Local Smoothness Prior for Learning Neural Implicit Representations

Zhijie Wu
zhijiewu@cs.ubc.ca

Guanxiong Chen
gxchen@cs.ubc.ca

Xindong Lin
xlin114@student.ubc.ca

Tianyue Pan
sampty@student.ubc.ca

Helge Rhodin
rhodin@cs.ubc.ca

The University of British Columbia

Abstract

State-of-the-art coordinate-based neural networks often struggle to capture complex high-frequency details. A fundamental problem is reconstructing faithful details over regions with high variations while refraining from producing artifacts for smooth inputs. Based on the observation that high-frequency signals usually appear at boundaries, we propose the local smoothness prior to alleviating this problem: in signal space, two spatially nearby points with similar geometric relations to boundaries should possess similar network features. Thus sharp boundaries can be well preserved, and noisy artifacts can be filtered out. Following this line of thought, we develop a framework with two branches: one processes a queried point's Euclidean coordinates directly; another encodes the coordinates into a feature representing the point's relation to boundaries. By mixing the two-branched features in a multi-scale manner, our position-based local smoothness prior improves upon existing work in terms of adaptive detail modeling and network convergence when benchmarked on image synthesis and shape auto-encoding. Ablation studies for our critical architectural choices are also conducted to highlight our conceptual and empirical advantages.

1 Introduction

Deep implicit approaches [36, 39, 30, 13, 10, 14, 8] have appeared to be a powerful tool for modeling signals. Compared to conventional discrete representations [52, 40, 41, 16, 23, 22], neural implicits are continuous and thus enjoy strong capabilities in processing unknown number of vertices and arbitrary topology [44, 54]. Recently, neural implicits have attracted numerous academic attentions in a number of tasks ranging from 3D reconstruction [36, 30], neural rendering [31, 3], image translation [7] to deformation approximation [53].

Among these approaches, a central problem is how to capture fine-grained details over noisy areas but alleviate undesired artifacts for smooth parts, driving two streams of ideas. The first one is to partition the holistic 3D spaces into smaller ones [8, 10, 49, 5, 13, 14, 46, 28, 32]. While these methods enjoy the state-of-the-art empirical advantages, they are limited by in-compact representations, and their memory demands grow cubically with the spatial resolution. Another line of works are to convert the input signals to a high-frequency series via either periodic activation functions [44, 6] or Fourier transformations [31, 47] in an economical way. However, they frequently undergo convergence issues and are easily stuck in local minima; thus, they cannot attend to significant frequency variations and often introduce artifacts in flat regions. They fail to target locality and orientation aspects as well.

Preprint. Under review.

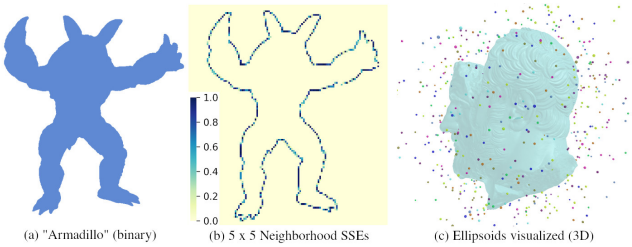

Figure 1: (a) A rasterized binary image of "Armadillo" [2]. (b) The heatmap for the squared errors over 5-by-5 neighborhoods. Neighborhoods near the object's boundary manifests greater variability, showing that nearby points with different geometric relations to boundaries will have different RGB values. (c) The learned Gaussian ellipsoids visualized on "Beard Man" [2].

To address these limitations, several efforts have been made as spatial adaptation schemes [17, 29] through combining the two mentioned idea streams. For example, Hertz et al. [17] propose to progressively encode the input coordinates by attending to time-spatial information jointly, while Mehta et al. [29] explicitly decompose the inputs into patches, based on which the activation functions are modulated. Nevertheless, these methods are still limited because they all depend on the uniform spatial adaptation strategies and thus are weak in capturing complex signals compactly.

We introduce a simple local smoothness prior to classical edge-aware filtering works [48, 11, 21, 20], which states that two spatially nearby points with similar relations to boundaries should possess similar network features. Specifically, we employ a structured set of non-uniform learnable Gaussians to represent the sharp boundaries, where each Gaussian defines a local coordinate system and provides more flexibility for orientation, scaling, and translation learning. Computing distances in local frames allows us to weigh up the corresponding features for each Gaussian. Thus, two nearby points with similar geometric relationships to the same set of Gaussians will have similar primitive feature vectors, meeting local smoothness requirements. To fully leverage the potential of local smoothness prior, we combine it with a spatial adaptation strategy for frequency transformation. In detail, we feed the weighted primitive features to further smooth the features from frequency transformation.

For each input point, we respect k-nearest primitives to encode locality, thus saving computational cost. Therefore we can count on a small number of primitives but take advantage of the non-uniform modeling through translation, rotation, and scaling learning economically with three benefits: 1) Rather than the fixed positions of regular primitives, we can explicitly model the movable locations, orientations, and scales, which are proved to be three essential elements for representation learning; 2) Compared to the vanilla coordinate-based networks, we can parse locality descriptions through relative relation modeling; 3) Imposing the smoothness in feature space enables filtering out the noisy artifacts in flat areas but preserve high-frequency signals near boundaries.

Combining the irregularity of non-uniform primitive learning and the success of frequency transformation-based spatial adaptations excels in learning implicit functions. Additionally, our local smoothness prior can be easily extended to various tasks, showing improvement consistently.

2 Related Work

In this section, we cover representative approaches for implicit neural representations [26, 4, 36, 47, 32, 46] and primitive-based learning framework respectively.

