Spline Positional Encoding for Learning 3D Implicit Signed Distance Fields

Peng-Shuai Wang[1] , **Yang Liu**[1] , **Yu-Qi Yang**[2,1] , **Xin Tong**[1]
[1]Microsoft Research Asia
[2]Tsinghua University
{penwan, t-yuqyan, yangliu, xtong}@microsoft.com

Abstract

Multilayer perceptrons (MLPs) have been successfully used to represent 3D shapes implicitly and compactly, by mapping 3D coordinates to the corresponding signed distance values or occupancy values. In this paper, we propose a novel positional encoding scheme, called *Spline Positional Encoding*, to map the input coordinates to a high dimensional space before passing them to MLPs, for helping to recover 3D signed distance fields with fine-scale geometric details from unorganized 3D point clouds. We verified the superiority of our approach over other positional encoding schemes on tasks of 3D shape reconstruction from input point clouds and shape space learning. The efficacy of our approach extended to image reconstruction is also demonstrated and evaluated.

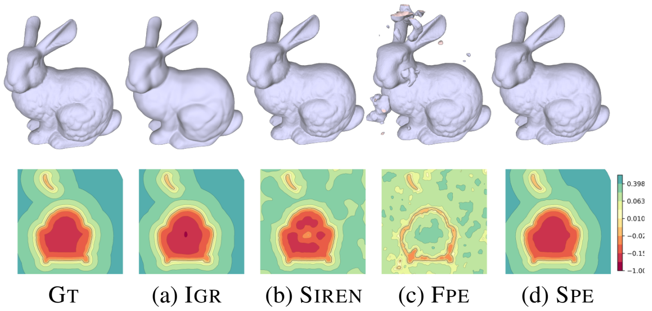


Figure 1: SDF learning via MLP-based methods. Upper: the extracted zero level set via marching cubes. Lower: a slice view of SDFs. SIREN [Sitzmann *et al.*, 2020], FPE [Tancik *et al.*, 2020] and our SPE fit the input point coordinates and normals well, but FPE contains many unwanted small branches. The SDFs from IGR [Gropp *et al.*, 2020] and SPE are more faithful to the ground-truth, while SPE recovers more details.

1 Introduction

Implicit neural representations learned via multilayer perceptrons (MLPs) have been proved to be effective and compact 3D representations [Park *et al.*, 2019; Mescheder *et al.*, 2019; Chen and Zhang, 2019] in computer vision and graphics fields. The MLPs take 3D coordinates as input directly, denoted by *coordinate-based MLPs*, and output the corresponding signed distance values or the occupancy values. They essentially define continuous implicit functions in 3D space whose zero level set depicts shape surfaces. Compared with conventional 3D discrete representations like point clouds or voxels, the MLP-based implicit representation has infinite resolutions due to its continuous nature, while being extremely compact. Apart from representing 3D shapes, coordinate-based MLPs are also capable of representing images, 3D textures, and 5D radiance fields, serving as general-purpose mapping functions.

In this paper, we are interested in learning the signed distance field (SDF) effectively from an unorganized input point cloud sampled from a 3D shape, by MLPs. SDF defines the distance of a given point $\mathbf{x}$ from the shape surface, with the sign determined by whether $\mathbf{x}$ is inside the shape volume or not. SDFs are needed by a broad range of applications [Jones *et al.*, 2006], including, but not limited to, 3D reconstruction [Curless and Levoy, 1996], constructive solid geometry (CSG), collision detection [Bridson, 2015] and volume rendering [Hart, 1996]. The recent MLP-based approaches [Gropp *et al.*, 2020; Atzmon and Lipman, 2020a] introduce the Eikonal equation constraint $|\nabla F(\mathbf{x})| \equiv 1$ to the mapping function $F : \mathbf{x} \in \mathbb{R}^3$ to enforce F to be an SDF, while its zero level set passes through the point cloud. However, due to the "spectral bias" of neural networks [Rahaman *et al.*, 2019; Mildenhall *et al.*, 2020; Tancik *et al.*, 2020], coordinate-based MLPs with ReLU activation are incapable of reconstructing high-frequency details of surfaces. An example produced by a coordinate-based MLP — IGR [Gropp *et al.*, 2020] is shown in Fig. 1(a), where the output shape is over-smoothed compared to the ground-truth.

To circumvent this problem, SIREN [Sitzmann *et al.*, 2020] uses Sine as the activation function in place of ReLU to improve the expressiveness of MLPs, and the Fourier Positional Encoding (abbreviated as FPE) [Mildenhall *et al.*, 2020; Tancik *et al.*, 2020; Zhong *et al.*, 2020] is proposed to improve network capability by lifting input coordinates to a high-dimensional Fourier space via a set of sinusoidal functions before feeding the coordinates as the input of MLPs. However, both approaches fail to recover SDFs in good quality and are even worse than IGR (see Fig. 1(b)&(c)), although their zero level sets may fit the point cloud well.

