BEYOND INSTANT-NGP: A UNIFIED VARIABLE-ORDER B-SPLINE FRAMEWORK FOR SPARSE-VIEW CT RECONSTRUCTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Achieving high-quality CT reconstruction from sparse measurements is essential for reducing radiation exposure without compromising diagnostic image quality. However, the limited number of projections makes image reconstruction severely underdetermined. In this context, Instant-NGP has emerged as an efficient implicit neural representation for sparse-view CT reconstruction, yet its representational properties underlying its effectiveness remain poorly understood. In this work, we provide a systematic mathematical analysis of the original methodology and specific implementation of Instant-NGP, and show that its multi-resolution grid encoding is equivalent to a multi-resolution linear B-spline representation, in which the CT image is approximated by multivariate polynomials constructed from locally supported linear B-spline basis functions. We further show that the approximation capability of this linear B-spline representation is fundamentally limited for piecewise-smooth signals, potentially leading to reconstruction errors and artifacts. Motivated by this observation, we generalize the linear representation to a unified variable-order B-spline framework, allowing higher-order basis functions to provide more accurate local approximation while retaining the efficiency of hash encoding. We theoretically establish that, under appropriate smoothness conditions, increasing the spline order tightens the bound on the optimal approximation error. Extensive quantitative and qualitative experiments on sparse-view CT reconstruction validate our analysis, showing that higher-order B-splines consistently improve reconstruction quality over the linear formulation, with quartic B-splines achieving the strongest overall performance. The code will be open-sourced upon acceptance of the paper.

1 INTRODUCTION

Computed Tomography (CT) is a widely used non-invasive imaging technique for visualizing internal anatomical structures. Reducing the number of X-ray projections can lower radiation exposure and shorten acquisition time, which is particularly desirable in clinical practice. However, fewer projections lead to a severely underdetermined inverse problem, making the development of sparse-view CT reconstruction algorithms an important and active research focus.

Recent advances in implicit neural representations (INRs) provide a promising self-supervised paradigm for sparse-view CT reconstruction. Typically, INRs encode coordinates into latent feature vectors and map them to the target CT attenuation by a neural network (NN). The training is driven by the consistency between the projection data and the attenuation field predicted by the NN.

As one of the most influential INR approaches, Instant-NGP (Müller et al. (2022), hereafter abbreviated as NGP) incorporates hash-based indexing with multi-resolution grid interpolation into the encoding stage, which substantially improves training efficiency and representation accuracy. It has since been successfully applied to CT reconstruction. Nevertheless, under severe view undersampling, reconstruction results can still exhibit noticeable artifacts. In addition, due to its heuristic nature, the mathematical principles underlying NGP remain insufficiently understood, limiting deeper insights into its reconstruction behavior and artifacts generation, thereby hindering further improvements and developments in medical imaging.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

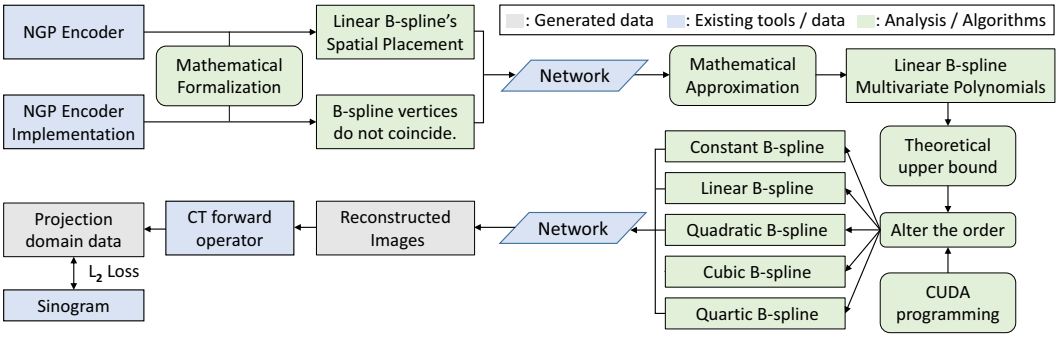


Figure 1: Systematic workflow of theoretical analysis and practical CT reconstruction algorithms.

To address these issues, we connect NGP with classical approximation theory and signal processing, and develop a unified mathematical formulation. Specifically, we establish a rigorous, step-by-step correspondence between NGP and the spatial arrangement of linear B-splines. To better reflect the actual implementation beyond the original paper, we characterize how specific implementation choices induce distinct spatial arrangements of B-splines. Moreover, by approximating the neural network activation functions via truncated Taylor expansions, we show that the network output can be expressed as a multivariate polynomial of B-spline basis functions.

Motivated by the piecewise-smooth nature of anatomical tissue and the observation that the approximation capability of locally supported B-splines is constrained by their order, we employ B-splines with progressively increasing orders as the encoder to improve CT reconstruction accuracy. To further uncover the performance boundaries imposed by the B-spline order on CT reconstruction, we construct five B-spline variants with different orders: constant, linear, quadratic, cubic, and quartic, and replace the encoding module in NGP with them. Our established approximation bound shows that, under appropriate smoothness conditions, increasing the spline order reduces the bound of optimal approximation error. The experimental results are consistent with the above-mentioned analysis: (i) replacing the linear B-spline with any higher-order variant improves reconstruction accuracy. (ii) across pairwise comparisons among the five orders, increasing the B-spline order is generally associated with improved reconstruction performance. Moreover, the variable-order B-splines can be integrated with hash encoding, enabling efficient reconstruction while preserving the representation capacity. For computational efficiency, we implement all five variants in CUDA. The systematic workflow is shown in Fig. 1. In summary, our main contributions are as follows:

❑ (1) We establish a mathematical formulation for NGP by linking its encoder to the building and placement of linear B-splines. We further analyze how implementation choices affect the B-spline placements and show that the NN output can be approximately interpreted as a multivariate polynomial composed of B-splines.

❑ (2) We generalize this formulation into a unified variable-order B-spline framework and derive an order-dependent approximation bound. We further implement five B-spline variants—constant, linear, quadratic, cubic, and quartic—for sparse-view CT reconstruction, demonstrating that reconstruction accuracy generally improves with increasing B-spline order.

❑ (3) We employ hash encoding to accelerate the five different variants and optimize their implementation with CUDA. Beyond achieving speedups, experiments demonstrate that three variants with higher-than-linear order outperform both traditional and self-supervised CT reconstruction methods, with the quartic variant consistently achieving the best performance.

2 Related Work

Classic CT reconstruction. Classic CT reconstruction techniques can be broadly categorized into analytical and iterative approaches. Analytical methods, such as FBP (Herman, 2009) and Spline-based FBP (Horbelt et al., 2002), are computationally efficient but offer limited reconstruction quality. Iterative methods, including ART (Gordon et al., 1970), SART (Andersen & Kak, 1984), SIRT

(Gregor & Benson, 2008) and ASD-POCS (Sidky & Pan, 2008) achieve better accuracy but suffer from slow convergence and inadequate exploitation of large-scale data prior.

Implicit neural representations (INRs). Existing INR research mainly focuses on encoder design and property analysis (Tancik et al., 2020; Müller et al., 2022; Yüce et al., 2022; Dai & Fan, 2026), as well as neural network design (Sitzmann et al., 2020; Shah & Sitawarin, 2024; Alsakabi et al., 2025). As a follow-up application of NGP, Barron et al. (2023) adopt a multisampling-like strategy for rendering, while a characterization of NGP is beyond the scope of their study. Feng et al. (2025) use cubic B-splines for surface reconstruction from point clouds, with the focus on the point clouds scenario rather than on formulating NGP or analyzing performance across different spline orders.

INRs based CT reconstruction. INRs have shown promising performance in CT reconstruction (Rückert et al., 2022; Shen et al., 2022; Saragadam et al., 2023; Xie et al., 2025; Wu et al., 2026; Su et al., 2026; Xie & Blaschko, 2026). Following the success of NGP, several subsequent studies tailor it to CT reconstruction (Zha et al., 2022; Wu et al., 2023; Shao et al., 2024; Cai et al., 2024; Zhou et al., 2025; Xu et al., 2025; Li et al., 2025; Park & Jeon, 2026). Despite these advances, a deeper theoretical understanding of NGP's strengths and limitations in CT reconstruction remains underexplored, which hinders its extension to B-splines of varying orders.

3 Analysis and Framework

3.1 Preliminaries

Without loss of generality, we consider the two dimensions case. For any query coordinate point $\mathbf{x} \in \mathbb{R}^2$ in the target object, the feature at level l produced by the NGP encoding module is computed via linear interpolation:

$$\begin{aligned} y_l(\mathbf{x}) = (1-\{\mathbf{x}N_l\}[0])(1-\{\mathbf{x}N_l\}[1])f_{l,(0,0)} + \{\mathbf{x}N_l\}[0](1-\{\mathbf{x}N_l\}[1])f_{l,(1,0)} \\ + (1-\{\mathbf{x}N_l\}[0])\{\mathbf{x}N_l\}[1]f_{l,(0,1)} + \{\mathbf{x}N_l\}[0]\{\mathbf{x}N_l\}[1]f_{l,(1,1)}. \end{aligned} \tag{1}$$

Please refer to sec. A.2 for definitions and details. NGP's multilinear interpolation can be equivalently expressed as a tensor product of 1-D linear B-spline functions, as derived below.

3.2 Equivalence of encoder and linear B-spline

According to the derivation in Appendix sec. A.4. The value at $\mathbf{x}$ is determined by four 2D linear B-spline centered at the four vertices of the cell containing $\mathbf{x}$. Due to the compact support of linear B-splines, the level-l encoding output of $\mathbf{x}$ can be expressed as $y_l(\mathbf{x})$, as defined in eq. (2):

$$\begin{aligned} y_l(\mathbf{x}) = & B_2(\mathbf{x}N_l; \lfloor \mathbf{x}N_l \rfloor + (0,0))\, \mathbf{T}_l[h_l(\lfloor \mathbf{x}N_l \rfloor + (0,0))] \\ & + B_2(\mathbf{x}N_l; \lfloor \mathbf{x}N_l \rfloor + (1,0))\, \mathbf{T}_l[h_l(\lfloor \mathbf{x}N_l \rfloor + (1,0))] \\ & + B_2(\mathbf{x}N_l; \lfloor \mathbf{x}N_l \rfloor + (0,1))\, \mathbf{T}_l[h_l(\lfloor \mathbf{x}N_l \rfloor + (0,1))] \\ & + B_2(\mathbf{x}N_l; \lfloor \mathbf{x}N_l \rfloor + (1,1))\, \mathbf{T}_l[h_l(\lfloor \mathbf{x}N_l \rfloor + (1,1))] \\ & + \sum_{p \in \{0,\dots,N_l\}^2 \setminus \mathcal{N}_n} B_2(\mathbf{x}N_l; p)\, \mathbf{T}_l[h_l(p)], \end{aligned} \tag{2}$$

where $\mathcal{N}_n := \{\lfloor \mathbf{x}N_l \rfloor + (0,0), \lfloor \mathbf{x}N_l \rfloor + (1,0), \lfloor \mathbf{x}N_l \rfloor + (0,1), \lfloor \mathbf{x}N_l \rfloor + (1,1)\}$. The symbol definitions and step-by-step proof are provided in sec. A.4. Thus, for any $\mathbf{x}$ and level l, the relationship between the value at $\mathbf{x}$ and the B-splines centered at all cell vertices follows the form in eq. (3):

$$y_l(\mathbf{x}) = \sum_{p \in \{0,\dots,N_l\}^2} B_2\big(xN_l; p\big)\, \mathbf{T}_l[h_l(p)]\,. \tag{3}$$

And the overall encoding result y in NGP can be expressed as $y := [y_1, y_2, ..., y_L]^\top$.

3.3 Multi-resolution analysis and wavelets

Using the translation invariance property from eq. (24) in Appendix sec. A.3 , eq. (3) can be equivalently written as:

$$y_l(\mathbf{x}) = \sum_{p \in \{0,\dots,N_l\}^2} B_2\big(xN_l - p; 0\big)\, \mathbf{T}_l[h_l(p)]\,. \tag{4}$$

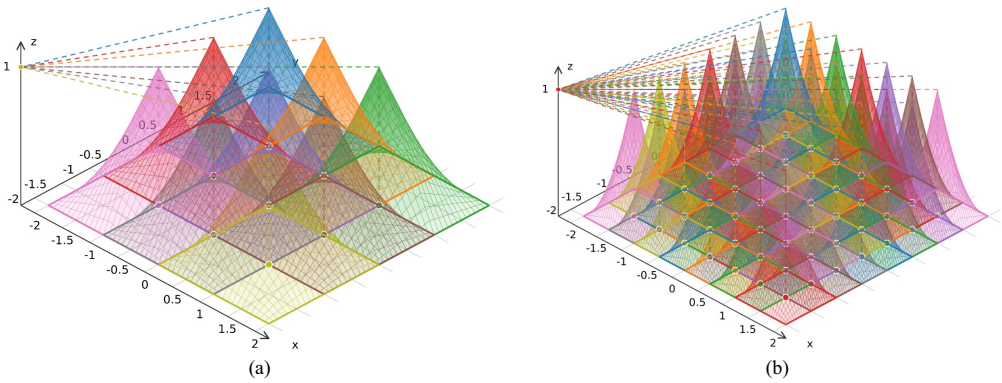


Figure 2: (a): Spatial arrangement and geometric shapes of two-dimensional linear B-splines. (b): Two-dimensional linear B-splines on a denser grid through scaling and translation.