Implicit neural representations. Building neural implicit representations of time and space-varying signals [36, 39, 30, 13, 10, 14, 8] has become an active area of research. Whilst tremendous progress has been made with neural implicits for RGB images [39, 7, 6], 3D geometries [30, 47, 5, 46, 35], audio signals [44], or any physical signals implicitly defined within a differential equation in general [44], neural implicits can still fall short of representing fine details [31], and incur high computational cost due to model complexity [18]. Prior works attempt to solve these problems by *parametric encodings* [46, 14, 32] and *frequency transformations* [44, 47, 31].

For *frequency transformations* [32], Vaswani et al. [51] encode the input feature vector into a higher dimension space using a sequence of periodic functions. Later, Tancik et al. [47] used a Fourier Feature mapping for the input coordinates to improve the result fidelity. Meanwhile, Sitzmann et

al. [44] insert the periodic functions into the intermediate network to facilitate high-order derivative learning. Recently, Lindell et al. [24] propose to advance and analyze spectral characteristics with a multi-scale strategy. They tend to improve high-frequency output details while still maintaining reasonable computation costs.

For *parametric encodings* [32, 46], the idea is basically to encode the input coordinates of an arbitrary query point by interpolating a learnable basis consisting of grid-point features (space partitioning) or parameters of shape primitives. Although dense grids of trainable features are significantly faster to train and tend to yield more accurate results, they incur a higher memory footprint than neural network weights. Similarly, Müller et al. [32] stores the trainable feature vectors in a compact hash table, whose size is a tunable hyperparameter to control the reconstruction quality.

Later, some works try to combine the advantages of *frequency transformations* and *parametric encodings*. For example, SAPE [17] progressively encodes the input coordinates by attending to time-spatial information jointly. Mehta et al. [29] decompose the inputs into patches, which are used to modulate the activation functions. Similarly, we explicitly investigate the basis signals to support detail synthesis like [46, 25]. However, the critical difference is that we employ a set of low-dimensional irregular primitives to approximate concentrated feature regions, based on which we do not need to store dense feature vectors during training. In contrast, these *parametric encodings* based methods are weak in capturing fine-grained patterns compactly due to their uniform grid size. Thus we can benefit from better scalability to large scenes and more adaptive detail representations.

Primitive-based learning. Our work is also related to the shape abstraction techniques, whose goal is to decompose input signals (e.g. 3D shapes) into semantically meaningful simpler elements using primitive parameters [55, 34, 22, 13, 14]. Previous primitive-based works apply cuboids [50, 34, 55], superquadrics [37, 38], convexes [10, 12, 8] and CSG trees [43] as the atomic ingredients. Due to the simplicity and empirical success, like [14, 13], we employ a structured set of ellipsoids for the shape decomposition. Each orientated ellipsoid provides a local coordinate system, providing more flexibility for orientation, scaling, and translation learning. The differences lie in 1) Genova et al. [14, 13] use ellipsoid templates for implicit shape modeling via SDF evaluation at the end of a network while we perform the primitive learning for feature smoothing. Thus we are not limited to the shape reconstruction task but enjoy more comprehensive application extensions; 2) [14, 13] compute the distances between each query point and all ellipsoid frames but we depend on the closest neighbors for the relative distance computation, advancing the training speed and the learning capability for locality; 3) We communicate the relationships between spatially-adaptive frequency transformation and primitive-based feature smoothing, which is not the case for [14, 13]. These differences enable unique advantages over existing neural implicits learning frameworks.

3 Local Smoothness Layer

In this project, we devise a new scheme to encode input positions so that we can not only allow fine details to be reconstructed with greater accuracy, but also demonstrate strong capability in reducing artifacts in smooth regions. To this end, we design a network module specialized to capture the local smoothness prior for input coordinates, thus naming it as the Local Smoothness Layer (LS Layer).

In Fig. 2, we illustrate the structure of a neural implicit pipeline with a LS Layer inside. Given an input $n \times d$ matrix containing the spatial coordinates of n query points in a d-dimensional space, the LS Layer outputs its concatenated primitive feature vectors. Specifically, we approximate the input signal space under a basis of m basic elements each parameterized as a scaled orientated anisotropic Gaussian. We pick the Gaussian primitive due to its simplicity, efficiency, accuracy and scalability [13, 14, 42, 45]. Each Gaussian consists of a geometric center $p \in \mathbb{R}^d$, a set of Euler angles $e \in \mathbb{R}^d$ to orientate the Gaussian element from the canonical frame, and the corresponding radii $r \in \mathbb{R}^d$ for the anisotropic learning.

To leverage locality and reduce computation overheads when computing the primitive feature vectors, we identify the nearest Gaussian ellipsoid for every query point $x \in \mathbb{R}^d$. Compared with using the entire set of m ellipoids as the basis, using the k nearest ellipsoids exploits locality, reduces computational cost significantly, and is more robust to fine-grained geometric details. As shown in Fig. 9, even with small k's, we can achieve reasonable synthesis results.

