Saragadam et al. 2023].

Analogously to the previous section, we propose to analyze the impact of the introduction of the PEPS method using existing encoders of the state of the art. We will be using Trilinear-interpolated grids (*TI grid*). Hash-grid methods, multi-grids, and multi-resolution-hash grids [Müller et al. 2022] are also compared. Finally, we propose their PEPS versions, *Grid-PEPS* for grid, *Hash-PEPS* for Hash, *m-PEPS* for multi-grids PEPS, *m-hashPEPS* for multi resolution hash grids with PEPS. As for the NTC section, the PEPS method uses the same encoder across three frequencies. No additional parameters are required except for the first layer of the MLP as the input is high-dimensional. We follow the experimental protocol setting from [Fujieda et al. 2023]. The single grid methods have a resolution of 32 and stores 18 dimensional feature vectors in each grid location. The hash grid method has a resolution of 64 and stores 18 dimensional feature vectors in a hash table of size 32^3. The multi-grid method has 3 grids of resolution [16,32,64] and stores 2 dimensional feature vectors in each grid location. The multi-hash grids are of resolutions [16,32,64,128] and use 2 dimensional feature vectors and maximum hash table size of 2^{17}. PE is a positional encoding with 10 frequencies. Finally, the MLP is composed of 3 layers of 64 neurons. The loss of this section is the mean absolute percentage error (MAPE) loss. The four SDF instances are defined using 512^3 values. The compression factor is 227 (ratio is 99.6%). Rendered results and results with *L1* loss are present in the appendix and supplementary materials. Figure 9 is one of those samples.

Table 3. (SDF) IoU (↑) results for all the methods and our PEPS counterparts. Loss is MAPE.

Model	PE	LPE	TI Grid	Grid-PEPS	Hash	hashPEPS	M-Grid	M-PEPS	M-Hash	M-HashPEPS
Lucy	0.893	0.902	0.904	*0.909*	0.640	0.876	0.902	0.908	**0.910**	**0.910**
Pitted Stonefish	0.316	0.389	0.410	**0.470**	0.339	0.428	0.413	*0.469*	0.460	**0.470**
Thai Statue	0.918	0.925	*0.926*	**0.930**	0.574	0.831	0.924	**0.930**	**0.930**	**0.930**
Armadillo	0.952	*0.956*	0.955	**0.957**	0.637	0.788	*0.956*	**0.957**	**0.957**	**0.957**
Global	0.770	0.793	0.799	*0.816*	0.547	0.731	0.799	*0.816*	0.814	**0.817**

Table 3 presents the overall results of the standard IoU metric for all the methods. Compared to the image and NTC cases, the use of multi-grids is not improving IoU results for SDF, but it is often improving the final rendering. Again, in all experiments, the PEPS counterparts show better IoU and better quality rendering. For the TI Grid, the introduction of PEPS improves the details of the render to a level of details higher than the multi-resolution grids. For the Hash, it is able to reduce the artifacts due to the hash table collision. Finally, for both multi-resolution versions, the introduction of PEPS both improved the metrics and the quality of rendering. Finally, the PEPS method seems to help unlock previously hard instances such as Pitted stonefish where the low resolution usually completely fails to render. More precisely, Table 4 shows the impact of increasing the number parameters of all the methods (except PE) for the hard instance Pitted Stonefish. The

goal of such a table is to show that the PEPS methods are able to achieve high reconstruction and that most of the other methods with 8 times more parameters are at the level of the grid-PEPS with 8 times less parameters. More results on that are provided in the appendix.

Table 4. (SDF) IoU (↑) results for all the methods using L1 loss. The bold line uses 8 times more parameters in the encoders (grid of resolution 64 instead of 32).