In this paper, we propose a novel *Spline Positional Encoding* (abbreviated as SPE), with which the MLP can not only recover the high-frequency details of the surface but also recover the SDF well, as shown in Fig. 1(d). Our SPE maps the input coordinates into a high-dimensional space via projecting them onto multiple trainable Spline functions, instead of hard-coded sinusoidal functions as FPE. The Spline functions are defined as the weighted sum of a series of uniformly spaced local-support B-Spline bases, and the weights are trainable. As the Spline function can be used to approximate other continuous functions, our SPE can be regarded as a generalization of FPE. SPE greatly increases the fitting ability of MLPs to reproduce high-frequency details. By subdividing the B-Spline bases, SPE can also be progressively refined. Based on this property, we also design a multi-scale training scheme to help MLPs converge to better local minima, which enables our network to recover SDFs and geometric details progressively and robustly.

Through experiments and ablation studies, we demonstrate the efficacy and the superiority over other state-of-the-art encoding schemes of our SPE on the tasks of learning SDFs from a point cloud or a set of point clouds. Additionally, to test the generalizability of SPE, we also apply it to image reconstruction and achieve good performance.

2 Related Work

Coordinate-based MLPs. The coordinate-based MLPs have caught great research interest as a continuous representation of shapes [Park *et al.*, 2019; Mescheder *et al.*, 2019; Chen and Zhang, 2019], scenes [Sitzmann *et al.*, 2019], images [Tancik *et al.*, 2020; Sitzmann *et al.*, 2020], textures [Oechsle *et al.*, 2019] and 5D radiance fields [Mildenhall *et al.*, 2020]. These methods train MLPs by regressing the ground truth SDFs, point/pixel colors, or volume radiance values. Our work is motivated by the works [Atzmon and Lipman, 2020b; Atzmon and Lipman, 2020a; Gropp *et al.*, 2020] that use MLPs to reconstruct SDFs from raw point clouds, without knowing the ground truth SDFs.

The limitation of coordinate-based MLPs with ReLU activation has been revealed by [Rahaman *et al.*, 2019; Mildenhall *et al.*, 2020]: the high-frequency fitting error decreases exponentially slower than the low-frequency error. To overcome this issue, there are multiple attempts to improve the representation power of MLPs as follows.

Activation function. SIREN [Sitzmann *et al.*, 2020] use Sine as the activation function and proposes a proper initialization method for training. It greatly improves the expressiveness of MLPs, and it is capable of recovering fine geometry details in the 3D reconstruction task. However, ReLU can provide strong implicit regularization when being under-constrained [Gropp *et al.*, 2020] and offer a good approximation to SDF in the whole space. In our work, we choose Softplus as our activation function, which can be regarded as a differentiable ReLU.

Positional encoding. Sinusoidal encoding is a kind of positional encoding that is first used for representing 1D positions in natural language processing [Vaswani *et al.*, 2017]. This type of positional encoding has proved to be able to improve the performance of MLPs in radiance fields fitting [Mildenhall *et al.*, 2020] and 3D protein structure reconstruction [Zhong *et al.*, 2020]. Tancik *et al.* [2020] build a theoretical connection between Sinusoidal mapping and Neural Tangent Kernels [Jacot *et al.*, 2018] for proving the efficacy of sinusoidal mapping and further improve its performance by using random Fourier features [Rahimi and Recht, 2008]. Their Fourier Positional Encoding (FPE) maps input points to a higher dimensional space with a set of sinusoids. However, FPE is not suitable to minimize the loss function containing function gradient constraints as we reveal in Section 4.

Local MLPs. Local MLPs improve the performance of a global MLP by dividing complex shapes or large-scale scenes into regular subregions [Peng *et al.*, 2020; Chabra *et al.*, 2020; Jiang *et al.*, 2020; Genova *et al.*, 2020] and fitting each subregion individually with the consideration of fusing the local output features or local output results. Our Spline Positional Encoding is composed of uniformly spaced locally supported basis functions along with different project directions. It shares the same sprite to local MLPs, but executes the local representation at the beginning of MLP.

3 Spline Positional Encoding

In this section, we first briefly review the loss functions for learning SDFs from a point cloud in Section 3.1, then introduce our spline positional encoding and its relations to prior arts in Section 3.2, and our training scheme in Section 3.3.

3.1 SDF Learning

Given a set of points with oriented normals sampled from the unknown surface $\mathcal{S}$ of a 3D shape, denoted by $\mathcal{X} = \{(\mathbf{x}_i, \mathbf{n}_i)\}_{i\in\mathcal{I}}$, the goal is to train an MLP $F(x)$ which represents the SDF of $\mathcal{S}$ and keeps $F(\mathbf{x}_i) = 0, \nabla F(\mathbf{x}_i) = \mathbf{n}_i$, $\forall i \in \mathcal{I}$. To ensure $F(x)$ is an SDF, an additional constraint from the Eikonal equation $\|\nabla F(x)\| = 1$ is added as recommended by [Gropp *et al.*, 2020]. The final loss function is in the following form.

$$\begin{aligned} L_{sdf} = & \sum_{i\in\mathcal{I}} (F(x_i)^2 + \tau \, \|\nabla F(\mathbf{x}_i) - \mathbf{n}_i\|^2) + \\ & \lambda \, \mathbb{E}_x(\|\nabla F(\mathbf{x})\| - 1)^2. \end{aligned} \tag{1}$$

After training, $F(\mathbf{x})$ approximates the underlying SDF induced by the input point clouds, and the zero level set $F(\mathbf{x}) = 0$ approximates $\mathcal{S}$, which can be extracted as a polygonal mesh via Marching Cubes [Lorensen and Cline, 1987].