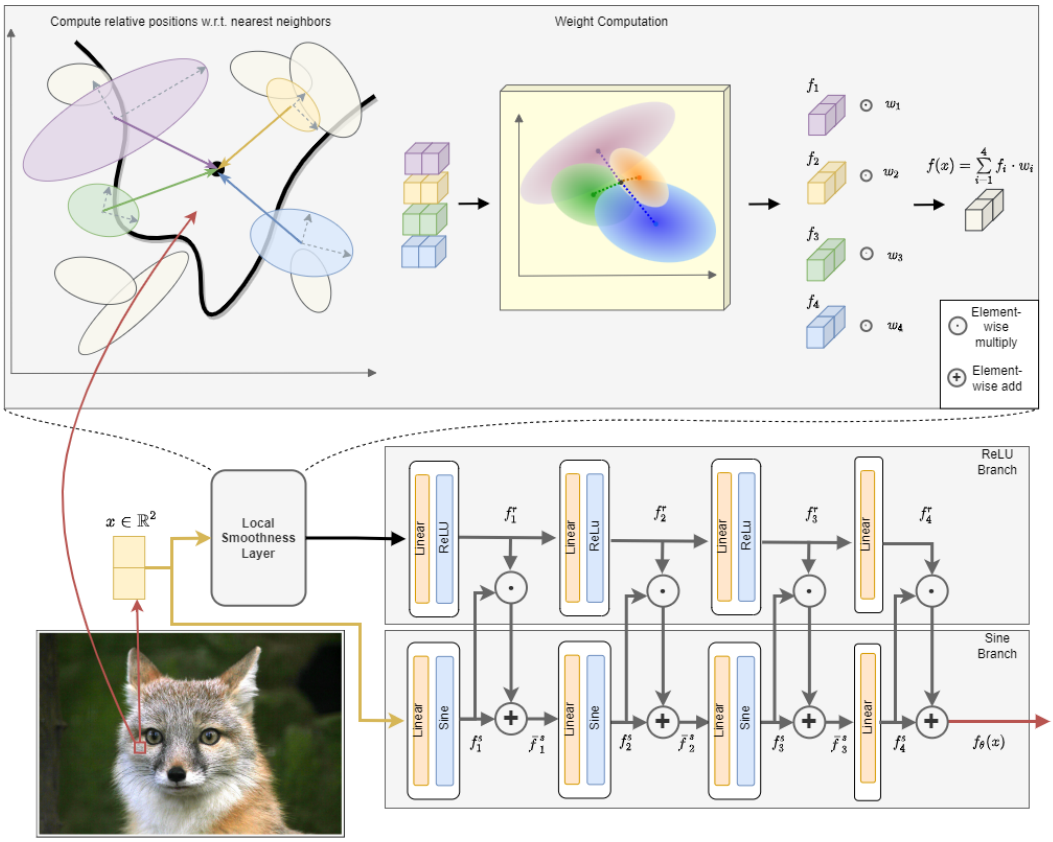

Figure 2: Architecture overview. Without loss of generality we consider inputs in the 2D Euclidean space.

Given the i-th computed closest ellipsoid parameterized as $\{p_i, e_i, r_i\}$, we define the relative position $\bar{x}_i$ of point x with respect to ellipsoid i as follows: first we translate the query point x as

$$t(x, p_i) = x - p_i.$$

Then we rotate the local frame of $t(x, p_i)$ as:

$$r(x, e_i) = R(e_i) \cdot t(x, p_i),$$

where $R(e_i)$ indicates the rotation matrix defined by e_i. Next, we scale each frame axis and get $\bar{x}_i$ as:

$$\bar{x}_i = s(x, r_i), \quad s(x, r_i) = r_i \odot r(x, e_i),$$

where $\odot$ represents element-wise multiplication. At last, we compute the relative distance by $\{\bar{x}_i\}$ as weights and define the weighted primitive feature vectors as

$$f(x) = \sum_{i \in [K]} f_i \cdot w_i, \qquad w_i = \frac{\lambda_i}{\sum_{i \in [K]} \lambda_i}, \qquad \lambda_i = e^{-\frac{\|\bar{x}_i\|^2}{2}},$$

where f_i and $[K]$ are the learnable feature vector of i-th Gaussian and the found neighbor index set respectively. For instance, we set $K = 4$ for 2D images and $K = 8$ for 3D shapes. By aggregating all neighboring primitive features when defining $f(x)$, we embed position, orientation and scale information in the approximated relative relationships of a point with respect to the signal space.

Finally, we feed $f(x)$ into a ReLU-based branch to further gradually modulate frequency transformation-based features. The motivating reason for primitive feature learning is that we can make the spatially-adaptive feature smoothing so that desired details near surfaces are well preserved but artifacts in the flat regions are reduced, which can be seen in Fig. 1.

Comparison with SOTAs. Compared to state-of-the-art coordinate encoding algorithms, such as the spatial partition methods [32, 46, 28], we argue that our method is advantageous in that (1) The

size of our primitive features is small. We only use a 9D vector to portray each 3D ellipsoid, which is the minimum necessary to achieve good results. In contrast, the spatial partition methods have to keep dense feature maps in memory throughout the entirety of training and inference; (2) The primitive sets are irregularly distributed over the 3D space, therefore more adaptive to fine input details; (3) Our combination of the spatial adaptations for frequency-based features and non-uniform primitive learning further advances high-frequency signal reconstruction and reduces network analytic parameters in an elegant manner.

4 Pipeline Overview

As illustrated in Fig. 2, our general architecture consists of one ReLU-based branch equipped with one local smoothness layer for primitive feature learning and another Sine-based branch to regress fine-grained details. To estimate the implicit signals, we take the Euclidean coordinate of a query point $x \in \mathbb{R}^d$ as input and output a set of feature vectors $\{f_1^s, f_2^s, \ldots, f_L^s\}$ for the Sine-based branch, where L represents the linear layer number. Similarly, taking $f(x)$ as input, the ReLU-based branch will output another set of feature vectors $\{f_1^r, f_2^r, \ldots, f_L^r\}$.

For the i-th scale, before feeding into the next linear layer, we combine f_i^r with f_i^s to filter f_i^s out as

$$\bar{f}_i^s = f_i^s + f_i^s \odot f_i^r.$$

Here $\odot$ represents the feature element-wise multiplication. In this way, we can achieve more compatible residual learning and feature smoothing for f_i^s.