For NGP with an exponential growth factor of 2, choosing $N_{\min}$ to be a power of 2 ensures that every N_l remains a power of 2 ($N_l{=}2^{k_l}$). In this case, eq. (4) can be written precisely in the standard form of multi-resolution analysis:

$$y_l(\mathbf{x}) = \sum_{p\in\{0,\dots,2^{k_l}\}^2} B_2\left(2^{k_l}(x-\frac{p}{2^{k_l}});0\right)\,\mathbf{T}_l[h_l(p)]\,. \tag{5}$$

Geometrically, this can be viewed as a two-step transformation. First, the standard B-spline centered at $(0,0)$ is shrunk by a factor of $\frac{1}{2^{k_l}}$ while its center remains fixed at the origin; it is then shifted from the origin by the displacement vector $\frac{p}{2^{k_l}}$. Consequently, when $2^{k_i} > 2^{k_j}$, the B-splines at level i undergo stronger shrinkage than those at level j and are distributed more densely. The placement of B-splines at different scales can be visualized by comparing panels (a) and (b) in Fig. 2. Building on above analysis, we define the closure of a linearly expanding space V_{k_l}:

$$V_{k_l} := \overline{\operatorname{span}\left\{B_2\left(2^{k_l}\left(x-\frac{p}{2^{k_l}}\right);0\right)\,\middle|\,p\in\mathbb{Z}^2\right\}} \tag{6}$$

The space V_{k_l} satisfies: $V_0 \subset V_1 \subset V_2 \subset \cdots$ and the wavelet space W_i is the orthogonal complement of V_i in V_{i+1}: $V_{i+1} = V_i \oplus W_i$ (Chui & Wang, 1991). Controlling the contributions from the wavelet space can enable structure preservation and artifact suppression (Belge et al., 2000). We thus establish a natural connection between the NGP's encoder and wavelet theory.

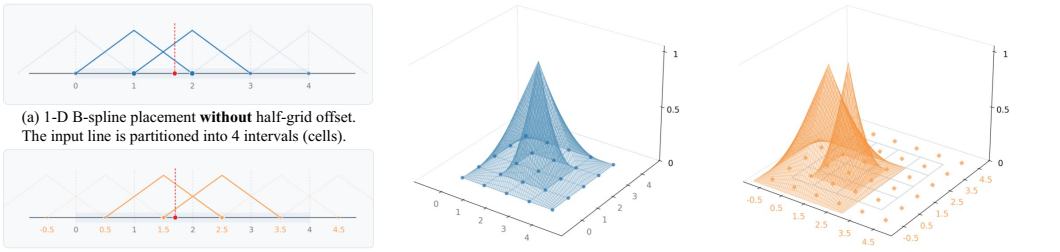
3.4 Locality of B-splines and Hash Collisions

From the result of the above derivation and perspective of function approximation theory, the encoding part of NGP can be viewed as learning a linear combination of fixed linear B-splines to approximate an underlying unknown function. Owing to their compact spatial support, these localized basis functions are suited to CT images dominated by spatially localized structures. This contrasts with the globally supported trigonometric bases used in random Fourier-encoded INRs (Tancik et al., 2020). This advantage is evident under severe view undersampling in Fig. 4, where Fourier encoding produces global wave-like artifacts, while linear B-splines (NGP) do not.

In the representation shown in eq. (3), hash collisions of NGP essentially force distinct B-spline basis functions to share the same coefficient. At a given resolution level, multiple spatially localized B-splines will be merged into a single basis function through coefficient sharing. As a result, their supports may span long distances across the image. Such long-range coupling conflicts with the local Markov property commonly assumed for medical images, thereby potentially degrading reconstruction quality when hash collisions are severe. This effect can be formalized as follows:

$$y_l(\mathbf{x}) = \sum_{j\in\{0,1,\dots,T-1\}} \left(\sum_{p:\,h_l(p)=j} B_2\big(N_l\mathbf{x};p\big)\right)\mathbf{T}_l[j]. \tag{7}$$

If there exists a j such that $|\{p|h_l(p)=j\}|>1$, a nonlocal basis function $\sum_{p:\,h_l(p)=j} B_2\big(N_l\mathbf{x};p\big)$ is formed. Conversely, no nonlocal basis function is formed if and only if the hash function h_l



(a) 1-D B-spline placement **without** half-grid offset. The input line is partitioned into 4 intervals (cells).

(b) 1-D B-spline placement **with** half-grid offset. The input line is partitioned into 4 intervals (cells).

(c) 2-D B-spline placement **without** half-grid offset. The input plane is partitioned into 4×4 grids (cells).

(d) 2-D B-spline placement **with** half-grid offset. The input plane is partitioned into 4×4 grids (cells).

Figure 3: Visualization of the offset effect, which prevents multi-scale B-splines from sharing peaks.

is injective in $\{0, \dots, N_l\}^2$ and the $h_l(\{0, \dots, N_l\}^2) \subseteq \{0, \dots, T-1\}$, where T denotes the number of rows in the hash table $\mathbf{T}_l$. However, decreasing the T does not necessarily result in a monotonic degradation in performance, even though it increases the number of hash collisions. This is because forcing multiple basis functions to share the same coefficient can also induce an implicit regularization. This phenomenon is empirically observed in panel (a) of Fig. 6 and table 1.

3.5 The Offset in NGP

Unlike the coordinate mapping described in the original NGP paper, where the query point P with coordinates $\mathbf{x}$ is mapped to $\mathbf{x}N_l$, the official implementation of NGP introduces a half-grid-cell offset, $\frac{\mathbf{1}_2}{2}$, by mapping the coordinates $\mathbf{x}$ to $\mathbf{x}N_l + \frac{\mathbf{1}_2}{2}$, where $\mathbf{1}_2 := (1,1)^\top$. A geometric explanation can be given based on our derived spatial geometric position and affine transformation of B-splines. When using $xN_l + \frac{\mathbf{1}_2}{2}$, $y_l(x)$ can be written as in eq. (8):

$$y_l(\mathbf{x}) = \sum_{p \in \{0,\dots,N_l+1\}^2} B_2\big(xN_l + \frac{\mathbf{1}_2}{2} - p; 0\big)\, \mathbf{T}_l[h_l(p)]\,. \tag{8}$$

For the commonly used case $N_l = 2^{k_l}$, the eq. (8) can be rewritten as eq. (9):

$$y_l(\mathbf{x}) = \sum_{p \in \{0,\dots,2^{k_l}+1\}^2} B_2\big(2^{k_l}(x + \frac{\mathbf{1}_2}{2} \cdot \frac{1}{2^{k_l}} - \frac{p}{2^{k_l}}); 0\big)\, \mathbf{T}_l[h_l(p)]\,. \tag{9}$$

This is equivalent to shifting the shrunken B-spline basis functions by half a grid interval, $1/2^{k_l+1}$, in the negative direction, followed by translations in the positive direction with spacing $p/2^{k_l}$. The corresponding one- and two-dimensional configurations are illustrated in Fig. 3.

A consequence of this half-grid-cell offset is that the peak locations of the B-spline basis functions at different resolution levels no longer coincide. Its mathematical proof is in Appendix sec. A.8.

3.6 Neural Network Output

Assuming the neural network in NGP has D hidden layers and M_d neurons in d^{th} layer ($d = 1, \dots, D+1$). Define the output of 0^{th} layer as $\mathbf{h}^{(0)} := y$. Let $\mathbf{h}^{(d)}$ denote the d-th layer's output, and the relationship between adjacent layers is given by $\mathbf{h}^{(d)} = \sigma(W^{(d)}\mathbf{h}^{(d-1)} + \mathbf{b}^{(d)})$.

Suppose that the activation function is approximated by its truncated Taylor expansion of order K, i.e., $\sigma(z) \approx \sum_{r=0}^{K} a_r z^r$. In fact, trigonometric activations can be approximated by Taylor expansions, while ReLU can be approximated using Chebyshev polynomials (Mehmeti-Göpel et al., 2020; Yüce et al., 2022). Then the i^{th} neuron in d^{th} layer's output becomes:

$$h_i^{(d)} = \sum_{r=0}^{K} a_r \left(\sum_{j=1}^{M_{d-1}} W_{ij}^{(d)} h_j^{(d-1)} + b_i^{(d)} \right)^r, \tag{10}$$

Using the expression of y in sec. 3.2, after iterative expansion, $\mathbf{h}^{(D+1)}$ can be written as:

$$\mathbf{h}^{(D+1)}(\mathbf{x}) = \sum_{\boldsymbol{\alpha}\in\mathcal{A}} C_{\boldsymbol{\alpha}} \prod_{l=1}^{L} \prod_{p\in\mathcal{P}_l} \left[B_2(N_l\mathbf{x}; p)\right]^{\alpha_{l,p}}, \tag{11}$$

where $\mathcal{A} := \left\{ \boldsymbol{\alpha} = (\alpha_{l,p})_{\substack{1\le l\le L \\ p\in\mathcal{P}_l}} \;\middle|\; \alpha_{l,p}\in\mathbb{N}_0,\ \sum_{l=1}^{L}\sum_{p\in\mathcal{P}_l}\alpha_{l,p} \le K^{D+1} \right\}, \mathcal{P}_l := \{0,\dots,N_l\}^2$. The above is merely an algebraic expression. When evaluating eq. (11) for any specific input x, we adopt the convention $0^0 = 1$. Consequently, the output of the neural network can be approximated as a multivariate polynomial surrogate whose variables are distinct linear B-spline. The network parameters, together with the hash-encoding parameters, jointly determine the coefficients $C_{\boldsymbol{\alpha}}$ in this multivariate polynomial. Moreover, eq. (11) reveals the locality of the neural network output: for given $\mathbf{x}$ and grid resolution N_l, only four neighboring linear B-splines take a non-zero value on $\mathbf{x}$, **so the network output at $\mathbf{x}$ depends on the nearest $4L$ linear B-splines across L grid resolutions. This suggests that the B-spline locality in NGP encoding is transferred to the network output**, making it better suited to the local nature of CT images than Fourier encoding.

4 Generalization Beyond Linear B-spline INRs

We replace the linear B-spline in eq. (3) with B-spline of arbitrary order r:

$$y_l^{(r)}(x) := \sum_{p\in\{-\lfloor (r-1)/2\rfloor,\dots,N_l+\lfloor (r-1)/2\rfloor\}^2} B_2^{(r)}(xN_l; p)\, \mathbf{T}_l[h_l(p)]. \tag{12}$$

The two-dimensional $B_2^{(r)}$ is defined as:

$$B_2^{(r)}\big(xN_l; p\big) = B_y^{(r)}(xN_l; (p_1, 0))\, B_x^{(r)}(xN_l; (0, p_2)), \tag{13}$$

where $p := (p_1, p_2) \in \{-\lfloor (r-1)/2\rfloor, \dots, N_l + \lfloor (r-1)/2\rfloor\}^2$ and

$$B_y^{(r)}((t,s); (p_1,0)) := B_1^{(r)}(t; p_1) \in R^1 \tag{14}$$

$$B_x^{(r)}((t,s); (0,p_2)) := B_1^{(r)}(s; p_2) \in R^1. \tag{15}$$

The constant, linear, quadratic, cubic and quartic B-spline ($r = 1, 2, 3, 4, 5$) is defined in Sec. A.5. And the visualization of $B_y^{(r)}$, $B_x^{(r)}$ and $B_2^{(r)}$ are shown in Fig. 8 to Fig. 12. Because CT images are not only piecewise, but each local region also exhibits smoothness. Moreover, taking into account the locality of the NGP neural network output discussed in sec. 3.6, replacing B_2 in eq. (3) from the linear B-spline to a higher order B-spline can improve the reconstruction accuracy, we confirmed this improvement in CT reconstruction experiments. Intuitively, Sec. A.7 illustrates the inherent approximation limitation of linear B-spline under the encoder-only setting.

In addition to improving NGP, we also characterized the impact of B-spline order-based encoders followed by a neural network. Specifically, we systematically examine arbitrary pairs of orders on CT reconstruction. The corresponding experimental results are presented in sec. 5.2, while the theoretically optimal approximation capability in the no-hashing scenario is given as follows:

Let F denote the neural network approximation obtained after Taylor truncation. We assume the ground truth corresponding to the reconstruction result is f^* with associated latent features $(y_1^*, \dots, y_L^*)$, such that $f^* = F(y_1^*, ..., y_L^*)$. Since CT images are typically piecewise smooth, we consider a smooth local region, restrict f^* and y_l^* to it, and let s denote the minimum smoothness of the encoded manifold in the corresponding area. The optimal approximation error E associated with the r order B-spline encoding can then be bounded as in eq. (16).

$$E(F(y_1^{(r)}, ..., y_L^{(r)}), f^*) \le \sup_{y_1,\dots,y_L,x} \|\nabla F(y_1, ..., y_L)\|_2 \left(\sum_{l=1}^{L} C_l \left(\frac{1}{N_l}\right)^{\min(r,s)} \right). \tag{16}$$

This error bound suggests that higher-order B-splines, when used as the encoder component, offer better optimal approximation capability. These properties also hold without the Taylor truncation assumption, where the gradient bound is replaced by the neural network's Lipschitz constant (Virmaux & Scaman, 2018). A detailed proof is provided in the Appendix sec. A.16.