3.2 Spline Positional Encoding

The key idea of our SPE is to use a set of parametric Spline functions as encoding functions. Different from FPE which uses predefined sinusoidal functions, our SPE is trainable and optimized together with MLPs. In our implementation, we choose the B-Spline function due to its simplicity.

B-Spline function. We first briefly introduce the B-Spline basis and B-Spline functions. The B-Spline basis $B^i(x)$ is a locally supported function, where $B^i : \mathbb{R} \mapsto \mathbb{R}$ and i is its polynomial degree. The $B^0(x)$ is defined as follows:

$$B^0(x) = \begin{cases} 1 & \text{if } |x| < 0.5; \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

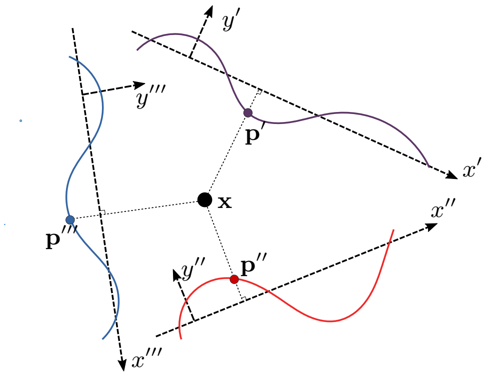


Figure 2: Illustration of Spline positional encoding on 2D. Point $\mathbf{x}$ is projected onto three Splines along three directions. The local heights of $\mathbf{p}'$, $\mathbf{p}'\prime$, $\mathbf{p}'''$ with respect to their own y-axis: y', y'', y''' form the Spline positional encoding of $\mathbf{x}$.

$B^i(x)$ is set to the n^{th} convolution of $B^0(x)$. The linear B-Spline basis $B^1(x)$ is supported in $[-1, 1]$, and the quadratic B-Spline $B^2(x)$ is supported in $[-1.5, 1.5]$. And for simplicity, we use $B(x)$ and omit the superscript.

Given an input 1D domain, we first uniformly subdivide it to K segments and get $(K + 1)$ knot points $\{c_i\}_{i=0}^K$, and denote the interval between two knots as δ. We scale and translate the B-Spline basis $B(x)$ to each knot point, and get $B_{\delta,c_i}(x) = B(\frac{x-c_i}{\delta})$. At the i^{th} node point, we define an optimizable parameter W_i. The parametric Spline function is defined as

$$\psi(x) = \sum_{i=0}^{K} W_i B_{\delta,c_i}(x). \tag{3}$$

If we further define W_i as a C-channel vector, we can obtain C spline functions.

Spline positional encoding. Without loss of generality, we assume the input domain for training MLP is $[-1, 1]^d$. We randomly select a set of unit directions $\mathbf{D}_1, \ldots, \mathbf{D}_m$, and these directions can determine a set of line segments with the same direction passing through the origin, whose two ends are on the unit sphere. On each line segment L_k, we can define a spline function ψ_k within the interval $[-1, 1]$. Given a point $\mathbf{x} \in [-1, 1]^d$, its spline positional encoding is defined as follows. We first compute the 1-D coordinate of $\mathbf{x}$ with represent to each direction $\mathbf{D}_k$, denoted by x_k, by projecting it onto L_k:

$$x_k := \langle \mathbf{x}, \mathbf{D}_k \rangle. \tag{4}$$

The SPE of $\mathbf{x}$ is defined as:

$$\mathbf{\Phi}(\mathbf{x}) = [\psi_1(x_1), ..., \psi_M(x_M)]. \tag{5}$$

To be able to differentiate different points in $\mathbb{R}^d$, the projection directions should be independent, and the projection direction number should be larger than d.

The above spline positional encoding lifts the point from a d-dimension vector up to a $C \times M$ tensor. In our experiments, we simply sum up the M projections and get a C dimension positional encoding. The total number of parameters used by SPE is $C \times (K+1) \times M + (d-1) \times M$. Here $(d-1) \times M$ is for the projection directions. All the parameters are differentiable in Eq. (5), thus can be trained to find their optimal values.