Finally, we estimate the final output $f_\theta(x)$ by combining f_L^r and f_L^s as

$$f_\theta(x) = f_L^r + f_L^s \odot f_L^r.$$

Our framework can be trained for a multitude of applications: for instance, it can fit a signed distance function (SDF) in 3D space or an RGB image in 2D space. We train the network by minimizing a task-dependent loss function, where the task can be but not limited to 2D image synthesis or 3D surface reconstruction. For 2D image synthesis, we use the following loss:

$$\mathcal{L}_{\text{2D}}(y_{rgb}, f_\theta(x)) = \|y_{rgb} - f_\theta(x)\|_2,$$

where the y_{rgb} are the ground truth RGB color of the input pixel and $f_\theta(x)$ is the output of the network. And for the 3D surface reconstruction, we use the same L1-loss as DeepSDF [36]:

$$\mathcal{L}_{\text{3D}}(s, f_\theta(x)) = \|s - f_\theta(x)\|_1,$$

where s is the ground truth signed distance and $f_\theta(x)$ is the output of the network. The clamp of SDF values also enables more concentrated surface reconstruction [36].

Implementation details. Currently, we are using the same network setting like [36, 32, 46, 47], which composes of 5 fully connected layers, where each layer has 256 neurons. For the ReLU-based branch, we apply the ReLU activation [33] while for the Sine-based branch, we use sine function for activation. The weights of ReLU-based branch are initialized using a Xavier initialization [15]. Correspondingly, we follow the default setting in [44] to initialize the Sine-based branch. For optimizing the network, we use Adam [19] with parameters $\beta_1 = 0.9$, $\beta_2 = 0.99$, $\epsilon = 10^{-5}$ and a learning rate of $l = 0.0001$.

5 Experiments

We demonstrate the benefits of our coordinate encoding scheme on (1) 1D signal synthesis; (2) 2D high-resolution image synthesis; (3) 3D shape reconstruction. For each task, we present experimental results to show that by leveraging the local smoothness layer, our spatially adaptive frequency encoding scheme achieves reconstruction quality better than state-of-the-art frequency-based methods [44, 24, 17, 47], attends to fine geometric details, without introducing spurious artifacts to smooth regions. Further, we demonstrate that (1) The LS Layer and the learnable affine transformations of shape primitives within it contribute significantly to the representation power of our pipeline; (2) Our nearest neighborhood search can effectively leverage locality. All experiments are run on a single RTX 3090 of 24GiB memory.

5.1 1D Signal Synthesis

We use one-dimensional temporal signal synthesis as a motivating task to demonstrate the effectiveness of spatially adaptive positional encoding. The 1D ground-truth signal (Figure 3) contains multiple frequency components: near the ends, the signal changes rapidly over time, implying the presence of high-frequency components; toward the center, we have a smooth region in which low-frequency components are dominant. We train our network to predict a range value from temporal coordinates. We notice from the figure that (1) Positional encoding is crucial to learning high frequency components. In Figure 3 (a) we ablate the LS Layer from our pipeline but the network struggles to learn both high and low-frequency components. (2) Existing frequency-based methods that attends to all frequencies across the entire spectrum uniformly in space, such as SIREN [44] and FFN [47] fails to fit smooth regions despite performing fairly well on boundaries with high-frequency components. (3) Spatially adatpive methods such as SAPE [17] and ours can effectively learn both high and low-frequency components.

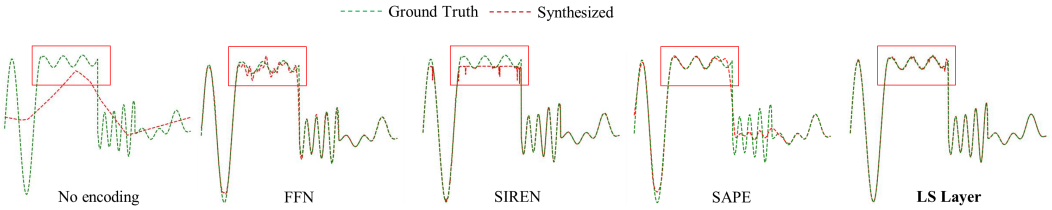

Figure 3: Qualitative comparison of 1D signal synthesis.

5.2 High-Resolution Image Synthesis

Prior work on neural implicit representation have demonstrated incredible ability to fit high-resolution images [28, 32]. Specifically, the task is to predict RGB color from 2D pixel coordinates. We benchmark the performance of our method on this task with state-of-the-art frequency-based methods: 1) FFN [47], 2) SIREN [44], 3) SAPE [17], 4) BACON [24]; also we compare our performance with InstantNGP [32] as a parametric encoding method. We curate images from a multitude of sources; these include prior work [28] as well as Flickr Creative Commons [1]. The resolutions of our images range from 1K to 6K; see the Supplemental for details.

Method \ Task	2D		3D			
	SSIM (↑)	PSNR (↑)	Chamfer Distance (↓)	Metro (↓)	F_score (↑)	IoU (↑)
InstantNGP	7.773e-1	2.974e1	2.723e-3	1.693e-2	9.972e-1	9.996e-1
FFN	6.857e-1	2.625e1	9.209e-3	37.19e-2	7.502e-1	9.914e-1
SIREN	7.707e-1	3.009e1	4.223e-3	1.729e-2	6.450e-1	9.917e-1
BACON	2.489e-1	2.001e1	2.856e-3	1.880e-2	9.921e-1	9.982e-1
SAPE	7.379e-1	2.766e1	4.348e-3	1.778e-2	6.084e-1	9.912e-1
Ours	**8.495e-1**	**3.314e1**	**2.789e-3**	**1.684e-2**	**9.918e-1**	**9.986e-1**

Table 1: Quantitative comparison of our method against the baselines on 2D and 3D signal fitting.

Table 1 shows the average SSIM and PSNR over "Tokyo" (6K-by-2k resolution) [28] and "Einstein" (3k-by-4k resolution) [32]. We train each network to overfit each image until loss converges. We see that our method surpasses all baselines on the two metrics.

We also qualitatively compare our reconstruction results with the baseline on the aforementioned two images. We see from Figure 4 that on "Tokyo" our method attends to fine details better than prior frequency-based methods. In 5 we show that on "Einstein", our pipeline not only preserves high-frequency details but also precludes spurious artifacts in smooth regions better than other baselines which employ uniformly large bandwidths. This observation suggests our spatially adaptive frequency encoding scheme does allow the network to pick up high frequency details only in regions enriched with fine geometric structures.