5 EXPERIMENTS

5.1 SETUP

Datasets. We use four lung instances (task 6) and two liver instances (task 3) from the training set of Medical Segmentation Decathlon (Antonelli et al., 2022). For each lung instances, we uniformly sample 25 phantoms at slice indices ranging from 1 to 193 with a step size of 8, yielding 100 phantoms. The corresponding parallel-beam projections are generated using 80, 60, 40 and 20 views. We refer to this collection as the "Lung dataset". For each liver instances, we sample the primary liver regions (slices 380-456) with interval 4, yielding 40 phantoms, and generate projections using the same CT geometry and view settings. We refer to this collection as "Liver dataset".

Implementation details. We implement the B-spline encoder in both Python and CUDA versions, with CUDA accelerating encoding. To ensure a fair comparison, the layer depths, layer widths, activation function, network initialization, learning rates of neural network and encoding module are kept identical across all different B-spline orders. More details are provided in sec. A.9.

5.2 PERFORMANCE COMPARISON

Comparison methods. We compare our approach against several representative methods, including the analytical method FBP(Herman, 2009), iterative methods SIRT(Gregor & Benson, 2008) and NAG-LS(Nesterov, 2013; Neubauer, 2017), and INR-based methods such as SIREN(Sitzmann et al., 2020), RDFN (a Random Fourier Encoding-based Network)(Tancik et al., 2020), and NGP. All INR methods use neural networks with the same depth and width. To assess the effect of hash collisions in B-spline-based methods and NGP, we compare no-hash and hash collision-prone ($\log_2 T \leq 18$) settings. More details are provided in sec. A.10.

Table 1: Performance on the Lung dataset. Mean PSNR and SSIM consistently improve with B-spline order in both hashed and non-hashed settings (**Best**, <u>second-best</u>).

Methods	80-views		60-views		40-views		20-views	
	PSNR↑	SSIM↑	PSNR↑	SSIM↑	PSNR↑	SSIM↑	PSNR↑	SSIM↑
FBP	23.33	0.608	22.45	0.529	20.85	0.418	17.42	0.261
SIRT	29.08	0.748	27.88	0.693	26.41	0.610	24.38	0.490
NAG_LS	29.58	0.738	28.31	0.690	26.81	0.618	24.69	0.498
Siren	31.35	0.846	30.03	0.803	28.33	0.741	25.58	0.645
RDFN	33.13	0.899	32.38	0.879	30.68	0.825	26.07	0.653
NGP - w/o hash	33.88	0.905	32.55	0.879	30.70	0.835	27.41	0.738
NGP - w/ hash - $\log_2 T = 17$	33.73	0.901	32.40	0.873	30.57	0.829	27.39	0.736
Constant B-spline - w/o hash	33.44	0.897	32.07	0.868	30.20	0.821	27.10	0.727
Linear B-spline - w/o hash	33.90	0.906	32.58	0.879	30.70	0.835	27.44	0.740
Quadratic B-spline - w/o hash	34.16	0.911	32.78	0.884	30.90	0.841	27.61	0.749
Cubic B-spline - w/o hash	34.23	<u>0.912</u>	32.86	0.886	30.98	<u>0.844</u>	27.69	<u>0.753</u>
Quartic B-spline - w/o hash	34.26	**0.913**	32.89	**0.887**	<u>31.01</u>	**0.845**	<u>27.74</u>	**0.756**
Constant B-spline - w/ hash - $\log_2 T = 17$	33.11	0.889	31.81	0.860	30.03	0.813	27.02	0.722
Linear B-spline - w/ hash - $\log_2 T = 17$	33.79	0.903	32.48	0.875	30.64	0.832	27.39	0.737
Quadratic B-spline - w/ hash - $\log_2 T = 17$	34.20	0.911	32.81	0.884	30.90	0.840	27.58	0.745
Cubic B-spline - w/ hash - $\log_2 T = 17$	<u>34.28</u>	<u>0.912</u>	<u>32.90</u>	<u>0.886</u>	31.00	0.843	27.69	0.752
Quartic B-spline - w/ hash - $\log_2 T = 17$	**34.31**	**0.913**	**32.91**	<u>0.886</u>	**31.05**	**0.845**	**27.77**	**0.756**

Quantitative evaluation. Table 1 shows that NGP does not outperform the linear B-spline variant, as the offset in NGP becomes ineffective when the grid resolution is not a power of two. Across four viewing scenarios in table 1, the B-spline with linear and higher-order outperforms RDFN, demonstrating the advantage of B-spline over sinusoid.

Without hash collisions, we observe a consistent improvement in PSNR as the B-spline order increases from constant to quartic across all four view settings. In Lung dataset shown in table 1, compared with the linear B-spline, the quartic B-spline achieves an average PSNR improvement of 0.32 dB across different numbers of views. With hash collisions ($\log_2 T = 17$) , we similarly observe a consistent improvement in PSNR as the B-spline order increases from linear to quartic across all four view settings. Compared with the linear B-spline, the quartic B-spline improves PSNR by 0.52 dB, 0.43 dB, 0.41 dB, and 0.38 dB for 80, 60, 40, and 20 views, respectively. With hash colli-

Table 2: Wilcoxon signed-rank test for paired per-phantom improvements in best PSNR from lower- to higher-order B-spline encodings at $\log_2 T = 17$ (Lung dataset, 80 views, significance level=0.05).

H_1: **lower < higher**	n	**wins**	**ties**	**stat**	p	**sig**
constant B-spline < linear B-spline	100	99	0	5037.5	2.84×10^{-18}	✓
constant B-spline < quadratic B-spline	100	100	0	5050.0	1.95×10^{-18}	✓
constant B-spline < cubic B-spline	100	99	0	5049.0	2.01×10^{-18}	✓
constant B-spline < quartic B-spline	100	99	0	5049.0	2.01×10^{-18}	✓
linear B-spline < quadratic B-spline	100	96	0	5020.5	4.72×10^{-18}	✓
linear B-spline < cubic B-spline	100	95	0	5015.5	5.48×10^{-18}	✓
linear B-spline < quartic B-spline	100	96	1	4895.0	1.50×10^{-17}	✓
quadratic B-spline < cubic B-spline	100	63	4	3587.5	2.08×10^{-6}	✓
quadratic B-spline < quartic B-spline	100	65	3	3586.5	6.68×10^{-6}	✓
cubic B-spline < quartic B-spline	100	58	5	2952.5	6.27×10^{-3}	✓

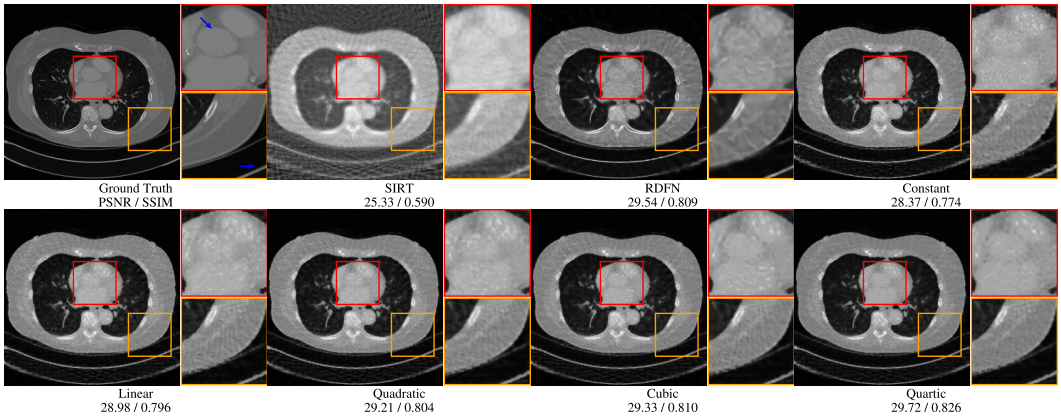


Figure 4: Reconstruction (40 views, $\log_2 T = 17$): Quartic achieves the best quality, whereas Linear exhibits noticeable artifacts. Fourier-encoded RDFN introduces severe wave-like artifacts.

sions, quartic B-splines perform better than their collision-free counterparts at all four view settings, suggesting that collisions degrade the advantage of local bases but bring regularization benefits. In the liver dataset, as shown in table 3, we also observed that performance increases with increasing order. For Lung dataset, table 2 and Appendix tables 6 to 8 present pairwise Wilcoxon signed-rank tests on the PSNR of different B-spline methods across 100 samples. For 20, 40, and 80 views, higher-order B-splines generally significantly outperformed B-splines of lower orders. The same trend held for 60 views, although the tendency between cubic and quartic was not significant. On the Liver dataset, we performed similar hypothesis tests as shown in Appendix tables 9 to 12 and found that the hypothesis that higher-order B-splines are superior to lower-order B-splines holds

Table 3: Performance on the Liver dataset. Mean PSNR and SSIM consistently improve with B-spline order in hashed settings (**Best**, second-best).

Methods	**80-views**		**60-views**		**40-views**		**20-views**	
	PSNR↑	SSIM↑	PSNR↑	SSIM↑	PSNR↑	SSIM↑	PSNR↑	SSIM↑
FBP	21.19	0.672	20.75	0.589	19.65	0.466	16.99	0.296
SIRT	29.11	0.809	27.85	0.748	26.15	0.651	23.67	0.528
NAG_LS	29.66	0.770	28.29	0.722	26.50	0.633	23.83	0.503
Siren	31.44	0.859	29.73	0.805	27.60	0.722	24.54	0.624
RDFN	33.28	0.927	32.56	0.908	31.14	0.859	26.53	0.700
Constant B-spline - w/ hash - $\log_2 T = 17$	32.54	0.903	31.49	0.878	30.06	0.839	26.79	0.746
Linear B-spline - w/ hash - $\log_2 T = 17$	33.46	0.926	32.40	0.903	30.82	0.864	27.19	0.767
Quadratic B-spline - w/ hash - $\log_2 T = 17$	33.76	0.933	32.70	0.911	31.06	0.872	27.32	0.774
Cubic B-spline - w/ hash - $\log_2 T = 17$	33.88	0.936	32.81	0.915	31.16	0.876	27.39	0.777
Quartic B-spline - w/ hash - $\log_2 T = 17$	**33.89**	**0.938**	**32.84**	**0.916**	**31.23**	**0.879**	**27.43**	**0.780**

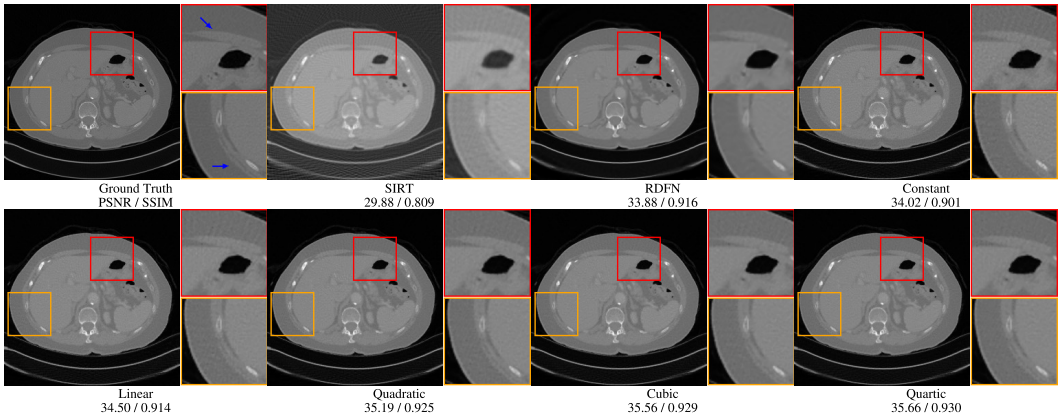
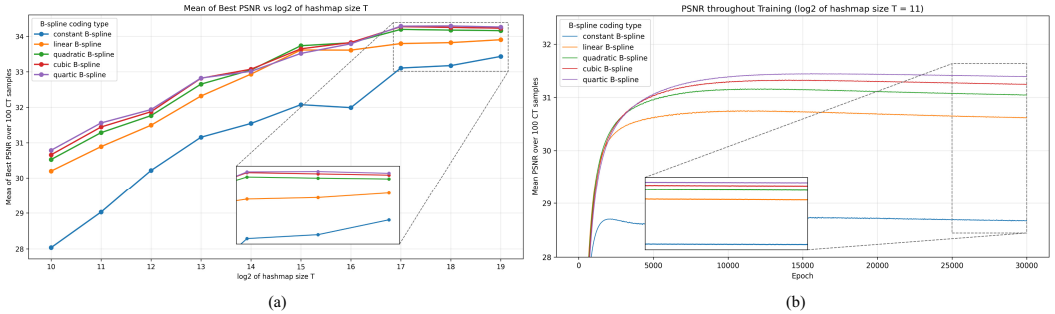


Figure 5: Reconstruction (80 views, $\log_2 T = 17$): Quartic achieves the best reconstruction quality, whereas the Linear exhibits more artifacts. Fourier-encoded RDFN blurs fine structures.