Relationship with prior positional encodings. For an MLP that directly takes coordinates as input, we can define its positional encoding as $\phi(\mathbf{x}) = \mathbf{x}$. The Fourier positional encoding proposed by [Tancik *et al.*, 2020; Mildenhall *et al.*, 2020; Zhong *et al.*, 2020] is composed of a set of sinusoidal functions with different frequencies, which can be defined as

$$\begin{aligned}\Phi(\mathbf{x}) = [&\sin(2\pi w_1^T x), \cos(2\pi w_1^T x), \cdots, \\ &\sin(2\pi w_M^T x), \cos(2\pi w_M^T x)].\end{aligned}$$

Since the spline function with sufficient knots can well approximate the Identity, Sine, and Cosine functions, our SPE can be regarded as a generalization of prior positional encodings. Actually, we can properly initialize W_i in Eq. (3) according to FPE and fix W_i during the optimization process and achieve the same effect as FPE.

3.3 Training Scheme

Multi-scale optimization of SPE. The B-Spline bases can be subdivided in a multi-scale manner, which is widely used in the multi-resolution optimization in Finite Element Analysis [Logan, 2017]. Suppose a Spline function is composed by K *linear* Spline bases, as defined in Eq. (3), we can refine it by subdividing the input domain to $2K$ segments and initialize the new weights $\hat{W}_i$ via the following formula:

$$\hat{W}_j = \sum_{i=0}^{K} W_i B_{\delta,c_i}(\hat{c}_j) \tag{6}$$

where $\hat{c}_j$ represents the j-th refined knot. Other higher-order Spline bases can also be subdivided similarly, and we omit the detailed formulas for simplicity. When training the network with the loss function, we first warm up the training process with a coarse resolution SPE. With a coarse SPE, the MLP quickly fits the low-frequency part of SDFs and provides a good initialization. Then we progressively refine SPE to increase the fitting ability of MLPs. In this way, our network can converge to better local minima: both the SDF away from the input points and the geometric details on the surface are better recovered.

Network training. By default, we use an MLP with 4 fully-connected (FC) layers with the Softplus activation function, each of which contains 256 hidden unit, and choose linear B-Spline bases for SPE. In each iteration during the training stage, we randomly sample 10k to 20k points from the input point cloud and the same number of random points from the 3D bounding box containing the shape. All input points are encoded via our SPE. We set the parameters of SPE to $K = 256, C = 64, M = 3$, resulting a 64 dimension encoding for each point. As a reference, with FPE, the dimension of per-point encoding is 256. The encoded point features are forwarded by the MLP. Then the loss in Eq. (1) is calculated. The parameters λ and τ in Eq. (1) are set to 0.1 and 1. The MLP and SPE are optimized via the Adam [Kingma and Ba, 2014] solver with a learning rate of 0.0001, without using the weight decay and normalization techniques.

For the multiscale optimization, we first initialize SPE with $K = 2$, then progressively increase K to 8, 32, 128, and 256, with the initialization method provided in Eq. (6). When

Model	Armadillo		Bimba		Bunny		Dragon		Fandisk		Gargogle		Dfaust	
	Chamfer	MAE	Chamfer	MAE	Chamfer	MAE	Chamfer	MAE	Chamfer	MAE	Chamfer	MAE	Chamfer	MAE
IGR	13.6	**1.9**	5.1	1.1	2.5	**0.7**	62.1	1.8	2.3	1.0	17.2	6.1	17.6	**1.4**
SIREN	2.2	22.1	5.6	18.9	1.5	16.3	**1.4**	2.3	243.3	20.3	2.8	17.0	9.3	32.9
FPE	207.5	28.3	3867.2	27.1	263.7	25.4	528.2	30.3	6956.8	27.7	7342.5	24.9	116.7	38.5
SPE	**1.3**	3.1	**1.6**	**0.6**	**1.5**	0.8	1.8	**2.1**	**1.3**	**0.4**	**2.4**	**1.3**	**9.1**	2.0

Table 1: Numerical results on SDF reconstruction from unorganized point clouds. The Chamfer distance and MAE are multiplied by 10000 and 100. Our SPE has much lower Chamfer distance than FPE and IGR, and better MAE than SIREN and FPE.

optimizing MLPs with $K = 2$, we occasionally observe the extracted surface containing spurious patches away from the input point cloud. Inspired by the geometric initialization proposed by [Atzmon and Lipman, 2020a] which initializes the network to approximate a sphere, we train a randomly initialized MLP to fit the SDF of a sphere. After training, the network weights are stored and used as the initialization of MLPs with $K = 2$ in SPE.

For learning shape spaces, we train an Auto-Decoder proposed by [Park *et al.*, 2019]. Instead of relying on a global shape code to identity each shape [Park *et al.*, 2019; Gropp *et al.*, 2020], our SPE itself is optimizable for each shape, which can be directly used to distinguish different shapes. Therefore, we train a shared MLP and specific SPE for each shape in the training set. The MLP is also composed of 4 FCs with 256 hidden units. After training, the network weights are fixed, and only the SPE is optimized to fit new shapes in the testing set.