Model	Grid	hash	LPE	Grid-PEPS	PE	m-PEPS	m-grid	m-hash	m-hashPEPS
Pitted Stonefish	0.390	0.242	0.363	**0.453**	0.206	0.446	0.399	0.442	0.444
Pitted Stonefish	0.456	0.350	0.450	**0.466**	0.206	0.462	0.454	**0.466**	**0.466**

8 Conclusion

Implicit neural representations and positional encoding are some of the most efficient methods for learning coordinate to signal functions. In this paper, we proposed an analysis of the dynamic of projecting coordinates into the positional encoding space. We showed that the motion is of interest as it leads to a unique curve and can help the learning process. We proposed a generic framework called positional encoding projected sampling (PEPS) that tries to use the uniqueness of the positional encoding curve to design a basis for doing learned positional encoding. This method can be seen as a wrapper around encoders and we proposed two implementations of this framework. The first one; named grid-PEPS, is a wrapper around a simple grid and strongly improves its learning capabilities in our experimental section. The second one, named PinkPEPS, is an aggregation strategy based on the power spectral density of natural processing making the process faster and the results more accurate. We applied this method to three competitive applications - implicit image representation, texture compression, and signed distance functions - and showed that it was in all the benchmarking better than its counterpart and reach new state of the art results on quality of rendering.

References

AMD. 2016. HIP: C++ Heterogeneous-Compute Interface for Portability. https://github.com/ROCm/hip. Accessed: 2025-07-31.

AMD. 2025. RDNA4 Instruction Set Architecture: Reference Guide. https://www.amd.com/content/dam/amd/en/documents/radeon-tech-docs/instruction-set-architectures/rdna4-instruction-set-architecture.pdf. Accessed: 2025-07-31.

Pontus Andersson, Jim Nilsson, Tomas Akenine-Möller, Magnus Oskarsson, Kalle Åström, and Mark D Fairchild. 2020. FLIP: A Difference Evaluator for Alternating Images. *Proc. ACM Comput. Graph. Interact. Tech.* 3, 2 (2020), 15–1.

Wolfgang Erb, Christian Kaethner, Mandy Ahlborg, and Thorsten M Buzug. 2016. Bivariate Lagrange interpolation at the node points of non-degenerate Lissajous curves. *Numer. Math.* 133, 4 (2016), 685–705.

Farzad Farhadzadeh, Qiqi Hou, Hoang Le, Amir Said, Randall Rauwendaal, Alex Bourd, and Fatih Porikli. 2024. Neural Graphics Texture Compression Supporting Random Access. In *European Conference on Computer Vision*. Springer, 412–429.

ffish.asia / floraZia.com. 2023. CT scan Pitted Stonefish, E. erosa. https://sketchfab.com/3d-models/ct-scan-pitted-stonefish-e-erosa-0cdc3d1419384fd78fd952dc251a3169, Accessed: 2025-07-31, organization Sketchfab.

David J Field. 1987. Relations between the statistics of natural images and the response properties of cortical cells. *Journal of the Optical Society of America A* 4, 12 (1987), 2379–2394.

Shin Fujieda and Takahiro Harada. 2024. Neural Texture Block Compression. In *Workshop on Material Appearance Modeling*Joint MAM - MANER Conference - Material Appearance Network for Education and Research*, Jon Yngve Hardeberg and Holly Rushmeier (Eds.). The Eurographics Association. doi:10.2312/mam.20241178

Shin Fujieda, Atsushi Yoshimura, and Takahiro Harada. 2023. Local Positional Encoding for Multi-Layer Perceptrons. In *Pacific Graphics Short Papers and Posters*. The Eurographics Association. doi:10.2312/pg.20231273

Martin Gardner. 1978. White and brown music, fractal curves and one-over-f fluctuations. *Scientific American* 238, 4 (1978), 16–32.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. 2017. Convolutional sequence to sequence learning. In *International conference on machine learning*. PMLR, 1243–1252.

Augustine Gray and John Markel. 2003. Distance measures for speech processing. *IEEE Transactions on Acoustics, Speech, and Signal Processing* 24, 5 (2003), 380–391.

