# Low-Rank Tensor Function Representation for Multi-Dimensional Data Recovery

Yisi Luo, Xile Zhao, *Member, IEEE*, Zhemin Li, Michael K. Ng, *Senior Member, IEEE*, Deyu Meng, *Member, IEEE*

**Abstract**—Since higher-order tensors are naturally suitable for representing multi-dimensional data in real-world, e.g., color images and videos, low-rank tensor representation has become one of the emerging areas in machine learning and computer vision. However, classical low-rank tensor representations can solely represent multi-dimensional discrete data on meshgrid, which hinders their potential applicability in many scenarios beyond meshgrid. To break this barrier, we propose a low-rank tensor function representation (LRTFR) parameterized by multilayer perceptrons (MLPs), which can continuously represent data beyond meshgrid with powerful representation abilities. Specifically, the suggested tensor function, which maps an arbitrary coordinate to the corresponding value, can continuously represent data in an infinite real space. Parallel to discrete tensors, we develop two fundamental concepts for tensor functions, i.e., the tensor function rank and low-rank tensor function factorization, and utilize MLPs to paramterize factor functions of the tensor function factorization. We theoretically justify that both low-rank and smooth regularizations are harmoniously unified in LRTFR, which leads to high effectiveness and efficiency for data continuous representation. Extensive multi-dimensional data recovery applications arising from image processing (image inpainting and denoising), machine learning (hyperparameter optimization), and computer graphics (point cloud upsampling) substantiate the superiority and versatility of our method as compared with state-of-the-art methods. Especially, the experiments beyond the original meshgrid resolution (hyperparameter optimization) or even beyond meshgrid (point cloud upsampling) validate the favorable performances of our method for continuous representation.

**Index Terms**—Tensor factorization, multi-dimensional data, data recovery

## 1 Introduction

Recently, due to the advance of technology, various types of multi-dimensional data (e.g., color images, multispectral images, point clouds, traffic flow data, user-item data, etc.) are increasingly emerging. Mathematically, higher-order tensors are naturally suitable for multi-dimensional data modeling and processing, which is one of the focus areas in machine learning, computer vision, and scientific computing [1], [2], [3], [4].

Most real-world data intrinsically exhibits low-dimensional structures, for example, images [5], videos [6], point clouds [7], and so on. Hence, low-rank modeling of matrices/tensors has been widely studied for data processing and representation [8], [9], [10]. Different from matrices, the rank definition of higher-order tensors is not unique. The most classic tensor ranks are the Tucker rank (defined by the rank of unfolding matrices) [11] and CANDECOMP/PARAFAC (CP) rank (defined as the smallest number of rank one tensor decomposition) [11]. It is shown that solving the low-Tucker/CP-rank programming can obtain effective low-rank representations of multi-dimensional data [12], [13]. Meanwhile, the low-Tucker/CP-rank models have been applied to facilitate the efficiency of modern deep learning algorithms [14], [15], which reveals their wide and promising applicabilities. Another type of tensor rank is based on the so-called tensor network decomposition [16], such as tensor train rank [17], tensor ring rank [18], and tensor tree rank [19]. These more sophisticated tensor ranks were proven to be highly representative for multi-dimensional modeling; see [6], [20], [21]. More recently, the tensor tubal-rank [22], which is based on the tensor singular value decomposition (t-SVD) [23], has attracted much attention due to its strong relationships to the definition of matrix rank; see for example [24], [25], [26], [27]. Besides, the effctiveness of tubal-rank minimization for various signal processing tasks has been validated [28], [29], [30]. In summary, low-rank tensor modeling has become increasingly popular for multi-dimensional data representation and processing.