4 Experiments and Evaluation

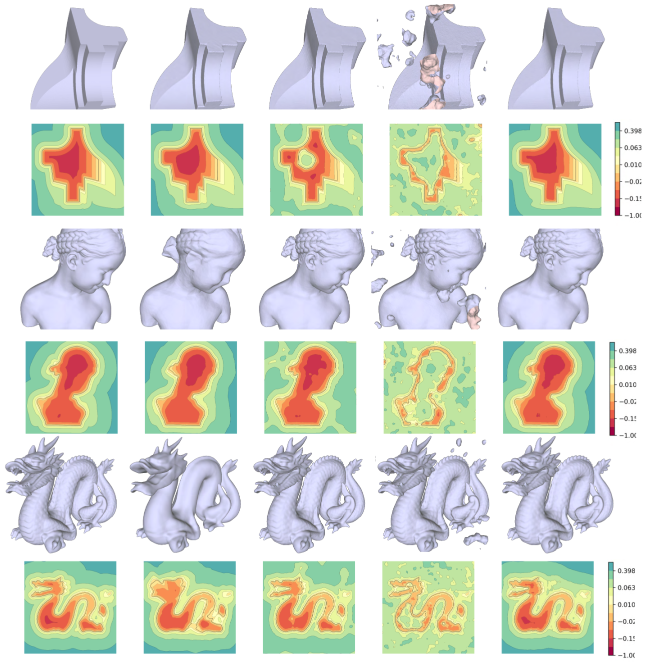
We have conducted the comparisons with several state-of-the-art methods to verify the effectiveness of our method. Specifically, we regard the MLP that directly takes the coordinates as input as the baseline, *i.e.*, IGR [Gropp *et al.*, 2020]. For the positional encoding, we compare our SPE with FPE proposed by [Tancik *et al.*, 2020], which is an enhanced and improved version of the positional encoding in [Mildenhall *et al.*, 2020; Zhong *et al.*, 2020], and SIREN [Sitzmann *et al.*, 2020]. By default, these networks are all composed of 4 FC layers with 256 hidden units.

Our implementation is based on PyTorch, and all experiments were done with a desktop PC with an Intel Core i7 CPU (3.6 GHz) and GeForce 2080 Ti GPU (11 GB memory). Our code and trained models are available at `https://wang-ps.github.io/spe`.

4.1 Single Shape Learning

In this section, we test our method on the task of reconstructing SDFs from raw point clouds. We evaluate both the quality of the reconstructed surface and SDFs.

Dataset. We collect 7 3D shapes as the benchmark, which include detailed geometric textures (Bunny, Armadillo, and Gargoyle), smooth surfaces (Bimba and Dragon), and sharp features (Fandisk). The Dfaust point cloud is produced by a real scanner provided by [Bogo *et al.*, 2017]. For other models, we sample points with normals from the surface via uniformly placing virtual depth cameras around each shape.



GT (a) IGR (b) SIREN (c) FPE (d) SPE

Figure 3: Visual comparisons on SDF reconstruction from raw point clouds. The reconstructed shapes and SDF slices are illustrated.

Evaluation metric. We use the Chamfer distance to measure the quality of the extracted surface. Specifically, we randomly sample a set of N points $\mathcal{X} = \{x_i\}_{i=1}^{N}$ from the extracted surface and ground-truth surface $\hat{\mathcal{X}} = \{\hat{x}_i\}_{i=1}^{N}$, where $N = 25k$. And we use the following formula to calculate the Chamfer distance:

$$D(\mathcal{X}, \hat{\mathcal{X}}) = \frac{1}{N}\sum_{i}\min_{j}\|x_i - \hat{x}_j\| + \frac{1}{N}\sum_{j}\min_{i}\|\hat{x}_i - x_j\| \tag{7}$$

We use the mean absolute error (MAE) between the predicted and ground-truth SDFs to measure the quality of the predicted SDFs. To calculate the MAE, we uniformly draw 256^3 samples on both the predicted and ground-truth SDFs.

Results. The numerical results are summarized in Table 1, and the visual results are shown in Fig. 3. As we can see, compared with IGR, the Chamfer distance of our method is greatly reduced, and the high-frequency geometric details are reconstructed much better, which verifies that with our SPE MLPs can easily reproduce the high-frequency details. For SIREN and FPE, their extracted surfaces may contain spurious

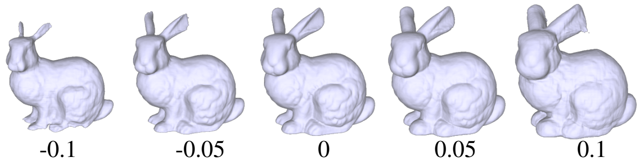


Figure 4: Different level sets of trained SDFs by SPE.