N Jeremy Kasdin. 1995. Discrete simulation of colored noise and stochastic processes and 1/f/sup/spl alpha//power law noise generation. *Proc. IEEE* 83, 5 (1995), 802–827.

Marvin S Keshner. 1982. 1/f noise. *Proc. IEEE* 70, 3 (1982), 212–218.

Eastman Kodak. 2022. Kodak lossless true color image suite (photocd pcd0992), 1993. *URL http://r0k. us/graphics/kodak* 10 (2022).

Belcour Laurent and Benyoub Anis. 2025. Hardware Accelerated Neural Block Texture Compression with Cooperative Vectors. *arXiv preprint arXiv:2506.06040* (2025).

Jules-Antoine Lissajous. 1857. *Mémoire sur l'étude optique des mouvements vibratoires*. Mallet-Bachelier.

Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. 2021. Nerf: Representing scenes as neural radiance fields for view synthesis. *Commun. ACM* 65, 1 (2021), 99–106.

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

Stanley Osher and Ronald Fedkiw. 2003. Signed distance functions. In *Level set methods and dynamic implicit surfaces*. Springer, 17–22.

François Pachet, Pierre Roy, Alexandre Papadopoulos, and Jason Sakellariou. 2015. Generating 1/f Noise Sequences as Constraint Satisfaction: The Voss Constraint.. In *IJCAI*. 2482–2488.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in PyTorch. In *NIPS-W*.

Guillaume Perez, Brendan Rappazzo, and Carla Gomes. 2018. Extending the capacity of 1/f noise generation. In *International Conference on Principles and Practice of Constraint Programming*. Springer, 601–610.

Ken Perlin. 1985. An image synthesizer. *ACM Siggraph Computer Graphics* 19, 3 (1985), 287–296.

Lawrence Rabiner and Biing-Hwang Juang. 1993. *Fundamentals of speech recognition*. Prentice-Hall, Inc.

Nasim Rahaman, Aristide Baratin, Devansh Arpit, Felix Draxler, Min Lin, Fred Hamprecht, Yoshua Bengio, and Aaron Courville. 2019. On the spectral bias of neural networks. In *International conference on machine learning*. PMLR, 5301–5310.

Sameera Ramasinghe and Simon Lucey. 2022. Beyond periodicity: Towards a unifying framework for activations in coordinate-mlps. In *European Conference on Computer Vision*. Springer, 142–158.

Vishwanath Saragadam, Daniel LeJeune, Jasper Tan, Guha Balakrishnan, Ashok Veeraraghavan, and Richard G Baraniuk. 2023. Wire: Wavelet implicit neural representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 18507–18516.

Rui Shi, Yishun Dou, Zhong Zheng, Xiangzhong Fang, Wenjun Zhang, and Bingbing Ni. 2025. Neural Block Compression: Variable Bitrates Feature Blocks for Texture Representation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 39. 6878–6886.

Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. 2020. Implicit neural representations with periodic activation functions. *Advances in neural information processing systems* 33 (2020), 7462–7473.

J Josiah Steckenrider, Mitchell Miller, Rory Blankenship, Victor Trujillo, and James Bluman. 2024. Lissajous curves as aerial search patterns. *Scientific Reports* 14, 1 (2024), 11144.

P. Szendro, G. Vincze, and A. Szasz. 2001. BIO-RESPONSE TO WHITE NOISE EXCITATION. *Electro- and Magnetobiology* 20, 2 (2001), 215–229. doi:10.1081/JBC-100104145

Towaki Takikawa, Joey Litalien, Kangxue Yin, Karsten Kreis, Charles Loop, Derek Nowrouzezahrai, Alec Jacobson, Morgan McGuire, and Sanja Fidler. 2021. Neural geometric level of detail: Real-time rendering with implicit 3d shapes. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 11358–11367.