Except for the low-rankness, smoothness is another frequently considered regularization in multi-dimensional data representation. Many literatures incorporated the smoothness into the low-rank representation [13], [31], [32]. In such a hybrid model, low-rankness was usually revealed by low-rank factorization [25], [33] or surrogate functions [34], [35], [36]. Besides, there are two main categories of smooth regularization forms, i.e., explicit and implicit ones. The explicit smooth regularizations are mainly based on the total variation (TV) and its variants [13], [31], [32], [37]. The implicit smooth regularizations are revealed by using

*This research is supported by the NSFC (No. 12371456, 12171072, 62131005), the National Key Research and Development Program of China (No. 2020YFA0714001), and the Open Research Fund Program of Data Recovery Key Laboratory of Sichuan Province (Grant No. DRN2302).*

- *Yisi Luo and Deyu Meng are with the School of Mathematics and Statistics, Xi'an Jiaotong University, Xi'an, P.R.China (e-mail: yisiluo1221@foxmail.com, dymeng@mail.xjtu.edu.cn).*
- *Xile Zhao is with the School of Mathematical Sciences, University of Electronic Science and Technology of China, Chengdu, P.R.China (e-mail: xlzhao122003@163.com).*
- *Zhemin Li is with the Department of Mathematics, National University of Defense Technology, Changsha, P.R.China (e-mail: lizhemin@nudt.edu.cn).*
- *Michael K. Ng is with the Department of Mathematics, The University of Hong Kong, Hong Kong (e-mail: mng@maths.hku.hk).*

*(Corresponding authors: Xile Zhao and Deyu Meng)*

This article has been accepted for publication in IEEE Transactions on Pattern Analysis and Machine Intelligence. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2023.3341688



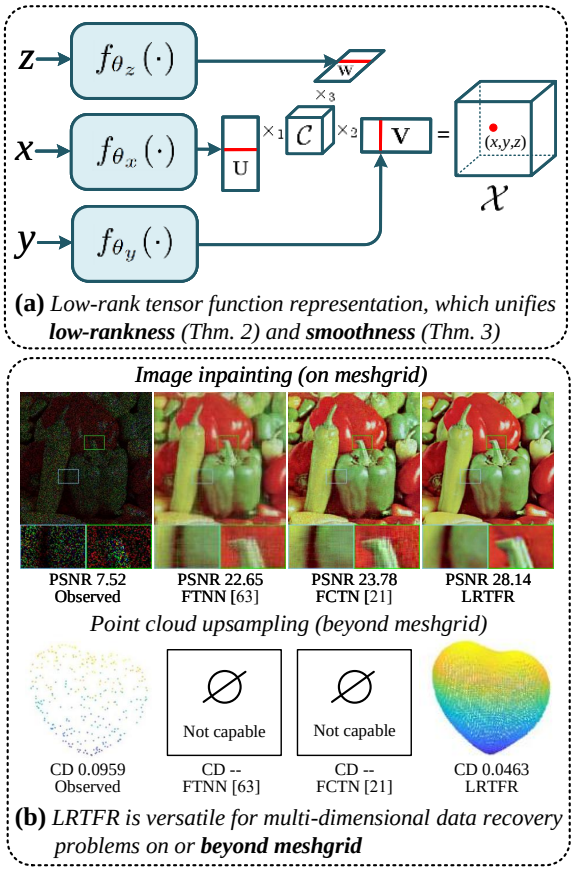



Fig. 1. (a) The proposed LRTFR can continuously represent multi-dimensional data via the low-rank tensor function factorization, which intrinsically unifies the low-rankness and smoothness into the representation. (b) The continuous LRTFR is versatile for general multi-dimensional data recovery problems on or beyond meshgrid.

basis functions to parameterize the model, which naturally extends the model to function representations that are related to this work. For example, the pioneer works [38], [39] utilized non-negative matrix factorization parameterized by basis functions to reveal the implicit smoothness. It was further extended to higher-order tensor models [34] by using the low-Tucker-rank regularization. More recently, the CP factorization was elegantly generalized to multivariate functions by using Fourier series [40]. The main aim of these implicit smooth representations is to employ some basis functions (e.g., Gaussian basis [38] or Fourier basis [40]) to represent the low-rank matrix/tensor, which implicitly induces the smoothness between adjacent elements of the matrix/tensor. Nevertheless, these shallow basis functions may sometimes not be suitable to capture the complex fine details of real-world data. Meanwhile, these basis function-based representations still rely on the low-rank regularization defined on original meshgrid to handle plain data analysis [38] or regression problems [40], which may not be applicable to more challenging multi-dimensional data recovery problems on or beyond meshgrid.