(K, C, M)	(128, 64, 3)	(**256**, 64, 3)	(128, **128**, 3)	(128, 64, **6**)	(128, 64, 3)*
Chamfer	1.72	1.63	1.67	1.69	1.71

Table 2: Chamfer distances on the Bimba model with different parameter settings. (128, 64, 3)* uses quadratic B-Spline bases.

components, as shown in Fig. 3. Note that it is non-trivial to fix this issue for their results: although the isolated small meshes can be easily removed, the incorrect components attached to the real surface are hard to remove and repair. Moreover, the implicit fields of FPE and SIREN have large deviations from the ground-truth SDFs, as revealed by significantly larger MAE. In Fig. 4, we show an application using SDFs trained with our SPE to extract the different level sets for shape shrinking and dilation.

4.2 Ablation Study

Expressiveness of SPE. We did an ablation study on how the choices of hyper-parameters (the segmentation number K, the channel of weights C, the projection number M, and the order of B-Spline basis), affect the performance of SPE. The experiments were done on the reconstruction of the Bimba model and we increased one hyper-parameter while keeping others unchanged. The baseline is $K = 128, C = 64, M = 3$ with linear B-Spline bases. The results summarized in Table 2 show that larger hyper-parameters can result in better apprimation quality, while the segmentation number brings the most effective improvements.

Can a vanilla MLP compete with SPE? To check whether a vanilla MLP can fit the high-frequency details, we train vanilla MLP -IGR by increasing its network depth and training time on the task of reconstructing the Bimba model. The results are shown in Fig. 5. As we can see, even by increasing the network depth by 4 times or increasing the training time by 10 times, the results of vanilla MLPs are still worse than SPE. Without any kind of positional encoding, the vanilla MLP converges too slow to recover the fine details.

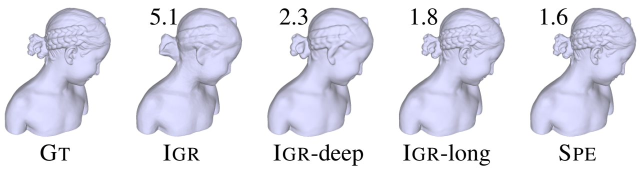


Figure 5: Test on vanilla MLP with different settings. From left to right: the ground-truth, the result of IGR as the baseline, IGR-deep: IGR with 4 times increased network depth, IGR-long: IGR with 10 times longer training time, and our SPE result. The numbers in the figure are the Chamfer distance.

Method	IGR	SIREN	FPE	SPE
Chamfer	14.1	15.3	18.1	**11.5**
MAE	3.6	34.7	30.4	**3.4**

Table 3: Comparisons of shape space learning on the D-Faust dataset.

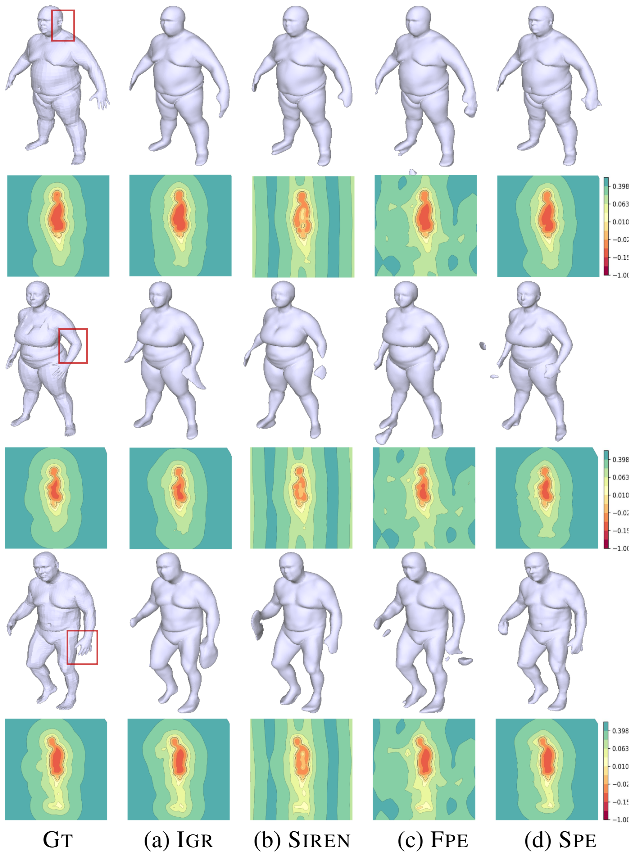


Figure 6: Visual comparisons on shape space learning. 3 human body shapes from the testing set reconstructed by different methods and the corresponding SDF slices are shown.

Specialization of our SPE. With proper initialization, our SPE can reproduce the MLP with FPE, since B-Spline functions can fit sinusoidal functions well with sufficiently small δ. In practice, we find that as long as the δ of B-Spline functions is similar to the period of a sinusoidal function, we can achieve similar effects.

4.3 Shape Space Learning

SPE is also suitable to learn shape spaces from raw scans using auto-decoders [Park *et al.*, 2019]. We compared our method with IGR, SIREN, and FPE, for which each shape has a specific shape code with dimension 256 and all the shapes share the same decoder. Our SPE optimizes the specific spline positional encoding for each shape and does not use a shape code. The parameters are set to $K = 64, C = 32, M = 3$. We use the Chamfer distance and MAE defined in Section 3.1 as the evaluation metrics to compare the quality of the reconstructed surface and SDFs on the unseen testing shapes.