Matthew Tancik, Pratul Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan Barron, and Ren Ng. 2020. Fourier features let networks learn high frequency functions in low dimensional domains. *Advances in neural information processing systems* 33 (2020), 7537–7547.

Karthik Vaidyanathan, Marco Salvi, Bartlomiej Wronski, Tomas Akenine-Moller, Pontus Ebelin, and Aaron Lefohn. 2023. Random-Access Neural Compression of Material Textures. *ACM Transactions on Graphics (TOG)* 42 (2023), 1–25.

van A Van der Schaaf and JH van van Hateren. 1996. Modelling the power spectra of natural images: statistics and information. *Vision research* 36, 17 (1996), 2759–2770.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems* 30 (2017).

Delio Vicini, Sébastien Speierer, and Wenzel Jakob. 2022. Differentiable signed distance function rendering. *ACM Transactions on Graphics (ToG)* 41, 4 (2022), 1–18.

Richard F Voss and John Clarke. 1975. 1/f noise in music and speech. *Nature* 258, 5533 (1975), 317–318.

Richard F Voss and John Clarke. 1978. "1/f noise"in music: Music from 1/f noise. *The Journal of the Acoustical Society of America* 63, 1 (1978), 258–263.

Yusong Wang, Shaoning Li, Tong Wang, Bin Shao, Nanning Zheng, and Tie-Yan Liu. 2023. Geometric transformer with interatomic positional encoding. *Advances in Neural Information Processing Systems* 36 (2023), 55981–55994.

Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. 2004. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing* 13, 4 (2004), 600–612.

Clément Weinreich, Louis De Oliveira, Antoine Houdard, and Georges Nader. 2024. Real-Time Neural Materials using Block-Compressed Features. In *Computer Graphics Forum*, Vol. 43. Wiley Online Library, e15013.

Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. 2018. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 586–595.

Zelin Zhao, Fenglei Fan, Wenlong Liao, and Junchi Yan. 2024. Grounding and enhancing grid-based models for neural fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 19425–19435.

A Global notes

Proof of proposition 2.1 . Suppose there are two non-null points (x, y) and (x', y') having different ratio a/b and a'/b' and equal Lissajous curve. As these ratio are different, either $a \neq a'$ or $b \neq b'$. Consider the first case. We must have $x \neq x'$. Yet, if they have the same Lissajous curve, we have $\sin(x\phi) = \sin(x'\phi)$ for any real ϕ. Using ϕ as the variable of our sines, the wave frequency of the left function is $\frac{2\pi}{x}$, for the second it is $\frac{2\pi}{x'}$. For two sine function to be equal they need to have at least the same frequency. This implies that $\frac{2\pi}{x} = \frac{2\pi}{x'}$, but $x \neq x'$. This is a contradiction. The same can be done for $b \neq b'$. Two points with different ratio cannot have the same Lissajous curve.

We will now focus on points having the same ratio a/b. Suppose that it exists two points having the same ratio and Lissajous curve. Those two points can be written by: $x = nt$,$y = mt$,$x' = nt\alpha$, $y' = mt\alpha$, with n, t, α a strictly positive number and $\alpha > 1$. Since their Lissajous curve is equal, we have: $\sin(nt\phi) = \sin(nt\alpha\phi)$ for any strictly positive real ϕ. Using ϕ as the variable of our sines, the wave frequency of the left function is $\frac{2\pi}{nt}$, for the second it is $\frac{2\pi}{nt\alpha}$. We should have $\frac{2\pi}{nt} = \frac{2\pi}{nt\alpha}$, but $nt < nt\alpha$ because $\alpha > 1$. This is a contradiction and the two points must be equal.

No sharing of the grid. We implemented several version of the grid-PEPS methods, versions where for each coefficient, a grid was defined and trained, and we implemented several strategy of parameters split between grids etc. This was in most of our experiments dominated by the single shared grid version of PEPS. One of the reasons for this is that the high frequency grid might miss a lot of the entries of the grid. This implies that dedicated grid might receive less gradient flow than shared one.