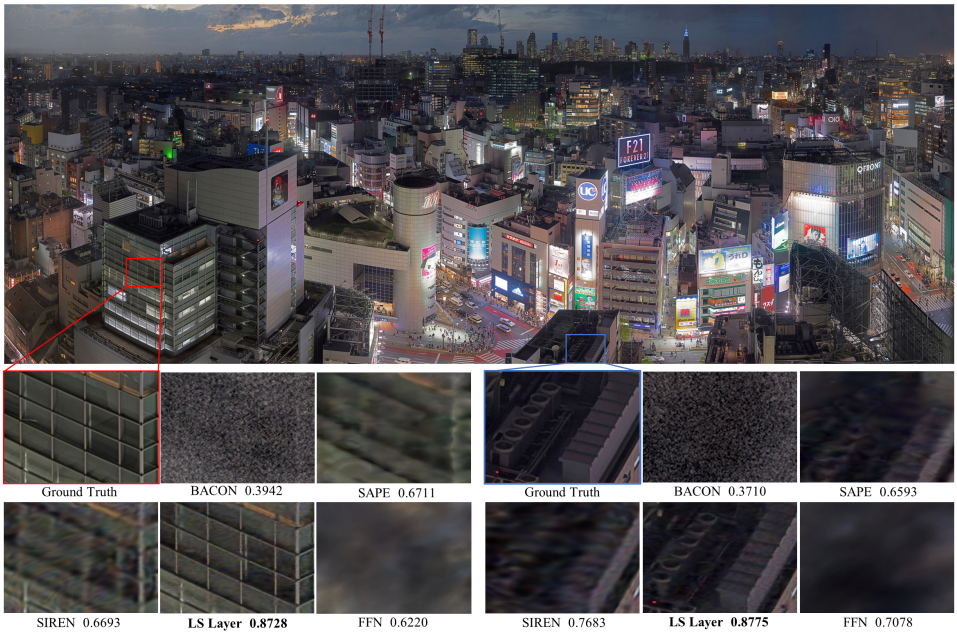

Figure 4: Qualitative comparison of 2D image construction on "Tokyo" from [28]. In the patches shown, our method yields more distinguishable boundaries than prior encoding methods.

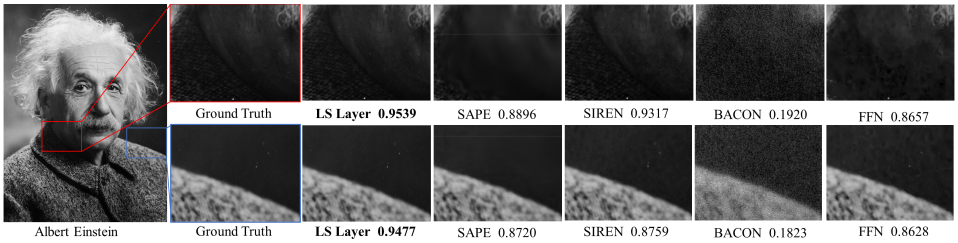

Figure 5: Qualitative comparison of 2D image construction on "Einstein" [32]. In the first row, we show that our method produces sharper boundaries. In the second row, we show that our method's reconstructed background is less noisy than the baselines. Zoom in for a detailed evaluation.

5.3 3D Shape Reconstruction

In addition to 2D images, we evaluate our method's ability to reconstruct 3D shapes on which prior work [44, 47, 46, 32, 54, 30] excel at. The task is to train our network to predict an SDF value given the 3D coordinates of a queried point as input. We use the following networks as baselines: 1) Frequency-based methods including SAPE [17], SIREN [44], FFN [47]; 2) InstantNGP [32] as a parametric method. We train our network and the baselines to overfit on shape "Beard Man" from the Stanford 3D Scanning Repository [2] with plenty of fine details, then pass the fitted SDF to the Marching Cube Algorithm [27] to extract meshes at the 1024 resolution. We use Chamfer Distance, the Metro Criteria [9], F-Score, and IoU for a comprehensive comparison of our network with the baselines. For more results on other shapes, please refer to the Supplemental.

Table 1 shows that by quantitative measures our network surpasses most of the baselines, especially those based-on frequency-based encoding schemes [44, 24, 17, 47]. Only BACON achieves slightly better F-Score than ours. Qualitatively, we see from Figure 6 our method is able to fit boundaries at fine scales better than the selected baselines.

5.4 Ablation Study

Further, we investigate the impacts of our design choices on the representation power of our neural implicit pipeline. Without loss of generality, we constrain our discussion to the 2D image synthesis task and use the same dataset from Sec. 5.2.

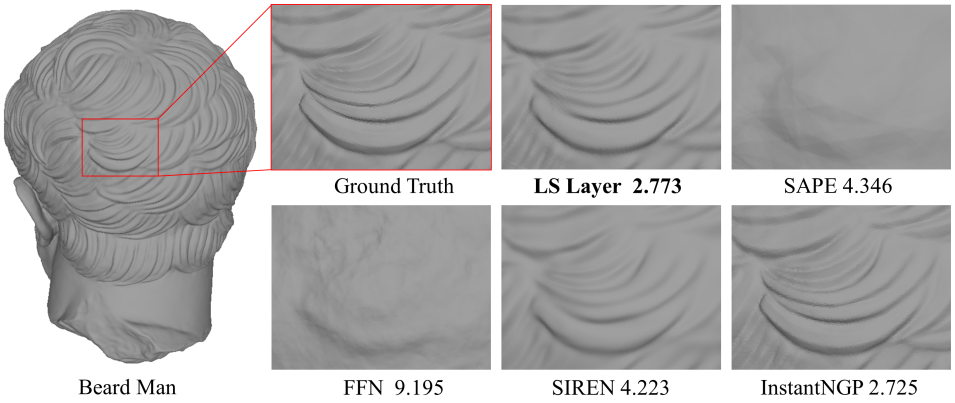

Figure 6: Qualitative comparison of 3D shape reconstruction on "Beard Man"[2]. Similar to the 2D image reconstruction, our method has better ability to fit geometric details (hair). Chamfer distances scaled by 10^3 for easier comparison.