Method	MLP	SIREN	FPE	SPE	SPE*
Natural Images	18.3	31.1	30.8	30.1	**33.6**
Text Images	18.4	35.6	33.7	37.4	**40.4**

Table 4: PSNRs of the image reconstruction task. SPE* is SPE with $M = 32$.

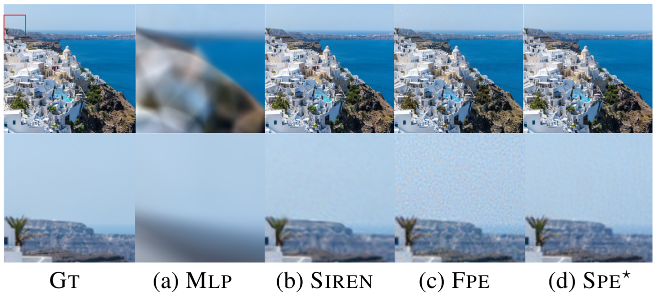


Figure 7: Results of image reconstruction. The bottom row shows zoom-in views.

Dataset. We conducted the experiment on the D-Faust dataset [Bogo *et al.*, 2017] by following the setup of [Gropp *et al.*, 2020]. D-Faust contains real scans of human bodies in different poses, in which 6258 shapes are used for training and 181 shapes are used for testing. For each raw scans, we sample 250k points with normals as the input.

Results. After training, all the methods are tested on the test shapes by fixing the network weights and optimizing the shape code or our spline weights. Table 3 lists the Chamfer distance and the MAE of SDFs of the results generated by four methods. It is clear that our method outperforms the other three methods, and the second-best method is IGR. Although SIREN and FPE have a better fitting ability in single shape reconstruction, they are even worse than IGR in the shape space learning within the auto-decoder framework. The visual comparisons in Fig. 6 further confirm the better performance of our method.

4.4 Generalizability of SPE

In this section, we show the generalizability of our method via the tasks of fitting images and SDFs.

Image fitting. In this experiment, we trained an MLP to map 2D coordinates to the corresponding pixel values. We use the image dataset provided by [Tancik *et al.*, 2020], which contains 16 natural images and 16 text images. The MLP is trained with an L_2 loss. We use PSNR as the evaluation metric. The results are summarized in Table 4. With similar size of parameters, our SPE achieves comparable results to SIREN and FPE, and is much better than a vanilla MLP. With more projection directions ($M = 32$), the performance of our SPE can be significantly improved. Fig. 7 illustrates the reconstruction results and zoom-in views by different methods. We notice that the resulting images of FPE have visible noise patterns, as shown by the zoom-in figure, and the result of Siren is more blur than ours.

SDF fitting. Instead of learning SDFs from raw points, we sample the ground-truth SDFs in the resolution of 256^3. The MLP takes 3D coordinates as input and output 1D distance values and directly fits the ground-truth SDFs with an L_1 loss [Park *et al.*, 2019]. We set the projection direction of our SPE as 16 to get smoother results. We use the same dataset in Section 4.1 and summarize the fitting results in Table 5. The zero isosurface results of the Fandisk example by different methods are illustrated in Fig. 8. It can be seen that the results of SPE and SIREN are more faithful to the ground truth than other methods, while the result of IGR is over-smoothed and the result of FPE has visible noise patterns.

Model	Arm.	Bimba	Bunny	Dragon	Fandisk	Garg.
MLP	9.69	4.69	3.76	11.59	2.06	9.94
SIREN	1.19	1.41	1.48	1.23	1.44	**1.66**
FPE	1.24	1.44	1.49	1.29	1.47	1.71
SPE	**1.18**	**1.39**	**1.46**	**1.19**	**1.36**	1.69

Table 5: Comparisons on the SDF regression task. The numbers are Chamfer distances.

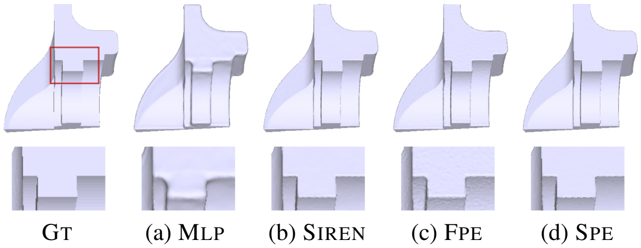


Figure 8: Comparisons on shape regression. The results are shown in the first row, and the zoom-in figures are shown in the second row.

5 Conclusion

We present a novel and effective Spline positional encoding scheme for learning 3D implicit signed distance fields from raw point clouds. The spline positional encoding enhances the representation power of MLPs and outperforms the existing positional encoding schemes like Fourier positional encoding and SIREN in recovering SDFs.

In the future, we would like to explore SPE in the following directions.