Full LPE. Learned positional encoding is usually difficult because the largest element might not be present during training. We performed an ablation study that only uses the APE coordinate and not the original one (we removed point x from Figure 1 from the main paper). In our results the impact was often marginal, yet always led to slightly less good results. This implies that PEPS could be an alternative for unseen position in the general case.

Aggregation function. We tested several aggregation functions instead of the concatenation of PEPS. We tried to sum all the latent vectors of the PEPS (additive learning). The results were drastically worse (several order of magnitudes). We tried to sum them by frequency ($l_{S_i} + l_{C_i}$), results were clearly better than the full sum, yet still dominated by the full concatenation one.

B Image Representation

In the paper, the use of a L1 loss, of a fixed learning rate and a simple leaky ReLU activation function was to ensure that the variability in the results was not a product of several potentially unstable components. In addition to the results in the paper, we tested our encoders using MLP having a GeLU activation function, two distinct learning rates (0.001 for the MLP, and 0.1 for the grids) and a cosine adaptative learning rate. The impact was that all methods have been improved, but the global

order remain the same. Table 5 shows the difference in results for the grid and ntc PEPS method for the kodak dataset Yet those adaptive learning rates and various activation functions often led to instability. Another example is the use of the WIRE activation function from [Saragadam et al. 2023]. Those led to large instability and forced decrease in the learning rate, and most importantly strictly longer training, for marginal gains. This prevent fair comparison. That is the reason why the paper is focusing on "vanilla" MLPs to ensure the stability of the result and the measurement of the impact of the encoder.

Table 5. Impact of the loss, architecture, and learning rate for the Kodak dataset.

Metric	Grid	G-P-PEPS	NTC_N
PSNR L1 (↑)	40.871	44.237	41.229
PSNR L2 (↑)	45.30	47.83	44.87
SSIM L1 (↑)	0.973	0.975	0.968
SSIM L2 (↑)	0.992	0.993	0.992

B.1 Metrics

Compression quality: PSNR.. Peak signal-to-noise ratio (PSNR) is a well-known metric used for the measurement of quality of lossy compression technics. It is defined by $PSNR(I, \hat{I}) = 20\log_{10}(Max(I)) - 10\log_{10}(MSE(I, \hat{I}))$, where MSE is the mean squared error.

Perceptual quality: SSIM and FLIP.. Structural Similarity Index Measure (SSIM) is a perception based measure trying to characterize of the degradation of the image. For a complete definition, please refer to [Wang et al. 2004]. In our implementation, we used the *StructuralSimilarityIndexMeasure* from the *torchmetrics.image* package. For the flip error [Andersson et al. 2020], we used the *flip_evaluator* python packaged. The FLIP error reported in this paper is the average of the FLIP error of an image.

Spectral quality: LSD and LSPD.. In this paper, we introduce the use of the two spectral metrics log-spectral distance (LSD) and log-power spectral distance (LPSD). Intuitively, the first one is an analysis of the spectral quality directly through its 2D Fourier transform. This measure can be seen as a global structure measure done through the average amplitude distance between phases. The second one is a direct measurement of the distance in power spectral density. While it is known that natural images follows a $1/f^{\alpha}$ PSD law, the LPSD quantify the distance between this law and the learned image. More precisely. the 2D Fourier transform of the image is 2 dimensionsial. Thos two dimensions are aggregated by taking the average statistics over all coordinate vectors whose norm 2 are equals. More precisely $\|f\|_2 = \|f'\|_2$ for $f, f' \in \mathbb{R}^2$ with the norm 2 defined by $\|f\|_2\sqrt{f_1^2 + f_2^2}$. The resulting graph is for example the one showed in Figure 3. Then the LPSD is simply the L1 distance between the original image and the learned one. More information about the PSD or those metrics can be found in [Field 1987; Gray and Markel 2003; Rabiner and Juang 1993; Van der Schaaf and van Hateren 1996]