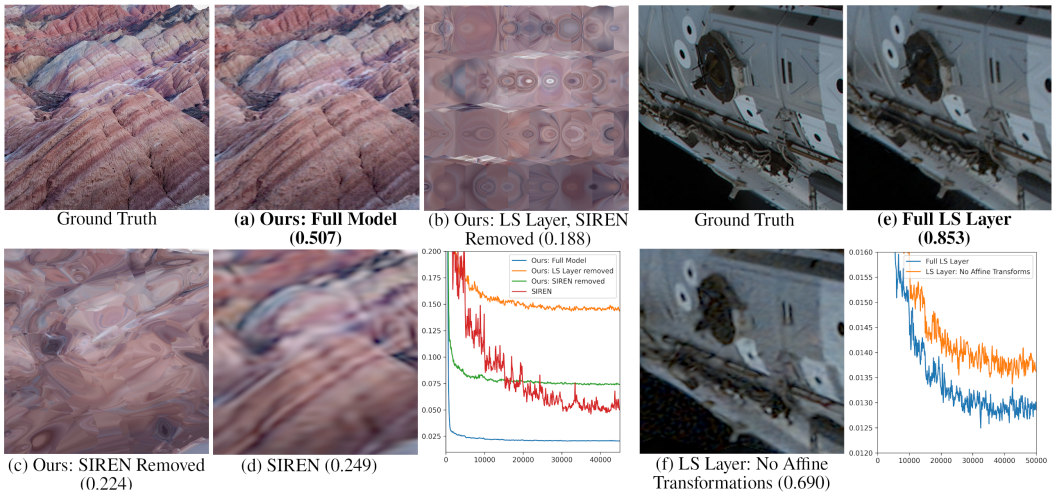

Figure 7: Ablation test: the impact of LS Layer, the SIREN backbone and affine transformations. We report SSIM on each patch.

(A) Effectiveness of the LS Layer. To demonstrate our encoding scheme can indeed boosts reconstruction performance significantly, we train our proposed pipeline on "Zhangye" [1] with the same loss function from SIREN [44] under four configurations, as shown on the left in Figure 7: (a) Having the full model as illustrated in Figure 2; (b) The LS Layer with all ellipsoids anchored to the grid, orientation, and scale set to fixed unlearnable parameters. The SIREN backbone (i.e., the Sine Branch) is also ablated, leaving only the ReLU Branch; (c) The SIREN backbone ablated, leaving the The LS Layer and the ReLU branch; (d) The LS Layer and the ReLU Branch ablated, leaving the SIREN backbone only. We see that our full model (a) exceeds SIREN (d) by a large margin (SSIM: 0.507 vs. 0.249), and our loss converges faster and to a lower value. However, one may still argue that the ReLU backbone, not the LS Layer, bolsters our representation power. We show this is not the case: comparing (b) with (c), performance drops from 0.224 to 0.188 after we ablate affine transformations within the LS Layer. Visually inspecting the outcomes from the two configurations, we see that without the LS Layer the network fails to synthesize both coarse and fine geometric structures. In fact, having ellipsoids anchored to the grid leads to the grid-like pattern in (b). On the other hand, we acknowledge the crucial role of the SIREN backbone: comparing (a) with (c), performance drops significantly from 0.507 to 0.224 after we remove the Sine Branch, and the stripes become more blurry. So sinusoidal activation indeed allows our full network to capture complex structures that otherwise would fail.

(B) Impact of affine transformations within the LS Layer. We then investigate if applying learnable affine transformations to the shape primitives contributes much to our method's representation power. Under the entire model, we train and evaluate on "Canadarm" [1] under two configurations: (e) all transformation parameters (position, rotation, scale) set to be learnable; (f) all affine transformation parameters fixed, so the Gaussian ellipsoids become unit spheres anchored uniformly on a grid. From the right of Figure 7, we see that SSIM drops drastically from 0.853 to 0.690, and the loss converges slower and to a larger value after we make transformation parameters unlearnable. Qualitatively the image looks more blurry with less detail present.

(C) Number of shape primitives. Next, we vary the number of learnable Gaussian ellipsoids m in our LS Layer over the range of 1.3K to 25K and train on "Sydney" [1] to demonstrate that our final pick of this hyperparameter (25K) has fully exploited the representation capacity of our network. Figure 8 shows that using more ellipsoids to represent the image not surprisingly leads to faster loss convergence, as well as better reconstruction results, both by metrics (PSNR / SSIM) and by qualitative measurement: object boundaries are more distinguishable and fewer artifacts are present in smooth regions (the sky). While the trends shown in the metric bar plots suggest that with more primitives, we can achieve even better results, we believe our pick strikes a balance between computational cost and reconstruction performance — learning more ellipsoids would incur more training time and memory footprint but diminishing pay-off in visual quality.

Figure 8: Ablation test: varying the number of shape primitives. We report PSNR and SSIM on each patch.

(D) Number of nearest neighbors.

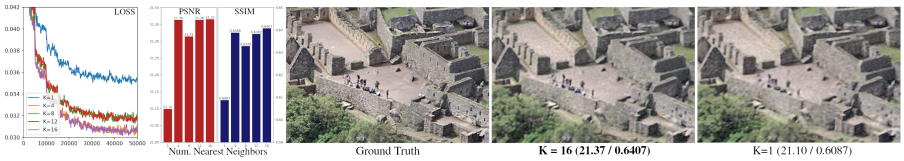

Figure 9: Ablation test: varying the number of nearest neighbors to be searched.