To sum up, low-rank tensor representations are suitable for representing data on discrete meshgrid. However, how to extend low-rank tensor models for continuous data representation beyond meshgrid is a pressing challenge driven by real-world applications. For example, point cloud representation, where classical low-rank representations can not represent such signals beyond meshgrid. An important question naturally arises: Can we develop a multi-dimensional data representation that can not only preserve the low-rank structure, but also continuously represent data beyond meshgrid with infinite resolution?

To meet this challenge, this work presents the low-rank tensor function representation (LRTFR) for multi-dimensional data continuous representation. Specifically, we consider representing data with a tensor function, which maps a multi-dimensional coordinate to the corresponding value to continuously represent data. Parallel to classical low-rank tensor representations, we develop two fundamental concepts for tensor functions, i.e., the tensor function rank (Definition 3) and low-rank tensor function factorization (Theorem 2), which intrinsically encodes the low-rankness into the continuous representation. We use multilayer perceptrons (MLPs) to parameterize the factor functions of the tensor function factorization, which makes the model highly expressive for real-world data representation. Moreover, we theoretically justify a Lipschitz smooth regularization hidden in the representation. The low-rankness and smoothness are harmoniously unified in our LRTFR, making it effective and efficient for data continuous representation. Compared to classical low-rank tensor representations, LRTFR is more versatile for representing real-world discrete data on or beyond meshgrid; see Fig. 1. In summary, this work makes the following contributions:

**(i)** To continuously represent multi-dimensional data on or beyond meshgrid, we suggest the low-rank tensor function representation parameterized by MLPs. Thanks to the continuous nature of function representation and the expressive power of MLPs, our LRTFR is effective and powerful to represent various real-world multi-dimensional data on or beyond meshgrid.

**(ii)** We suggest two concepts of tensor functions, i.e., the tensor function rank and low-rank tensor function factorization, which establish connections between discrete and continuous representations for multi-dimensional data. We theoretically justify that the low-rank and smooth regularizations are harmoniously unified in the MLP-parameterized LRTFR, which reveals its potential effectiveness for multi-dimensional data recovery.

**(iii)** The proposed LRTFR is versatile for multiple multi-dimensional data recovery tasks on or beyond meshgrid, including multi-dimensional image inpainting and denoising (on meshgrid with original resolution), hyperparameter optimization (on meshgrid beyond original resolution), and point cloud upsampling (beyond meshgrid). Extensive experiments validate the broad applicability and superiority of our method as compared with state-of-the-art methods.

## 2 Related Work

### 2.1 Data Continuous Representation

The continuous representation of data has recently attracted considerable attention [41], [42], [43]. One of the most famous techniques for continuous data representation is the implicit neural representation (INR) [44]. The INR constructs differentiable functions (deep neural networks) w.r.t. the coordinates (inputs) to implicitly represent the continuous data (outputs). The implicit representation is obtained

by feeding each coordinate to the implicit function and outputting the corresponding value. Since the implicit function is defined on a continuous domain, the resulting data representation is also continuous. The success of INR-based methods for different tasks has been witnessed, e.g., neural rendering [45], [46], [47], image/shape generation [48], [49], and scene representation [50]. Besides, some important improvements over INR (e.g., more effective training strategy [46], [51] and more expressive INR structures [52], [53]) have also been studied recently.

Despite these great efforts, INR still faces two challenges. First, INR requires relatively large computational cost, mainly because the size of the input coordinate matrix is too large (See Sec. 3.4 for detailed computational analyses). The computational cost tremendously increases when dealing with multi-dimensional data, e.g., multispectral images and videos. Second, INR itself may not be stable enough to directly learn a valid continuous representation from raw data, which usually results in overfitting and restricts its applicability in real-world scenarios. The underlying reason of the two limitations is that INR ignores the important domain knowledge of data. As compared, our LRTFR disentangles the continuous representation into several simpler factor representations by introducing insightful domain knowledge (i.e., low-rankness); see Fig. 1 (a). Consequently, LRTFR achieves lower computational costs and increases the stability of the continuous representation against overfitting; see Sec. 3.4 for details.