(a) (b)

Figure 6: (a) Five variants' performance when the hash table length T traversing from 2^{10} to 2^{19}. (b) Average accuracy per epoch when $T = 2^{11}$, with hash collisions occurring for all methods.

true, except for the comparison between cubic and quartic in the 80-view setting. Moreover, the average PSNR per epoch of B-splines under severe hash collisions ($T = 2^{11}$, Fig. 6 (b)) shows that the performance gain from higher-order B-splines is consistent across epochs.

Qualitative evaluation. Zooming in Fig. 5 and Fig. 4, we observe that all B-spline-based methods reconstruct local tissue structures more accurately than RDFN. This suggests that locally supported B-splines are better suited to reconstruct small regions with intricate structures than global Fourier encoding. As highlighted by the red and yellow boxes, the quartic B-spline formulation substantially reduces artifacts than the linear one. More visualization results are shown in Figs. 15 and 16. Complexity analysis and time consumption are provided in Appendix sec. A.11. Notably, the quartic version required only 5% more training time than the linear version.

6 Conclusion

We establish a correspondence between the encoding mechanism of NGP and the construction and placement of linear B-splines, while precisely characterizing the B-spline rearrangement induced by coordinate offsets in its official code. By applying Taylor truncation, we further show that the overall output of NGP can be expressed as a multivariate polynomial of B-splines, where updates to the neural network and encoding parameters are equivalent to adjusting the polynomial coefficients. Based on the locality of this approximate representation and the piecewise smoothness of CT images, we generalize NGP to a family of INRs based on different B-spline orders and instantiate five variants. In addition to the established bound, our experiments show that higher-order variants outperform the linear variant. Moreover, accuracy generally improves as the B-spline order increases. Through extensive experiments, we find that three variants outperform the compared methods. Finally, we accelerate these variants through hash encoding and CUDA programming.

REFERENCES

Mohammed Alsakabi, Wael Mobeirek, John M Dolan, and Ozan K Tonguz. Fm-siren & fm-finer: Implicit neural representation using nyquist-based orthogonality. *arXiv preprint arXiv:2509.23438*, 2025.

Anders H Andersen and Avinash C Kak. Simultaneous algebraic reconstruction technique (sart): a superior implementation of the art algorithm. *Ultrasonic imaging*, 6(1):81–94, 1984.

Michela Antonelli, Annika Reinke, Spyridon Bakas, Keyvan Farahani, Annette Kopp-Schneider, Bennett A Landman, Geert Litjens, Bjoern Menze, Olaf Ronneberger, Ronald M Summers, et al. The medical segmentation decathlon. *Nature communications*, 13(1):4128, 2022.

Jonathan T Barron, Ben Mildenhall, Dor Verbin, Pratul P Srinivasan, and Peter Hedman. Zip-nerf: Anti-aliased grid-based neural radiance fields. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 19640–19648. IEEE, 2023.

Murat Belge, Misha E Kilmer, and Eric L Miller. Wavelet domain image restoration with adaptive edge-preserving regularization. *IEEE Transactions on Image Processing*, 9(4):597–608, 2000.

Yuanhao Cai, Jiahao Wang, Alan Yuille, Zongwei Zhou, and Angtian Wang. Structure-aware sparse-view x-ray 3d reconstruction. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11174–11183. IEEE, 2024.

Charles K Chui and Jian-zhong Wang. A cardinal spline approach to wavelets. *Proceedings of the American Mathematical Society*, 113(3):785–793, 1991.

Tianxiang Dai and Jonathan Fan. Characterizing and optimizing the spatial kernel of multi resolution hash encodings. *arXiv preprint arXiv:2602.10495*, 2026.

Carl De Boor and Carl De Boor. *A practical guide to splines*, volume 27. springer New York, 1978.

Yixiao Feng, Weihua Tong, and Zhangjin Huang. High-quality neural surface reconstruction from unoriented point clouds via multilevel tensor product b-spline hash encoding and viscosity regularization. *The Visual Computer*, 41(9):6701–6714, 2025.

Richard Gordon, Robert Bender, and Gabor T Herman. Algebraic reconstruction techniques (art) for three-dimensional electron microscopy and x-ray photography. *Journal of theoretical Biology*, 29(3):471–481, 1970.

Jens Gregor and Thomas Benson. Computational analysis and improvement of sirt. *IEEE transactions on medical imaging*, 27(7):918–924, 2008.

Gabor T Herman. *Fundamentals of computerized tomography: image reconstruction from projections*. Springer Science & Business Media, 2009.

S Horbelt, M Liebling, and M Unser. Discretization of the radon transform and of its inverse by spline convolutions. *IEEE Transactions on Medical Imaging*, 21(4):363–376, 2002.

Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, George Drettakis, et al. 3d gaussian splatting for real-time radiance field rendering. *ACM Trans. Graph.*, 42(4):139–1, 2023.

Xin Li, Haochuan Wan, Kangjie Long, Qing Wu, Chenhe Du, Ying Su, Zhe Luo, Xin Lou, and Yuyao Zhang. A real-time neural representation via algorithm-hardware synergy for sparse-view ct reconstruction. *IEEE Transactions on Neural Networks and Learning Systems*, 2025.

Christian HX Ali Mehmeti-Göpel, David Hartmann, and Michael Wand. Ringing relus: Harmonic distortion analysis of nonlinear feedforward networks. In *International Conference on Learning Representations*, 2020.

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

Yurii Nesterov. *Introductory lectures on convex optimization: A basic course*. Springer Science & Business Media, 2013.

Andreas Neubauer. On nesterov acceleration for landweber iteration of linear ill-posed problems. *Journal of Inverse and Ill-posed Problems*, 25(3):381–390, 2017.

Hyoung Suk Park and Kiwan Jeon. Adaptive multi-resolution hash-encoding framework for inr-based dental cbct reconstruction with truncated fov. *Medical Physics*, 53(1):e70226, 2026.

Darius Rückert, Yuanhao Wang, Rui Li, Ramzi Idoughi, and Wolfgang Heidrich. Neat: Neural adaptive tomography. *ACM Transactions on Graphics (TOG)*, 41(4):1–13, 2022.

Vishwanath Saragadam, Daniel LeJeune, Jasper Tan, Guha Balakrishnan, Ashok Veeraraghavan, and Richard G Baraniuk. Wire: Wavelet implicit neural representations. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 18507–18516, 2023.

Kathan Shah and Chawin Sitawarin. Spder: Semiperiodic damping-enabled object representation. In *International Conference on Learning Representations*, volume 2024, pp. 55780–55810, 2024.

Hua-Chieh Shao, Tielige Mengke, Tinsu Pan, and You Zhang. Dynamic cbct imaging using prior model-free spatiotemporal implicit neural representation (pmf-stinr). *Physics in Medicine & Biology*, 69(11):115030, 2024.

Liyue Shen, John Pauly, and Lei Xing. Nerp: implicit neural representation learning with prior embedding for sparsely sampled image reconstruction. *IEEE transactions on neural networks and learning systems*, 35(1):770–782, 2022.

Emil Y Sidky and Xiaochuan Pan. Image reconstruction in circular cone-beam computed tomography by constrained, total-variation minimization. *Physics in Medicine & Biology*, 53(17):4777–4807, 2008.

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

Shiyan Su, Ruyi Zha, Danli Shi, Hongdong Li, and Xuelian Cheng. Diffnr: Diffusion-enhanced neural representation optimization for sparse-view 3d tomographic reconstruction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 9144–9152, 2026.

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

Aladin Virmaux and Kevin Scaman. Lipschitz regularity of deep neural networks: analysis and efficient estimation. *Advances in neural information processing systems*, 31, 2018.

Qing Wu, Ruimin Feng, Hongjiang Wei, Jingyi Yu, and Yuyao Zhang. Self-supervised coordinate projection network for sparse-view computed tomography. *IEEE Transactions on Computational Imaging*, 9:517–529, 2023.

Qing Wu, Hongjiang Wei, Jingyi Yu, and Yuyao Zhang. Unsupervised multi-parameter inverse solving for reducing ring artifacts in 3d x-ray cbct. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 10709–10717, 2026.

Wangduo Xie and Matthew Blaschko. Nab: Neural adaptive binning for sparse-view ct reconstruction. In *International Conference on Learning Representations*, volume 2026, pp. 140098–140115, 2026.

Wangduo Xie, Richard Schoonhoven, Tristan Van Leeuwen, and Matthew B Blaschko. Ac-ind: Sparse ct reconstruction based on attenuation coefficient estimation and implicit neural distribution. In *2025 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pp. 3074–3083. IEEE Computer Society, 2025.

Zhuowei Xu, Han Li, Dai Sun, Zhicheng Li, Yujia Li, Qingpeng Kong, Zhiwei Cheng, Nassir Navab, and Shaohua Kevin Zhou. Nerf-based cbct reconstruction needs normalization and initialization. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pp. 354–364, 2025.

Gizem Yüce, Guillermo Ortiz-Jiménez, Beril Besbinar, and Pascal Frossard. A structured dictionary perspective on implicit neural representations. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 19206–19216. IEEE, 2022.

Ruyi Zha, Yanhao Zhang, and Hongdong Li. Naf: neural attenuation fields for sparse-view cbct reconstruction. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pp. 442–452. Springer, 2022.

Ruyi Zha, Tao Jun Lin, Yuanhao Cai, Jiwen Cao, Yanhao Zhang, and Hongdong Li. R^2-gaussian: Rectifying radiative gaussian splatting for tomographic reconstruction. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Qingyang Zhou, Yunfan Ye, Zhihuang Liu, Chang Liu, and Zhiping Cai. Non-local guided neural fields for 4d ct reconstruction. *IEEE Transactions on Circuits and Systems for Video Technology*, 2025.

A APPENDIX

TABLE OF CONTENTS

A.1 Limitations and future work 14
A.2 Visualizations and algorithm descriptions of NGP 14
A.3 Definition of 1D and 2D linear B-spline 15
A.4 Supplementary proof of encoder equivalent to linear B-spline 15
A.5 Definition of 1D constant, linear, quadratic, cubic, and quartic B-splines 16
A.6 Visualization of 1D and 2D constant, linear, quadratic, cubic, and quartic B-splines 17
A.7 Linear B-spline approximation 18
A.8 Non-Coincidence of Peak Points 18
A.9 Supplement to B-spline methods implementation details 19
A.10 Supplement to comparison methods implementation Details 19
A.11 Computational complexity analysis 19
A.12 Wilcoxon signed-rank test in the Lung dataset 20
A.13 Wilcoxon signed-rank test in the Liver dataset 21
A.14 Average reconstruction accuracy per epoch 22
A.15 Visual reconstruction results on the liver dataset 22
A.16 Proof of the approximation error 24
A.17 AI use statement 25

A.1 Limitations and future work

We do not include 3DGS-based methods (Kerbl et al., 2023; Zha et al., 2024) in comparison. Although 3DGS's approximation capability is generally lower than that of neural network-based methods, they offer advantages in computational efficiency and will therefore be considered in future work. In addition, the proposed method has not yet been evaluated on a broader range of reconstruction tasks, such as other biomedical imaging modalities. In future work, we plan to investigate its applicability to MRI, PET, Cryo-EM, Ultrasound reconstruction to further assess its generalizability. Furthermore, the proposed higher-order B-spline methods are closely related to wavelets. Exploring how wavelet theory can be leveraged to further improve these methods, both theoretically and algorithmically, represents another important direction for future work.

A.2 Visualizations and algorithm descriptions of NGP

This section is supplementary to the sec. 3.1 in the main text.

To establish a geometric and approximation-theoretic interpretation of NGP (Müller et al., 2022) and subsequently extend it to variable order representations, we first review the original formulation and use consistent notation throughout. Without loss of generality, we illustrate the formulation in two dimensions.

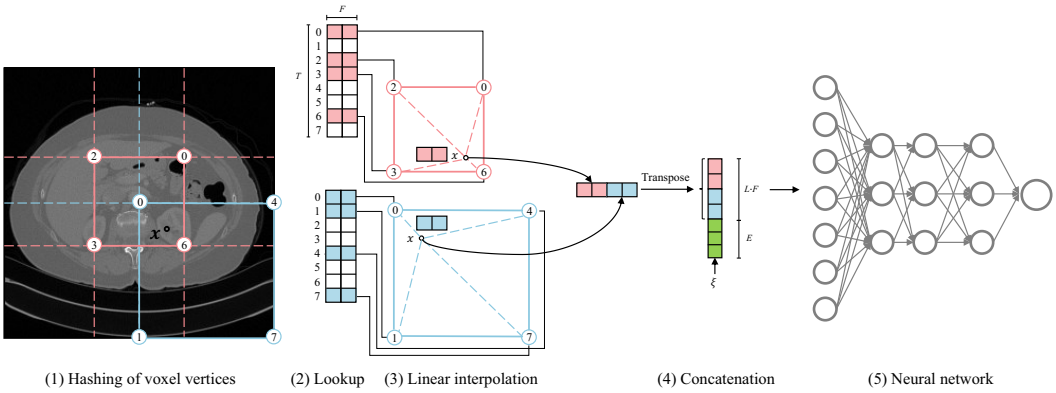


Figure 7: Illustration of multiresolution hash encoding in a 2D coordinate system, redrawn based on Müller et al. (2022).