At last, we vary the number of nearest neighboring ellipsoids K over the range of 1 to 16 to evaluate the extent to which our network can exploit locality with KNN search. We train and evaluate our pipeline on "Machu Picchu" [1] and showcase our reconstruction results in Figure 9. We notice that the network's performance is not very sensitive to our choice of K. This observation suggests that the shape primitives learned in the the LS Layer can effectively capture local information so much so that a small number of primitives would be enough to encapsulate the relation between a queried point and the target surface.

6 Conclusions and Limitations

We present a new learning pipeline for neural implicit functions. Specifically, we formulate a novel local smoothness layer to achieve spatially adaptive frequency transformation. By approximating the implicit functions with a group of parsimonious primitives, we can estimate the relationships between each query point and the signal space, which are used to extract primitive weights for further feature smoothing. The resulting frequency adaptation enables the network to concentrate on regions with rich high-frequency details and reduce unwanted artifacts in the flat regions, significantly advancing implicit signal representational capability. We evaluate our network design under several tasks, consistently revealing empirical improvements.

We also recognize that our method is far from perfect. For example, several prior works extensively explored multiscale feature encoding and synthesis, but in our tasks, we have yet to present convincing evidence that we are leveraging multiscale features effectively.

7 Broader Impact Statement

Our proposed method can be broadly applied to a multitude of computer vision tasks, e.g., audio signal synthesis, high-resolution image synthesis, and 3D shape reconstruction. We envision that the method can be put into good use in the far future on downstream tasks, such as shape generation for AR/VR devices or medical image analysis, bringing great benefit to humankind. Nevertheless, we recognize it is not invulnerable to exploitation: potential malicious uses include generating fake audio or images for misinformation. Nevertheless, they can be alleviated by existing methods.

References

[1] Flickr creative commons. https://www.flickr.com/creativecommons/.

[2] Stanford 3d scanning repository. http://graphics.stanford.edu/data/3Dscanrep//.

[3] J. T. Barron, B. Mildenhall, M. Tancik, P. Hedman, R. Martin-Brualla, and P. P. Srinivasan. Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. In *ICCV*, 2021.

[4] I. Bello, B. Zoph, A. Vaswani, J. Shlens, and Q. V. Le. Attention augmented convolutional networks. In *ICCV*, 2019.

[5] R. Chabra, J. E. Lenssen, E. Ilg, T. Schmidt, J. Straub, S. Lovegrove, and R. Newcombe. Deep local shapes: Learning local sdf priors for detailed 3d reconstruction. In *ECCV*.

[6] E. R. Chan, M. Monteiro, P. Kellnhofer, J. Wu, and G. Wetzstein. pi-gan: Periodic implicit generative adversarial networks for 3d-aware image synthesis. In *CVPR*.

[7] Y. Chen, S. Liu, and X. Wang. Learning continuous image representation with local implicit image function. In *CVPR*.

[8] Z. Chen, A. Tagliasacchi, and H. Zhang. Bsp-net: Generating compact meshes via binary space partitioning. In *CVPR*, 2020.

[9] P. Cignoni, C. Rocchini, and R. Scopigno. Metro: measuring error on simplified surfaces. In *Computer graphics forum*, 1998.

[10] B. Deng, K. Genova, S. Yazdani, S. Bouaziz, G. Hinton, and A. Tagliasacchi. Cvxnet: Learnable convex decomposition. In *CVPR*, 2020.

[11] S. Fleishman, I. Drori, and D. Cohen-Or. Bilateral mesh denoising. In *SIGGRAPH*. 2003.

[12] M. Gadelha, G. Gori, D. Ceylan, R. Mech, N. Carr, T. Boubekeur, R. Wang, and S. Maji. Learning generative models of shape handles. In *CVPR*, 2020.

[13] K. Genova, F. Cole, D. Vlasic, A. Sarna, W. T. Freeman, and T. Funkhouser. Learning shape templates with structured implicit functions. *ICCV*, 2019.

[14] K. Genova, F. Cole, A. Sud, A. Sarna, and T. Funkhouser. Local deep implicit functions for 3d shape. In *CVPR*, 2020.

[15] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *AISTATS*, 2010.

[16] R. Hanocka, A. Hertz, N. Fish, R. Giryes, S. Fleishman, and D. Cohen-Or. Meshcnn: a network with an edge. *ACM ToG*, 2019.

[17] A. Hertz, O. Perel, R. Giryes, O. Sorkine-Hornung, and D. Cohen-Or. Sape: Spatially-adaptive progressive encoding for neural optimization. *NeurIPS*, 2021.

[18] C. Jiang, A. Sud, A. Makadia, J. Huang, M. Nießner, T. Funkhouser, et al. Local implicit grid representations for 3d scenes. In *CVPR*, 2020.

[19] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *CoRR*, 2014.

[20] A. Levin, D. Lischinski, and Y. Weiss. Colorization using optimization. In *ACM SIGGRAPH*. 2004.

[21] A. Levin, D. Lischinski, and Y. Weiss. A closed-form solution to natural image matting. *IEEE TPAMI*, 2007.

[22] J. Li, K. Xu, S. Chaudhuri, E. Yumer, H. Zhang, and L. Guibas. Grass: Generative recursive autoencoders for shape structures. *ACM TOG*, 2017.

[23] Y. Li, R. Bu, M. Sun, W. Wu, X. Di, and B. Chen. Pointcnn: Convolution on x-transformed points. *NeurIPS*, 2018.

[24] D. B. Lindell, D. Van Veen, J. J. Park, and G. Wetzstein. Bacon: Band-limited coordinate networks for multiscale scene representation. *CVPR*, 2022.