Non-uniform spline knots. Compared with uniform knots we used in spline functions, non-uniform knots provide more freedom to model complex and non-smooth spline functions and would also help reduce the parameter sizes of SPE while keeping the same approximation power.

Composition of positional encoding. As positional encoding has proved to be an effective way to distinguish nearby points in a higher dimension space, it would be interesting to composite multiple scale SPEs to strengthen the capability of SPE while using fewer parameters for each SPE.

Acknowledgements

We wish to thank the anonymous reviewers for their valuable feedback.

References

[Atzmon and Lipman, 2020a] Matan Atzmon and Yaron Lipman. SAL: Sign agnostic learning of shapes from raw data. In *CVPR*, 2020.

[Atzmon and Lipman, 2020b] Matan Atzmon and Yaron Lipman. SAL++: Sign agnostic learning with derivatives. *arXiv preprint arXiv:2006.05400*, 2020.

[Bogo *et al.*, 2017] Federica Bogo, Javier Romero, Gerard Pons-Moll, and Michael J Black. Dynamic FAUST: Registering human bodies in motion. In *CVPR*, 2017.

[Bridson, 2015] Robert Bridson. *Fluid simulation for computer graphics*. CRC press, 2015.

[Chabra *et al.*, 2020] Rohan Chabra, Jan Eric Lenssen, Eddy Ilg, Tanner Schmidt, Julian Straub, Steven Lovegrove, and Richard Newcombe. Deep local shapes: Learning local SDF priors for detailed 3D reconstruction. In *ECCV*, 2020.

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

[Curless and Levoy, 1996] Brian Curless and Marc Levoy. A volumetric method for building complex models from range images. In *SIGGRAPH*, 1996.

[Genova *et al.*, 2020] Kyle Genova, Forrester Cole, Avneesh Sud, Aaron Sarna, and Thomas Funkhouser. Local deep implicit functions for 3D shape. In *CVPR*, 2020.

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

[Hart, 1996] John C Hart. Sphere tracing: A geometric method for the antialiased ray tracing of implicit surfaces. *The Visual Computer*, 12(10), 1996.

[Jacot *et al.*, 2018] Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. In *NeurIPS*, 2018.

[Jiang *et al.*, 2020] Chiyu Jiang, Avneesh Sud, Ameesh Makadia, Jingwei Huang, Matthias Nießner, and Thomas Funkhouser. Local implicit grid representations for 3D scenes. In *CVPR*, 2020.

[Jones *et al.*, 2006] Mark W Jones, J Andreas Baerentzen, and Milos Sramek. 3D distance fields: A survey of techniques and applications. *IEEE. T. Vis. Comput. Gr.*, 12(4), 2006.

[Kingma and Ba, 2014] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *ICLR*, 2014.

[Logan, 2017] Daryl L Logan. *A first course in the finite element method*. Nelson Education, 2017.

[Lorensen and Cline, 1987] William E. Lorensen and Harvey E. Cline. Marching Cubes: A high resolution 3D surface construction algorithm. In *SIGGRAPH*, 1987.

[Mescheder *et al.*, 2019] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3D reconstruction in function space. In *CVPR*, 2019.

[Mildenhall *et al.*, 2020] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. NeRF: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020.

[Oechsle *et al.*, 2019] Michael Oechsle, Lars Mescheder, Michael Niemeyer, Thilo Strauss, and Andreas Geiger. Texture fields: Learning texture representations in function space. In *ICCV*, 2019.

[Park *et al.*, 2019] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. DeepSDF: Learning continuous signed distance functions for shape representation. In *CVPR*, 2019.

[Peng *et al.*, 2020] Songyou Peng, Michael Niemeyer, Lars Mescheder, Marc Pollefeys, and Andreas Geiger. Convolutional occupancy networks. In *ECCV*, 2020.

[Rahaman *et al.*, 2019] Nasim Rahaman, Aristide Baratin, Devansh Arpit, Felix Draxler, Min Lin, Fred Hamprecht, Yoshua Bengio, and Aaron Courville. On the spectral bias of neural networks. In *ICML*, 2019.

[Rahimi and Recht, 2008] Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. In *NeurIPS*, 2008.

[Sitzmann *et al.*, 2019] Vincent Sitzmann, Michael Zollhöfer, and Gordon Wetzstein. Scene representation networks: Continuous 3D-structure-aware neural scene representations. In *NeurIPS*, 2019.

[Sitzmann *et al.*, 2020] Vincent Sitzmann, Julien NP Martel, Alexander W Bergman, David B Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. In *NeurIPS*, 2020.

[Tancik *et al.*, 2020] Matthew Tancik, Pratul P. Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan T. Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. In *NeurIPS*, 2020.

[Vaswani *et al.*, 2017] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017.

[Zhong *et al.*, 2020] Ellen D Zhong, Tristan Bepler, Joseph H Davis, and Bonnie Berger. Reconstructing continuous distributions of 3D protein structure from cryo-EM images. In *ICLR*, 2020.