Let $\mathbf{x} \in [0,1) \times [0,1)$ denote the normalized natural coordinate of a point P in a 2D domain. At each level l, each coordinate axis is uniformly partitioned into N_l intervals, yielding $(N_l \times N_l)$ cells over the domain. The 2D index of the cell containing P is given by $\lfloor \mathbf{x} N_l \rfloor$.

The local coordinate of P in its enclosing cell is defined as eq. (17):

$$\{\mathbf{x} N_l\} := \mathbf{x} N_l - \lfloor \mathbf{x} N_l \rfloor. \tag{17}$$

For each vertex of the cell containing the point P, namely $\lfloor \mathbf{x} N_l \rfloor + (0,0)$, $\lfloor \mathbf{x} N_l \rfloor + (0,1)$, $\lfloor \mathbf{x} N_l \rfloor + (1,0)$, and $\lfloor \mathbf{x} N_l \rfloor + (1,1)$, NGP retrieves a corresponding feature vector from the hash table:

$$\mathbf{f}_{l,\boldsymbol{\epsilon}} = \mathbf{T}_l[h_l(\lfloor \mathbf{x} N_l \rfloor + \boldsymbol{\epsilon})]\,. \tag{18}$$

where $\boldsymbol{\epsilon} \in \{0,1\}^2$ indexes the four neighboring vertices, $h_l(\cdot)$ denotes the spatial hash function and $\mathbf{T}_l$ is the learnable feature table (hash table) at level l.

The l-th-level feature produced by the encoding module is then obtained via linear interpolation, as defined in eq. (19).

$$\begin{aligned} y_l(\mathbf{x}) = {} & (1 - \{\mathbf{x} N_l\}[0])(1 - \{\mathbf{x} N_l\}[1]) f_{l,(0,0)} + \{\mathbf{x} N_l\}[0](1 - \{\mathbf{x} N_l\}[1]) f_{l,(1,0)} \\ & + (1 - \{\mathbf{x} N_l\}[0])\{\mathbf{x} N_l\}[1] f_{l,(0,1)} + \{\mathbf{x} N_l\}[0]\{\mathbf{x} N_l\}[1] f_{l,(1,1)}. \end{aligned} \tag{19}$$

Fig. 7 provides a visual illustration of the process described above.

A.3 DEFINITION OF 1D AND 2D LINEAR B-SPLINE

This section supplements both sec. 3.3 in the main text and sec. A.4 in the appendix.

The 1D linear B-spline basis function with symmetry center $c \in R^1$ is

$$B_1(t;c) = \max\big(0, 1 - |t - c|\big), t \in R^1 \tag{20}$$

which is the standard hat function. For a 2D input coordinate space R^2, translating the 1D linear B-spline along the y-direction yields a naive 2D linear B-spline, as shown in the subgraph (a) in Fig. 9. The formula is as follows:

$$B_y((u,v);(c_1,0)) = \max\big(0, 1 - |u - c_1|\big), c_1, u, v \in R^1 \tag{21}$$

Translating the 1D linear B-spline along the x-direction yields a naive 2D linear B-spline, as shown in the subgraph (b) in Fig. 9, the formula is as follows:

$$B_x((u,v);(0,c_2)) = \max\big(0, 1 - |v - c_2|\big), c_2, u, v \in R^1 \tag{22}$$

Visualization of $B_y((u,v);(c_1,0))$ and $B_x((u,v);(0,c_2))$ are show in (a) and (b) of Fig. 9 respectively.

According to the eq. (22) and eq. (21), we define an standard 2D linear B-spline $B_2((u,v);(c_1,c_2))$:

$$B_2((u,v);(c_1,c_2)) := B_x((u,v);(0,c_2))B_y((u,v);(c_1,0)), c_1, c_2, u, v \in R^1 \tag{23}$$

Geometrically, $B_2((u,v);(c_1,c_2))$ forms a pyramid shape, and its center of symmetry is located at (c_1,c_2). Visualization of $B_2((u,v);(c_1,c_2))$ is shown in the subgraph (c) in Fig. 9.

For any $\alpha, \beta \in R^1$, The eq. (23) has a translation invariance:

$$B_2((u,v);(c_1,c_2)) = B_2((u+\alpha, v+\beta);(c_1+\alpha, c_2+\beta)) \tag{24}$$

A.4 SUPPLEMENTARY PROOF OF ENCODER EQUIVALENT TO LINEAR B-SPLINE

This section is supplementary to the sec. 3.2 in the main text.

Using this 1D linear B-spline basis function given in eq. (20), eq. (1)(or, equivalently, eq. (19))can be written as:

$$\begin{aligned} y_l(\mathbf{x}) = {} & B_1(\{\mathbf{x}N_l\}[0];0)B_1(\{\mathbf{x}N_l\}[1];0)\mathbf{T}_l[h_l(\lfloor \mathbf{x}N_l \rfloor + (0,0))] \\ & + B_1(\{\mathbf{x}N_l\}[0];1)B_1(\{\mathbf{x}N_l\}[1];0)\mathbf{T}_l[h_l(\lfloor \mathbf{x}N_l \rfloor + (1,0))] \\ & + B_1(\{\mathbf{x}N_l\}[0];0)B_1(\{\mathbf{x}N_l\}[1];1)\mathbf{T}_l[h_l(\lfloor \mathbf{x}N_l \rfloor + (0,1))] \\ & + B_1(\{\mathbf{x}N_l\}[0];1)B_1(\{\mathbf{x}N_l\}[1];1)\mathbf{T}_l[h_l(\lfloor \mathbf{x}N_l \rfloor + (1,1))]. \end{aligned} \tag{25}$$

By using eqs. (21) and (22), the $y_l(\mathbf{x})$ in eq. (25) is equivalent to following expression:

$$\begin{aligned} & B_y((\{\mathbf{x}N_l\}[0],\{\mathbf{x}N_l\}[1]);(0,0))\, B_x((\{\mathbf{x}N_l\}[0],\{\mathbf{x}N_l\}[1]);(0,0))\, \mathbf{T}_l[h_l(\lfloor \mathbf{x}N_l \rfloor + (0,0))] \\ & + B_y((\{\mathbf{x}N_l\}[0],\{\mathbf{x}N_l\}[1]);(1,0))\, B_x((\{\mathbf{x}N_l\}[0],\{\mathbf{x}N_l\}[1]);(0,0))\, \mathbf{T}_l[h_l(\lfloor \mathbf{x}N_l \rfloor + (1,0))] \\ & + B_y((\{\mathbf{x}N_l\}[0],\{\mathbf{x}N_l\}[1]);(0,0))\, B_x((\{\mathbf{x}N_l\}[0],\{\mathbf{x}N_l\}[1]);(0,1))\, \mathbf{T}_l[h_l(\lfloor \mathbf{x}N_l \rfloor + (0,1))] \\ & + B_y((\{\mathbf{x}N_l\}[0],\{\mathbf{x}N_l\}[1]);(1,0))\, B_x((\{\mathbf{x}N_l\}[0],\{\mathbf{x}N_l\}[1]);(0,1))\, \mathbf{T}_l[h_l(\lfloor \mathbf{x}N_l \rfloor + (1,1))]. \end{aligned} \tag{26}$$

By using eq. (23), the eq. (26) are transformed to following expression:

$$
\begin{aligned}
y_l(\mathbf{x}) = & B_2((\{\mathbf{x}N_l\}[0], \{\mathbf{x}N_l\}[1]); (0,0))\, \mathbf{T}_l[h_l(\lfloor \mathbf{x}N_l \rfloor + (0,0))] \\
& + B_2((\{\mathbf{x}N_l\}[0], \{\mathbf{x}N_l\}[1]); (1,0))\, \mathbf{T}_l[h_l(\lfloor \mathbf{x}N_l \rfloor + (1,0))] \\
& + B_2((\{\mathbf{x}N_l\}[0], \{\mathbf{x}N_l\}[1]); (0,1))\, \mathbf{T}_l[h_l(\lfloor \mathbf{x}N_l \rfloor + (0,1))] \\
& + B_2((\{\mathbf{x}N_l\}[0], \{\mathbf{x}N_l\}[1]); (1,1))\, \mathbf{T}_l[h_l(\lfloor \mathbf{x}N_l \rfloor + (1,1))].
\end{aligned} \tag{27}
$$

Therefore, the encoding process of NGP can be interpreted as a linear combination of four 2D linear B-spline basis functions centered at the four corners (0,0), (1,0), (0,1), and (1,1). The value of each function is precisely controlled by the point x inside the square cell formed by the vertices (0,0), (0,1), (1,1), and (1,0). When performing the linear combination, 2D linear B-splines has the following linear combination coefficients: $\mathbf{T}_l[h_l(\lfloor \mathbf{x}N_l \rfloor + (0,0))]$, $\mathbf{T}_l[h_l(\lfloor \mathbf{x}N_l \rfloor + (1,0))]$, $\mathbf{T}_l[h_l(\lfloor \mathbf{x}N_l \rfloor + (0,1))]$, $\mathbf{T}_l[h_l(\lfloor \mathbf{x}N_l \rfloor + (1,1))]$.

By using eq. (24), the $y_l(\mathbf{x})$ in eq. (27) can be rewritten as:

$$
\begin{aligned}
& B_2((\mathbf{x}N_l[0], \mathbf{x}N_l[1]); (\lfloor \mathbf{x}N_l[0] \rfloor, \lfloor \mathbf{x}N_l[1] \rfloor))\, \mathbf{T}_l[h_l(\lfloor \mathbf{x}N_l[0] \rfloor, \lfloor \mathbf{x}N_l[1] \rfloor)] \\
& + B_2((\mathbf{x}N_l[0], \mathbf{x}N_l[1]); (\lfloor \mathbf{x}N_l[0] \rfloor + 1, \lfloor \mathbf{x}N_l[1] \rfloor))\, \mathbf{T}_l[h_l(\lfloor \mathbf{x}N_l[0] \rfloor + 1, \lfloor \mathbf{x}N_l[1] \rfloor)] \\
& + B_2((\mathbf{x}N_l[0], \mathbf{x}N_l[1]); (\lfloor \mathbf{x}N_l[0] \rfloor, \lfloor \mathbf{x}N_l[1] \rfloor + 1))\, \mathbf{T}_l[h_l(\lfloor \mathbf{x}N_l[0] \rfloor, \lfloor \mathbf{x}N_l[1] \rfloor + 1)] \\
& + B_2((\mathbf{x}N_l[0], \mathbf{x}N_l[1]); (\lfloor \mathbf{x}N_l[0] \rfloor + 1, \lfloor \mathbf{x}N_l[1] \rfloor + 1))\mathbf{T}_l[h_l(\lfloor \mathbf{x}N_l[0] \rfloor + 1, \lfloor \mathbf{x}N_l[1] \rfloor + 1)]
\end{aligned} \tag{28}
$$

The eq. (28) can be equivalently transformed in a more concise form:

$$
\begin{aligned}
y_l(\mathbf{x}) = & B_2(\mathbf{x}N_l; \lfloor \mathbf{x}N_l \rfloor + (0,0))\, \mathbf{T}_l[h_l(\lfloor \mathbf{x}N_l \rfloor + (0,0))] \\
& + B_2(\mathbf{x}N_l; \lfloor \mathbf{x}N_l \rfloor + (1,0))\, \mathbf{T}_l[h_l(\lfloor \mathbf{x}N_l \rfloor + (1,0))] \\
& + B_2(\mathbf{x}N_l; \lfloor \mathbf{x}N_l \rfloor + (0,1))\, \mathbf{T}_l[h_l(\lfloor \mathbf{x}N_l \rfloor + (0,1))] \\
& + B_2(\mathbf{x}N_l; \lfloor \mathbf{x}N_l \rfloor + (1,1))\, \mathbf{T}_l[h_l(\lfloor \mathbf{x}N_l \rfloor + (1,1))]
\end{aligned} \tag{29}
$$

Let

$$
\mathcal{N}_n := \{\lfloor \mathbf{x}N_l \rfloor + (0,0), \lfloor \mathbf{x}N_l \rfloor + (1,0), \lfloor \mathbf{x}N_l \rfloor + (0,1), \lfloor \mathbf{x}N_l \rfloor + (1,1)\},
$$

For any $p \in \{0, \ldots, N_l\}^2 \backslash \mathcal{N}_n$, the corresponding B-spline has a value of 0 at $\mathbf{x}$, i.e., $B_2(\mathbf{x}N_l; p) = 0$, for $p \in \{0, \ldots, N_l\}^2 \setminus \mathcal{N}_n$. Thus, eq. (29) can be rewritten as:

$$
\begin{aligned}
y_l(\mathbf{x}) = & B_2(\mathbf{x}N_l; \lfloor \mathbf{x}N_l \rfloor + (0,0))\, \mathbf{T}_l[h_l(\lfloor \mathbf{x}N_l \rfloor + (0,0))] \\
& + B_2(\mathbf{x}N_l; \lfloor \mathbf{x}N_l \rfloor + (1,0))\, \mathbf{T}_l[h_l(\lfloor \mathbf{x}N_l \rfloor + (1,0))] \\
& + B_2(\mathbf{x}N_l; \lfloor \mathbf{x}N_l \rfloor + (0,1))\, \mathbf{T}_l[h_l(\lfloor \mathbf{x}N_l \rfloor + (0,1))] \\
& + B_2(\mathbf{x}N_l; \lfloor \mathbf{x}N_l \rfloor + (1,1))\, \mathbf{T}_l[h_l(\lfloor \mathbf{x}N_l \rfloor + (1,1))] \\
& + \sum_{p \in \{0, \ldots, N_l\}^2 \backslash \mathcal{N}_n} B_2(\mathbf{x}N_l; p)\, \mathbf{T}_l[h_l(p)].
\end{aligned} \tag{30}
$$

A.5 Definition of 1D constant, linear, quadratic, cubic, and quartic B-splines

This section is supplementary to the sec. 4 in the main text.