C NTC

Dataset. We used 18 instances from https://polyhaven.com/ and https://ambientcg.com/. We extracted the available one from [Fujieda and Harada 2024; Weinreich et al. 2024] and some more. The complete list of instances is: bench vice 01, cardboard box 01, cannon 01, clay roof tiles 02, fabric pattern 07, garden gnome, garden sprinkler 01, wood planks, treasure chest, Paving Stones

070, Rails 001, red dirt mud 01, Aerial Rocks 02, Bricks 090, forest sand 01, Metal Plates 013, roof 09, Wood 063

Methods. The methods used in the NTC section of the paper are:

- BI grid. A basic grid of resolution r having in each discrete coordinate from $[0, r-1]^2$ a latent vector of dimension d. When an input coordinate is given this method extract the four closest latent vectors in the grid and applies a bilinear interpolation between them. The result is of dimension d.
- LPE. Local Positional Encoding [Fujieda et al. 2023], is a post process of the BI grid. It multiplies the result of the BI grid with the *local* positional encoding of the input. The local positional encoding being the positional encoding with respect to the cell of the grid used by the BI grid
- NTC_N. The neural texture compression [Vaidyanathan et al. 2023] is composed of 2 grids. The first one is a concatenation grid, which implies that the four closest latent vectors are returned by the grid. The second grid is a BI grid. The last element is the Tiled positional encoding, which correspong to the last 3 frequencies of the PE.

Two distinct learning rates (0.001 for the MLP, and 0.1 for the grids) and a cosine adaptative learning rate. the activation function of the network are Gelu for fair comparison with [Vaidyanathan et al. 2023], batch size is 60k, and 3k epochs of 40 batchs are done. For each batch, a random set of coordinate is sampled from pixel locations.

Results. We put in the supplementary random batches of samples from the 4K texture sets. The complete set of 4K results is about 10 GB, and cannot be part of the supplementary.

D SDF

The scan of Pitted Stonefish is available at [ffish.asia / floraZia.com 2023]. Other instances are from [Fujieda et al. 2023]. They are converted to SDFs using our application written in C++ and HIP. All rendering are available in the multimedia supplementary. We provide in this document two results for the L1 loss. The learning rate is 0.001, the activation function of the network are SILU, batch size is 60k, and 3k epochs of 40 batchs are done. For each batch, a random set of coordinate is sampled from $[0, 1]^3$. The bold line for stonefish is using 8 times more parameters for the encoder.

MAPE. Mean Absolute Percentage Error (MAPE) is a loss that measures the average magnitude of error in a set. It is defined by:

$$MAPE(p, gt) = 100\frac{1}{n}\sum_{i}^{n}\frac{|gt - p|}{|gt|}$$

Results for the four SDF instances, using the same setting of the paper and the L1 loss instead of MAPE loss are given in table 6. The only difference is the learning rate which is set to 0.01. As we can see, just like for images, the PEPS methods remain the best, and especially grid-PEPS that is clearly above.

Table 6. (SDF) IoU (↑) results for all the methods using the L1 loss.

Model	Grid	hash	LPE	Grid-PEPS	PE	m-PEPS	m-grid	m-hash	m-hashPEPS
Lucy	0.906	0.779	0.904	**0.909**	0.874	**0.909**	0.906	**0.909**	**0.909**
Pitted Stonefish	0.390	0.242	0.363	**0.453**	0.206	0.446	0.399	0.442	0.444
Thai Statue	0.927	0.747	0.925	**0.930**	0.900	**0.930**	0.928	**0.930**	**0.930**
Armadillo	0.956	0.761	0.956	**0.957**	0.943	**0.957**	**0.957**	**0.957**	**0.957**
global	0.795	0.632	0.787	**0.812**	0.731	*0.810*	0.798	*0.810*	*0.810*