### 2.2 Data Recovery via Continuous Representation

Several works have studied INRs for data recovery problems, which are related to our work. The local implicit image function [41] was proposed for image super-resolution. Its improved versions in terms of network capacity [54] and details recovery [55] were proposed to learn more realistic image continuous representations. The meta-learning strategies [44], [56] were also developed to learn mappings that generate INRs for image recovery. However, these image recovery methods were entirely dependent on supervised learning with pairs of images to train the INR. Comparatively, our LRTFR is a model-based and unsupervised method that finely encodes the low-rankness into the continuous representation. Hence, our method can be more easily and conveniently generalized over different applications; see Sec. 3.5. A very recent work [57] proposed to use INR for zero-shot blind image denoising, where decent denoising results were obtained. However, its accompanied theoretical explanations for image denoising are lacking, and it possesses high computational costs since it uses the standard INR. In comparison, our LRTFR implicitly and theoretically encodes the low-rankness into the continuous representation via the tensor function factorization, which is more efficient with clearer interpretations for data recovery.

### 2.3 Multivariate Function Representation

In the literature, there are some pioneer works [58], [59] that construct multivariate function representations in low-rank tensor formats from the function approximation perspective. Specifically, Oseledets [58] introduced explicit representations of some specific multivariate functions (e.g., the polynomial and sine functions) in tensor-train format. Hashemi and Trefethen [59] studied Tucker factorization of trivariate functions and proposed constructive algorithms for approximating trivariate functions in Tucker format, where the factor one-dimensional functions are represented by finite sum of Chebyshev functions.

The differences between our work and the methods proposed in [58], [59] are as follows. Both [58] and [59] aim to obtain the function approximation of the given multivariate function in tensor factorization format, where the factor functions in the tensor function factorization are parameterized by finite sum of basis functions (e.g., Chebyshev functions used in [59]). Comparatively, our method aims to obtain the low-rank continuous representation of the given multi-dimensional discrete data on or beyond meshgrid, where the factor functions in the tensor function factorization are parameterized by deep neural networks, i.e., MLPs. Hence, our method should be more suitable for representing complex discrete multi-dimensional data on or beyond meshgrid, such as multispectral images and point clouds, while the methods in [58], [59] are applicable for continuous function representation, and may not be applicable for such complex discrete data representation. Moreover, we theoretically establish the connections between the low-rank continuous representation and discrete low-rank representations. We justify the low-rankness and smoothness underlying the suggested LRTFR, which reveals its potential effectiveness for multi-dimensional data recovery. Our extensive experiments have also support such superiority of the our method in representing complex multi-dimentional discrete data, while such evaluation can hardly be executed for methods of [58], [59].

## 3 The Proposed Method

### 3.1 Preliminaries

Some frequently used notations in this paper are summarized in Table 1. In addition, we introduce the tensor Tucker rank and Tucker factorization as below.

**Definition 1.** *(Tucker rank [11]) The Tucker rank of a tensor $\mathcal{X} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$ is a vector defined as*

$$\operatorname{rank}_T(\mathcal{X}) := (\operatorname{rank}(\mathbf{X}^{(1)}), \operatorname{rank}(\mathbf{X}^{(2)}), \operatorname{rank}(\mathbf{X}^{(3)})), \quad (1)$$

*where $\mathbf{X}^{(i)}$ $(i = 1, 2, 3)$ denotes the mode-$i$ unfolding matrix of $\mathcal{X}$ and $\operatorname{rank}(\cdot)$ denotes the matrix rank. For simplicity, we sometimes use the notation $(\operatorname{rank}_T(\mathcal{X}))_{(i)} := \operatorname{rank}(\mathbf{X}^{(i)})$.*