The constant one-dimensional B-spline function with center of symmetry $c \in R^1$ is defined as:

$$
B_1^{(1)}(t; c) = \begin{cases} 1, & |t - c| \leq \frac{1}{2}, \\ 0, & |t - c| > \frac{1}{2}. \end{cases} \tag{31}
$$

The linear one-dimensional B-spline function with center of symmetry $c \in R^1$ is defined as:

$$
B_1^{(2)}(t; c) = \begin{cases} 1 - |t - c|, & |t - c| \leq 1, \\ 0, & |t - c| > 1. \end{cases} \tag{32}
$$

The quadratic one-dimensional B-spline function with center of symmetry $c \in R^1$ is defined as:

$$
B_1^{(3)}(t; c) = \begin{cases} \frac{3}{4} - (t - c)^2, & |t - c| \leq \frac{1}{2}, \\ \frac{1}{2}\left(|t - c| - \frac{3}{2}\right)^2, & \frac{1}{2} < |t - c| \leq \frac{3}{2}, \\ 0, & |t - c| > \frac{3}{2}. \end{cases} \tag{33}
$$

The cubic one-dimensional B-spline function with center of symmetry $c \in R^1$ is defined as:

$$B_1^{(4)}(t;c) = \begin{cases} \frac{4-6|t-c|^2+3|t-c|^3}{6}, & |t-c| < 1, \\ \frac{(2-|t-c|)^3}{6}, & 1 \le |t-c| < 2, \\ 0, & |t-c| \ge 2. \end{cases} \tag{34}$$

The quartic one-dimensional B-spline function with center of symmetry $c \in R^1$ is defined as:

$$B_1^{(5)}(t;c) = \begin{cases} \frac{1}{4}|t-c|^4 - \frac{5}{8}|t-c|^2 + \frac{115}{192}, & |t-c| < \frac{1}{2}, \\ -\frac{1}{6}|t-c|^4 + \frac{5}{6}|t-c|^3 - \frac{5}{4}|t-c|^2 + \frac{5}{24}|t-c| + \frac{55}{96}, & \frac{1}{2} \le |t-c| < \frac{3}{2}, \\ \frac{1}{24}|t-c|^4 - \frac{5}{12}|t-c|^3 + \frac{25}{16}|t-c|^2 - \frac{125}{48}|t-c| + \frac{625}{384}, & \frac{3}{2} \le |t-c| < \frac{5}{2}, \\ 0, & |t-c| \ge \frac{5}{2}. \end{cases} \tag{35}$$

A.6 Visualization of 1D and 2D constant, linear, quadratic, cubic, and quartic B-splines

This section is supplementary to the sec. 4 in the main text.

The 1D and 2D constant, linear, quadratic, cubic, and quartic B-splines are shown in Figs. 8 to 12.

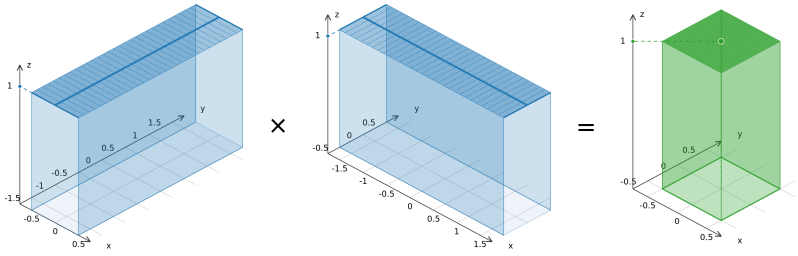


Figure 8: Visualization of the constant B-spline basis functions along the y and x axes and their product representation.

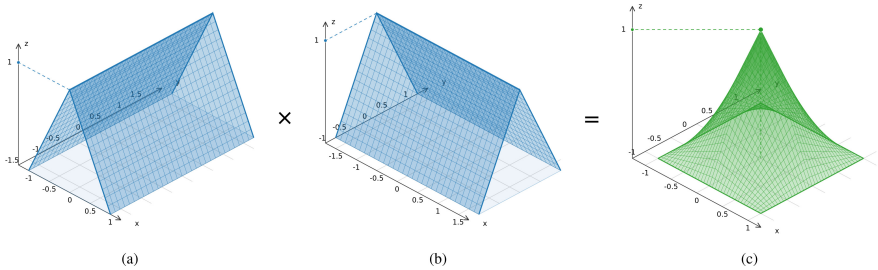


Figure 9: (a):The translation result of 1D linear B-spline along the y-direction.(b):The translation result of 1D linear B-spline along the x-direction. (c): The 2D linear B-spline, which is the product of the 1D linear B-spline from the x-direction and 1D linear B-spline from the y-direction.

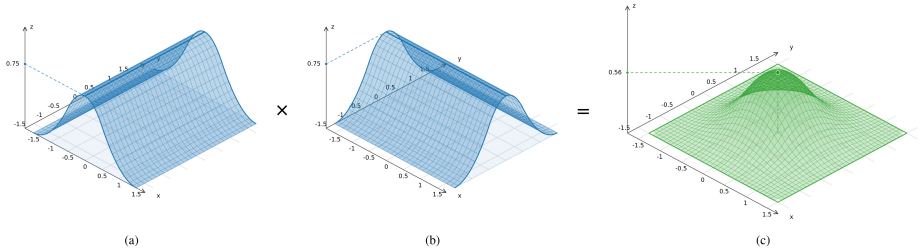


Figure 10: Visualization of the quadratic B-spline basis functions along the y and x axes and their product representation.

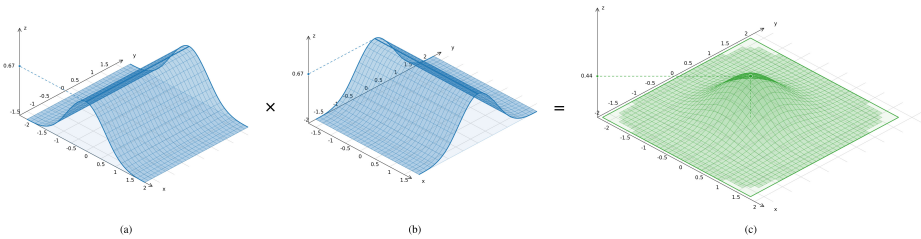


Figure 11: Visualization of the cubic B-spline basis functions along the y and x axes and their product representation.

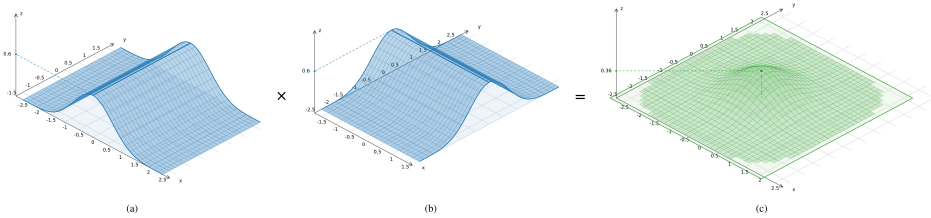


Figure 12: Visualization of the quartic B-spline basis functions along the y and x axes and their product representation.

A.7 Linear B-spline approximation

This section is supplementary to the sec. 4 in the main text.

We selected an aircraft image from CIFAR-10 as the approximation target. Fig. 13 illustrates how a set of B-spline basis functions with fixed spatial positions fits the target image without neural networks or hash encoding. Their coefficients are determined using the least-squares method.

A.8 Non-Coincidence of Peak Points

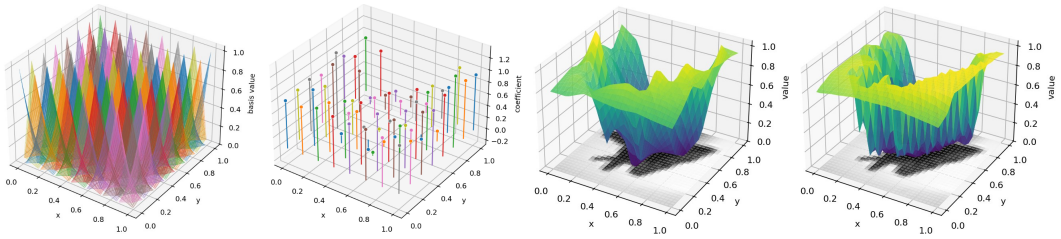
This section is supplementary to the sec. 3.5 in the main text.

At resolution level l, the set of all possible peak points in the offset grid is

$$C_l := \left\{ \frac{k}{2^l} \pm \frac{1}{2^{l+1}} \,\middle|\, k = 0, 1, \ldots, 2^l \right\}.$$

Moreover, for any two distinct resolution levels $l \neq p$, these sets are disjoint:

$$C_l \cap C_p = \emptyset.$$



(a) (b) (c) (d)

Figure 13: (a) Placement of linear B-spline clusters.(b) Approximation coefficients before each Bspline. (c) Approximation result. (d) Target from CIFAR-10.

The proof is as follows:

We use proof by contradiction. Assume the intersection is not empty, then there exist $i, j \in \mathbb{N}_0$, such that $\frac{i}{2^l} \pm \frac{1}{2^{l+1}} = \frac{j}{l} 2^p \pm \frac{1}{2^{p+1}}$. Based on symmetry, let's assume $p > l$, then $i = \frac{2j \pm 1 \pm 2^{p-l}}{2^{(p+1-l)}}$, since the numerator is odd and the denominator is even, the right side will not be an integer. So the two terms on the left and right sides of the equation are contradictory.

A.9 Supplement to B-spline methods implementation details

This section is supplementary to the sec. 5.1 in the main text.

All B-spline variants are trained for 30,000 epochs. Regardless of the B-spline order, all B-spline-based methods use a neural network with a depth of 3 and a hidden-layer width of 256. For the lung and liver datasets, the learning rates for the neural network and B-spline encoding are set to 1×10^{-5} and 1×10^{-1}, respectively. We set the finest and base grid resolutions to $R_{\text{finest}} = 512$ and $R_{\text{base}} = 16$, respectively. We set up n_{levels} ($n_{\text{levels}} = 16$) grids with progressively increasing resolutions, where the resolution R_l is calculated as follows:

$$R_l = \left\lfloor R_{\text{base}} \left[\exp\left(\frac{\log\left(R_{\text{finest}}/R_{\text{base}}\right)}{n_{\text{levels}} - 1} \right) \right]^i \right\rfloor, \qquad l = 0, 1, \ldots, n_{\text{levels}} - 1. \tag{36}$$

Accordingly, the grid resolutions $R_l (l = 0, 1, ..., 15)$ are: 16, 20, 25, 32, 40, 50, 64, 80, 101, 128, 161, 203, 256, 322, 406 and 512.

A.10 Supplement to comparison methods implementation Details

This section is supplementary to the sec. 5.2 in the main text.

We perform a hyperparameter search to determine the optimal standard deviation for the initial Fourier frequencies and the optimal learning rates (lr) of RDFN. At initialization, the Fourier frequencies are sampled from a Gaussian distribution with mean 0 and standard deviation 4. We sample 256 frequencies and apply sine and cosine functions to each, resulting in a 512-dimensional coordinate encoding. The neural network's lr of RDFN is set to 1×10^{-4} for the Lung dataset and 2×10^{-4} for the Liver dataset. For Siren, after the hyperparameter search, we set its neural network's lr to 1×10^{-5}. Both RDFN and Siren are trained for 30,000 epochs. For NGP, we use the official implementation and align its resolution definition with ours. All deep-learning-based comparison methods use a neural network with a depth of 3 and a hidden-layer width of 256. For both SIRT and NAG_LS, we set the number of iterations to 20,000. For a fair comparison, all methods included in the accuracy comparison use their best PSNR values for reporting.