[25] L. Liu, J. Gu, K. Zaw Lin, T.-S. Chua, and C. Theobalt. Neural sparse voxel fields. *NeurIPS*, 2020.

[26] R. Liu, J. Lehman, P. Molino, F. Petroski Such, E. Frank, A. Sergeev, and J. Yosinski. An intriguing failing of convolutional neural networks and the coordconv solution. *NeurIPS*, 2018.

[27] W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. *ACM SIGGRAPH*, 1987.

[28] J. N. Martel, D. B. Lindell, C. Z. Lin, E. R. Chan, M. Monteiro, and G. Wetzstein. Acorn: Adaptive coordinate networks for neural representation. *SIGGRAPH*, 2021.

[29] I. Mehta, M. Gharbi, C. Barnes, E. Shechtman, R. Ramamoorthi, and M. Chandraker. Modulated periodic activations for generalizable local functional representations. In *ICCV*, 2021.

[30] L. Mescheder, M. Oechsle, M. Niemeyer, S. Nowozin, and A. Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *CVPR*, 2019.

[31] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020.

[32] T. Müller, A. Evans, C. Schied, and A. Keller. Instant neural graphics primitives with a multiresolution hash encoding. *CoRR*, 2022.

[33] V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In *ICML*, 2010.

[34] C. Niu, J. Li, and K. Xu. Im2struct: Recovering 3d shape structure from a single rgb image. In *CVPR*, 2018.

[35] A. Noguchi, X. Sun, S. Lin, and T. Harada. Neural articulated radiance field. In *ICCV*.

[36] J. J. Park, P. Florence, J. Straub, R. Newcombe, and S. Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *CVPR*, 2019.

[37] D. Paschalidou, A. O. Ulusoy, and A. Geiger. Superquadrics revisited: Learning 3d shape parsing beyond cuboids. In *CVPR*, 2019.

[38] D. Paschalidou, L. V. Gool, and A. Geiger. Learning unsupervised hierarchical part decomposition of 3d objects from a single rgb image. In *CVPR*, 2020.

[39] T. Peleg, P. Szekely, D. Sabo, and O. Sendik. Im-net for high resolution video frame interpolation. In *CVPR*.

[40] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *CVPR*, 2017.

[41] C. R. Qi, L. Yi, H. Su, and L. J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *NeurIPS*, 2017.

[42] H. Rhodin, N. Robertini, C. Richardt, H.-P. Seidel, and C. Theobalt. A versatile scene model with differentiable visibility applied to generative pose estimation. In *ICCV*, 2015.

[43] G. Sharma, R. Goyal, D. Liu, E. Kalogerakis, and S. Maji. Csgnet: Neural shape parser for constructive solid geometry. In *CVPR*, 2018.

[44] V. Sitzmann, J. Martel, A. Bergman, D. Lindell, and G. Wetzstein. Implicit neural representations with periodic activation functions. *NeurIPS*, 2020.

[45] S. Sridhar, H. Rhodin, H.-P. Seidel, A. Oulasvirta, and C. Theobalt. Real-time hand tracking using a sum of anisotropic gaussians model. In *3DV*, 2014.

[46] T. Takikawa, J. Litalien, K. Yin, K. Kreis, C. Loop, D. Nowrouzezahrai, A. Jacobson, M. McGuire, and S. Fidler. Neural geometric level of detail: Real-time rendering with implicit 3d shapes. In *CVPR*, 2021.

[47] M. Tancik, P. Srinivasan, B. Mildenhall, S. Fridovich-Keil, N. Raghavan, U. Singhal, R. Ramamoorthi, J. Barron, and R. Ng. Fourier features let networks learn high frequency functions in low dimensional domains. *NeurIPS*, 2020.

[48] C. Tomasi and R. Manduchi. Bilateral filtering for gray and color images. In *ICCV*, 1998.

[49] E. Tretschk, A. Tewari, V. Golyanik, M. Zollhöfer, C. Stoll, and C. Theobalt. Patchnets: Patch-based generalizable deep implicit 3d shape representations. In *ECCV*, 2020.

[50] S. Tulsiani, H. Su, L. J. Guibas, A. A. Efros, and J. Malik. Learning shape abstractions by assembling volumetric primitives. In *CVPR*, 2017.

[51] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *NeurIPS*, 2017.

[52] J. Wu, C. Zhang, T. Xue, B. Freeman, and J. Tenenbaum. Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling. *NeurIPS*, 2016.

[53] G. Yang, S. Belongie, B. Hariharan, and V. Koltun. Geometry processing with neural fields. *NeurIPS*, 2021.

[54] W. Yifan, L. Rahmann, and O. Sorkine-Hornung. Geometry-consistent neural shape representation with implicit displacement fields. *ICLR*, 2022.

[55] C. Zou, E. Yumer, J. Yang, D. Ceylan, and D. Hoiem. 3d-prnn: Generating shape primitives with recurrent neural networks. In *ICCV*, 2017.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] See Abstract and Introduction.
   (b) Did you describe the limitations of your work? [Yes] See Section 6
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See Section 7
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [No] This paper does not contain theoretical results.
   (b) Did you include complete proofs of all theoretical results? [No] This paper does not contain theoretical results.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [No] The code, instructions, and the pre-trained models will be released upon acceptance. The data used is already publicly available.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Section 5.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] It will be in the Appendix.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Section 5.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...

(a) If your work uses existing assets, did you cite the creators? [Yes]

(b) Did you mention the license of the assets? [No] The data we used are common across research papers and the licences allow research purposes.

(c) Did you include any new assets either in the supplemental material or as a URL? [Yes] The new data that we used is cited as URLs.

(d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [No] The old data is commonly used across research papers and new data on public website is allowed to use as research purposes.

(e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [No] The old data is commonly used across research papers and new data on public website is allowed to use as research purposes.

5. If you used crowdsourcing or conducted research with human subjects...

(a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]

(b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]