A.11 Computational complexity analysis

This section is supplementary to the sec. 5.2 in the main text.

When `torch.where` is used to implement different B-spline basis functions in pure Python, the computational complexity of evaluating a 1-D B-spline function of order p is $O(p^2)$. Therefore, for ($p \geq 1$), the computational complexity T_c of encoding with order p B-splines is given by:

$$T_c = O\big(NL\left[dp^2 + Fp^d\right]\big)\,. \tag{37}$$

Here, N represents the number of pixels(voxel) of the object to be reconstructed. L represents the number of levels in the multi-resolution strategy. F denotes the number of columns in the weighting tables.

For a given object, table 4 also reports the actual runtime of the methods using CUDA-accelerated encoding for 80-view reconstruction.

Table 4: Total training time for 30,000 epochs for different methods.

Method	Constant	Linear	Quadratic	Cubic	Quartic
Order	1	2	3	4	5
Time (s)	258.6	264.4	264.9	269.1	276.8

A.12 Wilcoxon signed-rank test in the Lung dataset

This section is supplementary to the sec. 5.2 in the main text.

The Wilcoxon signed-rank tests for the Lung dataset are shown in tables 5 to 8.

Table 5: Wilcoxon signed-rank test for paired per-phantom improvements in best PSNR from lower- to higher-order B-spline encodings at $\log_2 T = 17$ (Lung dataset, 80 views, significance level=0.05).

H_1: **lower < higher**	n	**wins**	**ties**	**stat**	p	**sig**
constant B-spline < linear B-spline	100	99	0	5037.5	2.84×10^{-18}	✓
constant B-spline < quadratic B-spline	100	100	0	5050.0	1.95×10^{-18}	✓
constant B-spline < cubic B-spline	100	99	0	5049.0	2.01×10^{-18}	✓
constant B-spline < quartic B-spline	100	99	0	5049.0	2.01×10^{-18}	✓
linear B-spline < quadratic B-spline	100	96	0	5020.5	4.72×10^{-18}	✓
linear B-spline < cubic B-spline	100	95	0	5015.5	5.48×10^{-18}	✓
linear B-spline < quartic B-spline	100	96	1	4895.0	1.50×10^{-17}	✓
quadratic B-spline < cubic B-spline	100	63	4	3587.5	2.08×10^{-6}	✓
quadratic B-spline < quartic B-spline	100	65	3	3586.5	6.68×10^{-6}	✓
cubic B-spline < quartic B-spline	100	58	5	2952.5	6.27×10^{-3}	✓

Table 6: Wilcoxon signed-rank test for paired per-phantom improvements in best PSNR from lower- to higher-order B-spline encodings at $\log_2 T = 17$ (Lung dataset, 60 views, significance level=0.05).

H_1: **lower < higher**	n	**wins**	**ties**	**stat**	p	**sig**
constant B-spline < linear B-spline	100	99	0	5047.5	2.10×10^{-18}	✓
constant B-spline < quadratic B-spline	100	100	0	5050.0	1.95×10^{-18}	✓
constant B-spline < cubic B-spline	100	100	0	5050.0	1.95×10^{-18}	✓
constant B-spline < quartic B-spline	100	100	0	5050.0	1.95×10^{-18}	✓
linear B-spline < quadratic B-spline	100	95	0	5006.5	7.16×10^{-18}	✓
linear B-spline < cubic B-spline	100	97	0	5015.0	5.56×10^{-18}	✓
linear B-spline < quartic B-spline	100	95	0	5014.0	5.73×10^{-18}	✓
quadratic B-spline < cubic B-spline	100	68	5	3613.5	3.70×10^{-7}	✓
quadratic B-spline < quartic B-spline	100	69	0	3834.0	3.38×10^{-6}	✓
cubic B-spline < quartic B-spline	100	54	2	2762.5	1.16×10^{-1}	×

Table 7: Wilcoxon signed-rank test for paired per-phantom improvements in best PSNR from lower- to higher-order B-spline encodings at $\log_2 T = 17$ (Lung dataset, 40 views, significance level=0.05)

H_1: **lower < higher**	n	**wins**	**ties**	**stat**	p	**sig**
constant B-spline < linear B-spline	100	100	0	5050.0	1.94×10^{-18}	✓
constant B-spline < quadratic B-spline	100	100	0	5050.0	1.95×10^{-18}	✓
constant B-spline < cubic B-spline	100	100	0	5050.0	1.95×10^{-18}	✓
constant B-spline < quartic B-spline	100	100	0	5050.0	1.95×10^{-18}	✓
linear B-spline < quadratic B-spline	100	97	1	4944.0	3.41×10^{-18}	✓
linear B-spline < cubic B-spline	100	99	0	5048.0	2.06×10^{-18}	✓
linear B-spline < quartic B-spline	100	97	1	4945.0	3.31×10^{-18}	✓
quadratic B-spline < cubic B-spline	100	78	3	4154.5	7.84×10^{-11}	✓
quadratic B-spline < quartic B-spline	100	80	3	4236.0	1.10×10^{-11}	✓
cubic B-spline < quartic B-spline	100	65	2	3450.5	1.40×10^{-4}	✓

Table 8: Wilcoxon signed-rank test for paired per-phantom improvements in best PSNR from lower- to higher-order B-spline encodings at $\log_2 T = 17$ (Lung dataset, 20 views, significance level=0.05).

H_1: **lower < higher**	n	**wins**	**ties**	**stat**	p	**sig**
constant B-spline < linear B-spline	100	97	0	5028.0	3.77×10^{-18}	✓
constant B-spline < quadratic B-spline	100	100	0	5050.0	1.94×10^{-18}	✓
constant B-spline < cubic B-spline	100	100	0	5050.0	1.95×10^{-18}	✓
constant B-spline < quartic B-spline	100	100	0	5050.0	1.94×10^{-18}	✓
linear B-spline < quadratic B-spline	100	96	2	4732.5	1.47×10^{-16}	✓
linear B-spline < cubic B-spline	100	98	0	5027.0	3.88×10^{-18}	✓
linear B-spline < quartic B-spline	100	99	0	5049.0	2.00×10^{-18}	✓
quadratic B-spline < cubic B-spline	100	84	4	4382.0	3.02×10^{-14}	✓
quadratic B-spline < quartic B-spline	100	90	3	4639.0	1.95×10^{-16}	✓
cubic B-spline < quartic B-spline	100	74	5	3986.5	1.17×10^{-10}	✓

A.13 Wilcoxon signed-rank test in the Liver dataset

This section is supplementary to the sec. 5.2 in the main text.

The Wilcoxon signed-rank tests for the Liver dataset are shown in tables 9 to 12.

Table 9: Wilcoxon signed-rank test for paired per-phantom improvements in best PSNR from lower- to higher-order B-spline encodings at $\log_2 T = 17$ (Liver dataset, 80 views, significance level=0.05).

H_1: **lower < higher**	n	**wins**	**ties**	**stat**	p	**sig**
constant B-spline < linear B-spline	40	40	0	820.0	1.78×10^{-8}	✓
constant B-spline < quadratic B-spline	40	40	0	820.0	1.78×10^{-8}	✓
constant B-spline < cubic B-spline	40	40	0	820.0	1.78×10^{-8}	✓
constant B-spline < quartic B-spline	40	40	0	820.0	1.78×10^{-8}	✓
linear B-spline < quadratic B-spline	40	38	0	808.0	4.39×10^{-8}	✓
linear B-spline < cubic B-spline	40	40	0	820.0	1.77×10^{-8}	✓
linear B-spline < quartic B-spline	40	40	0	820.0	1.78×10^{-8}	✓
quadratic B-spline < cubic B-spline	40	33	0	723.5	1.25×10^{-5}	✓
quadratic B-spline < quartic B-spline	40	32	0	761.5	1.15×10^{-6}	✓
cubic B-spline < quartic B-spline	40	23	1	417.5	3.50×10^{-1}	×

Table 10: Wilcoxon signed-rank test for paired per-phantom improvements in best PSNR from lower- to higher-order B-spline encodings at $\log_2 T = 17$ (Liver dataset, 60 views, significance level=0.05).

H_1: lower < higher	n	wins	ties	stat	p	sig
constant B-spline < linear B-spline	40	40	0	820.0	1.78×10^{-8}	✓
constant B-spline < quadratic B-spline	40	40	0	820.0	1.78×10^{-8}	✓
constant B-spline < cubic B-spline	40	40	0	820.0	1.78×10^{-8}	✓
constant B-spline < quartic B-spline	40	40	0	820.0	1.78×10^{-8}	✓
linear B-spline < quadratic B-spline	40	40	0	820.0	1.78×10^{-8}	✓
linear B-spline < cubic B-spline	40	40	0	820.0	1.78×10^{-8}	✓
linear B-spline < quartic B-spline	40	40	0	820.0	1.78×10^{-8}	✓
quadratic B-spline < cubic B-spline	40	31	1	701.5	6.87×10^{-6}	✓
quadratic B-spline < quartic B-spline	40	33	1	742.5	4.32×10^{-7}	✓
cubic B-spline < quartic B-spline	40	25	1	552.5	1.17×10^{-2}	✓

Table 11: Wilcoxon signed-rank test for paired per-phantom improvements in best PSNR from lower- to higher-order B-spline encodings at $\log_2 T = 17$ (Liver dataset, 40 views, significance level=0.05).

H_1: lower < higher	n	wins	ties	stat	p	sig
constant B-spline < linear B-spline	40	40	0	820.0	1.78×10^{-8}	✓
constant B-spline < quadratic B-spline	40	40	0	820.0	1.78×10^{-8}	✓
constant B-spline < cubic B-spline	40	40	0	820.0	1.78×10^{-8}	✓
constant B-spline < quartic B-spline	40	40	0	820.0	1.78×10^{-8}	✓
linear B-spline < quadratic B-spline	40	40	0	820.0	1.78×10^{-8}	✓
linear B-spline < cubic B-spline	40	40	0	820.0	1.78×10^{-8}	✓
linear B-spline < quartic B-spline	40	40	0	820.0	1.78×10^{-8}	✓
quadratic B-spline < cubic B-spline	40	36	0	773.5	5.13×10^{-7}	✓
quadratic B-spline < quartic B-spline	40	36	0	794.0	1.22×10^{-7}	✓
cubic B-spline < quartic B-spline	40	31	0	673.0	2.03×10^{-4}	✓

Table 12: Wilcoxon signed-rank test for paired per-phantom improvements in best PSNR from lower- to higher-order B-spline encodings at $\log_2 T = 17$ (Liver dataset, 20 views, significance level=0.05).

H_1: lower < higher	n	wins	ties	stat	p	sig
constant B-spline < linear B-spline	40	40	0	820.0	1.77×10^{-8}	✓
constant B-spline < quadratic B-spline	40	40	0	820.0	1.78×10^{-8}	✓
constant B-spline < cubic B-spline	40	40	0	820.0	1.77×10^{-8}	✓
constant B-spline < quartic B-spline	40	40	0	820.0	1.78×10^{-8}	✓
linear B-spline < quadratic B-spline	40	39	0	815.5	2.48×10^{-8}	✓
linear B-spline < cubic B-spline	40	39	0	818.0	2.07×10^{-8}	✓
linear B-spline < quartic B-spline	40	40	0	820.0	1.77×10^{-8}	✓
quadratic B-spline < cubic B-spline	40	35	1	726.0	1.36×10^{-6}	✓
quadratic B-spline < quartic B-spline	40	35	1	759.0	1.30×10^{-7}	✓
cubic B-spline < quartic B-spline	40	28	2	601.0	4.13×10^{-4}	✓

A.14 Average reconstruction accuracy per epoch

This section is supplementary to the sec. 5.2 in the main text. The average reconstruction accuracy per epoch are shown in Fig. 14.

A.15 Visual reconstruction results on the liver dataset

This section is supplementary to the sec. 5.2 in the main text. The visual reconstruction are shown in Fig. 15 and Fig. 16.

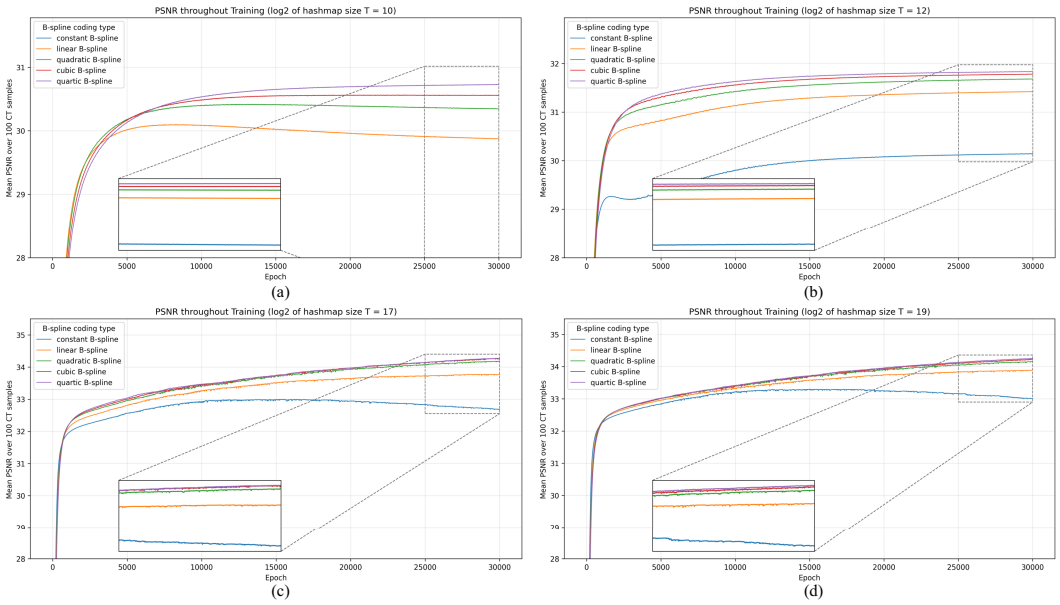


Figure 14: (a-d) Avg. accuracy in 80 views of each method per epoch when $T = 2^{10}, 2^{12}, 2^{17}, 2^{19}$.

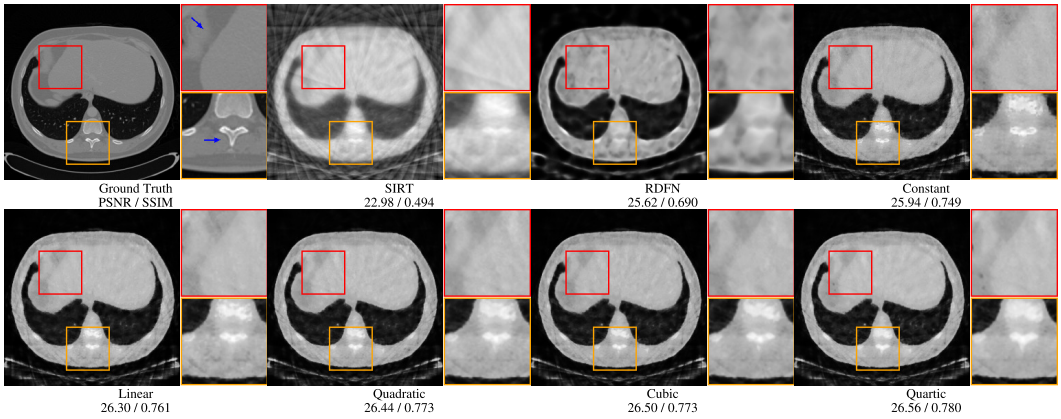


Figure 15: Results in 20 views (network depth $= 3$). B-spline use hashing with $\log_2 T = 17$.

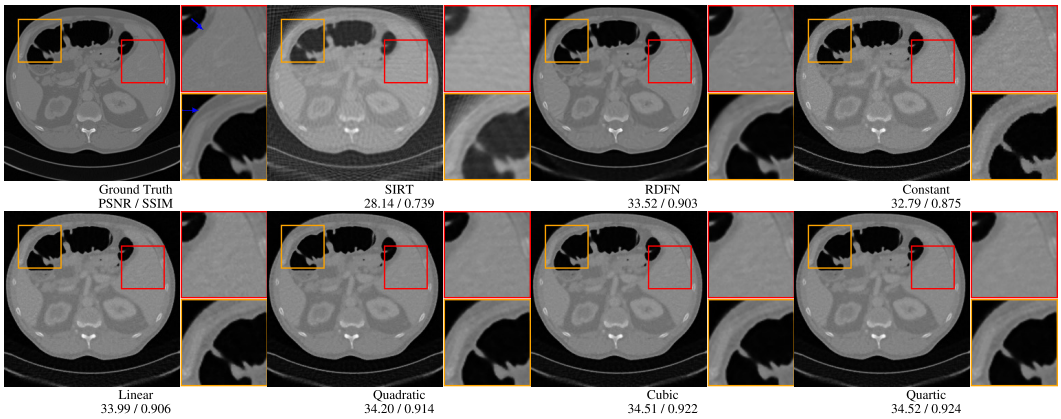


Figure 16: Results in 60 views (network depth $= 3$). B-spline use hashing with $\log_2 T = 17$.

A.16 PROOF OF THE APPROXIMATION ERROR

This section is supplementary to the sec. 4 in the main text.

For any given coordinate x, we denote:

$$\mathbf{y}^{(r)}(x) := \big(y_1^{(r)}(x), \ldots, y_L^{(r)}(x)\big), \qquad \mathbf{y}^*(x) := \big(y_1^*(x), \ldots, y_L^*(x)\big).$$

By the multivariable mean value theorem, there exist $\theta \in (0,1)$, such that

$$F(\mathbf{y}^{(r)}(x)) - F(\mathbf{y}^*(x)) = \nabla F(\widetilde{\mathbf{y}}(x))^\top (\mathbf{y}^{(r)}(x) - \mathbf{y}^*(x)) \tag{38}$$

where $\widetilde{\mathbf{y}}(x) = \mathbf{y}^*(x) + \theta\left(\mathbf{y}^{(r)}(x) - \mathbf{y}^*(x)\right)$.

According to the Cauchy–Schwarz inequality, we have:

$$\left|\nabla F(\widetilde{\mathbf{y}}(x))^\top \left(\mathbf{y}^{(r)}(x) - \mathbf{y}^*(x)\right)\right| \le \|\nabla F(\widetilde{\mathbf{y}}(x))\|_2 \left\|\mathbf{y}^{(r)}(x) - \mathbf{y}^*(x)\right\|_2. \tag{39}$$

Assume $\sup_{u\in\Lambda} \|\nabla F(u)\| < \infty$, where Λ denotes the domain of F. Then we obtain:

$$\left|\nabla F(\widetilde{\mathbf{y}}(x))^\top \left(\mathbf{y}^{(r)}(x) - \mathbf{y}^*(x)\right)\right| \le \left(\sup_{\mathbf{y},x} \|\nabla F(\mathbf{y}(x))\|_2\right) \|\mathbf{y}^{(r)}(x) - \mathbf{y}^*(x)\|_2. \tag{40}$$

We denote $\sup_{\mathbf{y},x} \|\nabla F(\mathbf{y}(x))\|_2$ as L_F, considering eq. (38), we can obtain:

$$\left|F(\mathbf{y}^{(r)}(x)) - F(\mathbf{y}^*(x))\right| \le L_F \|\mathbf{y}^{(r)}(x) - \mathbf{y}^*(x)\|_2. \tag{41}$$

Let Ω be a continuous coordinate space. Then, we have:

$$\int_{x\in\Omega} \left|F\left(\mathbf{y}^{(r)}(x)\right) - F(\mathbf{y}^*(x))\right| dx \le L_F \int_{x\in\Omega} \left\|\mathbf{y}^{(r)}(x) - \mathbf{y}^*(x)\right\|_2 dx. \tag{42}$$

Using the triangle inequality for L dimension vectors, we have:

$$\left\|\mathbf{y}^{(r)}(x) - \mathbf{y}^*(x)\right\|_2 \le \sum_{l=1}^{L} \left\|y_l^{(r)}(x) - y_l^*(x)\right\|_2 \tag{43}$$

Therefore, we have:

$$\int_{x\in\Omega} \left\|\mathbf{y}^{(r)}(x) - \mathbf{y}^*(\mathbf{x})\right\|_2 dx \le \int_{x\in\Omega} \sum_{l=1}^{L} \left\|y_l^{(r)}(x) - y_l^*(x)\right\|_2 dx \tag{44}$$

Substitute the inequality eq. (44) into inequality eq. (42), we have:

$$\int_{x\in\Omega} \left|F\left(\mathbf{y}^{(r)}(x)\right) - F(\mathbf{y}^*(x))\right| dx \le L_F \int_{x\in\Omega} \sum_{l=1}^{L} \left\|y_l^{(r)}(x) - y_l^*(x)\right\|_2 dx. \tag{45}$$

So, we have:

$$\inf_{\mathbf{y}^{(r)}\in S^{(r)}} \int_{x\in\Omega} \left|F\left(\mathbf{y}^{(r)}(x)\right) - F(\mathbf{y}^*(x))\right| dx \le L_F \inf_{\mathbf{y}^{(r)}\in S^{(r)}} \int_{x\in\Omega} \sum_{l=1}^{L} \left\|y_l^{(r)}(x) - y_l^*(x)\right\|_2 dx. \tag{46}$$

where:

$$S^{(r)} := \left\{ \left[\sum_{k\in\mathcal{I}} a_{1,k} B^{(r)}_{k,\frac{1}{N_1}}, \ldots, \sum_{k\in\mathcal{I}} a_{L,k} B^{(r)}_{k,\frac{1}{N_L}}\right]^\top : a_{l.k} \in \mathbb{R} \right\}. \tag{47}$$

Here, $B^{(r)}_{k,\frac{1}{N_l}}$ denotes the k-th B-spline of order r in a grid resolution of $N_l (l = 1, 2, ..., L)$.

Because y_l^* is defined on the entire coordinate space, we assume that the smoothness of each component y_i^* of y^* is s_l,$(l = 1, 2..., L)$. According to the Jackson-type theorem for splines(De Boor & De Boor, 1978), we have:

$$\inf_{\mathbf{y}_l^{(r)} \in S_l^{(r)}} \int_{x\in\Omega} \left\| y_l^{(r)}(x) - y_l^*(x) \right\|_2 dx. \leq C_l \left(\frac{1}{N_l} \right)^{\min(r,s_l)} \tag{48}$$

where C_l is a constant independent of N_l, the $S_l^{(r)}$ is defined as:

$$S_l^{(r)} := \left\{ \sum_{k\in\mathcal{I}} a_{l.k} B^{(r)}_{k,\frac{1}{N_l}} : a_{l.k} \in \mathbb{R} \right\} \tag{49}$$

Because:

$$\inf_{\mathbf{y}^{(r)} \in S^{(r)}} \int_{x\in\Omega} \sum_{l=1}^{L} \left\| y_l^{(r)}(x) - y_l^*(x) \right\|_2 dx = \inf_{\mathbf{y}^{(r)} \in S^{(r)}} \sum_{l=1}^{L} \int_{x\in\Omega} \left\| y_l^{(r)}(x) - y_l^*(x) \right\|_2 dx \tag{50}$$

$$\inf_{\mathbf{y}^{(r)} \in S^{(r)}} \sum_{l=1}^{L} \int_{x\in\Omega} \left\| y_l^{(r)}(x) - y_l^*(x) \right\|_2 dx \leq \sum_{l=1}^{L} \left(\inf_{\mathbf{y}_l^{(r)} \in S_l^{(r)}} \int_{x\in\Omega} \left\| y_l^{(r)}(x) - y_l^*(x) \right\|_2 dx \right) \tag{51}$$

Combined with eqs. (48), (50) and (51), we can get:

$$\inf_{\mathbf{y}^{(r)} \in S^{(r)}} \int_{x\in\Omega} \sum_{l=1}^{L} \left\| y_l^{(r)}(x) - y_l^*(x) \right\|_2 dx \leq \sum_{l=1}^{L} C_l \left(\frac{1}{N_l} \right)^{\min(r,s_l)} \tag{52}$$

Therefore, the eq. (46) can be written as:

$$\inf_{\mathbf{y}^{(r)} \in S^{(r)}} \int_{x\in\Omega} \left| F\left(\mathbf{y}^{(r)}(x) \right) - F(\mathbf{y}^*(x)) \right| dx \leq L_F \sum_{l=1}^{L} C_l \left(\frac{1}{N_l} \right)^{\min(r,s_l)} \tag{53}$$

We denote $s := \min_{1\leq i\leq L} s_l$ and substitute the original definitions of the variables back in eq. (53). we have:

$$\inf_{\mathbf{y}^{(r)} \in S^{(r)}} \int_{x\in\Omega} \left| F\left(\mathbf{y}^{(r)}(x) \right) - F(\mathbf{y}^*(x)) \right| dx \leq L_F \sum_{l=1}^{L} C_l \left(\frac{1}{N_l} \right)^{\min(r,s)} \tag{54}$$

We use the symbol E to represent the capability for optimal approximation, therefore, eq. (54) is equivalent to:

$$E(F(y_1^{(r)}, ..., y_L^{(r)}), f^*) \leq L_F \sum_{l=1}^{L} C_l \left(\frac{1}{N_l} \right)^{\min(r,s)} \tag{55}$$

Furthermore, in the more general setting where the Taylor-truncated activation function is not used, the neural network can no longer be represented as a polynomial. Nevertheless, by the Lipschitz continuity of the neural network (Virmaux & Scaman, 2018), we can replace L_F in eq. (41) with the Lipschitz constant L_p of the neural network. Therefore, we obtain:

$$E(f_{network}(y_1^{(r)}, ..., y_L^{(r)}), f^*) \leq L_P \sum_{l=1}^{L} C_l \left(\frac{1}{N_l} \right)^{\min(r,s)} \tag{56}$$

A.17 AI USE STATEMENT

We used large language models (LLMs) to assist with manuscript polishing, formatting, and literature review. LLMs were also used to assist with code modifications and improvements, as well as data organization. All LLM-assisted content was rigorously reviewed and verified by us.