**Theorem 1.** *(Tucker factorization [11]) Let $\mathcal{X} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$.*

- **(i)** *If $\operatorname{rank}_T(\mathcal{X}) = (r_1, r_2, r_3)$, then there exists a core tensor $\mathcal{C} \in \mathbb{R}^{r_1 \times r_2 \times r_3}$ and three factor matrices $\mathbf{U} \in \mathbb{R}^{n_1 \times r_1}$, $\mathbf{V} \in \mathbb{R}^{n_2 \times r_2}$, and $\mathbf{W} \in \mathbb{R}^{n_3 \times r_3}$ such that $\mathcal{X} = \mathcal{C} \times_1 \mathbf{U} \times_2 \mathbf{V} \times_3 \mathbf{W}$.*
- **(ii)** *Let $\mathcal{C} \in \mathbb{R}^{r_1 \times r_2 \times r_3}$ be an arbitrary tensor, $\mathbf{U} \in \mathbb{R}^{n_1 \times r_1}$, $\mathbf{V} \in \mathbb{R}^{n_2 \times r_2}$, $\mathbf{W} \in \mathbb{R}^{n_3 \times r_3}$ be arbitrary matrices ($r_i \leq n_i$ for $i = 1, 2, 3$). Then $\big(\operatorname{rank}_T(\mathcal{C} \times_1 \mathbf{U} \times_2 \mathbf{V} \times_3 \mathbf{W})\big)_{(i)} \leq r_i$ $(i = 1, 2, 3)$.*

### 3.2 Low-Rank Tensor Function Representation

In this section, we detailedly introduce the proposed LRTFR for multi-dimensional data. Without loss of generality, we

TABLE 1
Notations used in this paper.

| Notation | Description |
|---|---|
| $x, \mathbf{x}, \mathbf{X}, \mathcal{X}$ | Scalar, vector, matrix, tensor |
| $\mathbf{x}_{(i)}$ | The $i$-th element of $\mathbf{x}$ |
| $\mathbf{X}_{(i,j)}$ | The $(i,j)$-th element of $\mathbf{X}$ |
| $\mathcal{X}_{(i,j,k)}$ | The $(i,j,k)$-th element of $\mathcal{X}$ |
| $\mathbf{X}_{(i,:)}/\mathbf{X}_{(:,j)}$ | The $i$-th row/$j$-th column of $\mathbf{X}$ |
| $\|\mathcal{X}\|_F$ | The tensor Frobenius norm $\|\mathcal{X}\|_F := \sqrt{\langle \mathcal{X}, \mathcal{X}\rangle} = \sqrt{\sum_{ijk} \mathcal{X}^2_{(i,j,k)}}$ |
| $\|\mathcal{X}\|_{\ell_1}$ | The tensor $\ell_1$-norm $\|\mathcal{X}\|_{\ell_1} := \sum_{ijk} \|\mathcal{X}_{(i,j,k)}\|$ |
| $\texttt{unfold}_i(\cdot)$ $(i = 1,2,3)$ | The unfolding operator along the $i$-th mode $\texttt{unfold}_i(\cdot) : \mathbb{R}^{n_1 \times n_2 \times n_3} \to \mathbb{R}^{n_i \times (\prod_{j \neq i} n_j)}$ |
| $\mathbf{X}^{(i)}$ | The unfolding matrix $\mathbf{X}^{(i)} := \texttt{unfold}_i(\mathcal{X})$ |
| $\texttt{fold}_i(\cdot)$ | The inverse operator of $\texttt{unfold}_i(\cdot)$ |
| $\times_i$ | The mode-$i$ tensor-matrix product $\mathcal{X} \times_i \mathbf{A} := \texttt{fold}_i(\mathbf{A}\mathbf{X}^{(i)})$ |
| $\text{rank}(\mathbf{X})$ | The matrix rank of $\mathbf{X}$ |
| $\lfloor x \rfloor$ | The rounded-down of a constant $x$ |

consider the three-dimensional case, while it can be easily generalized to higher-dimensional cases. Let $f(\cdot) : X_f \times Y_f \times Z_f \to \mathbb{R}$ be a bounded real function, where $X_f, Y_f, Z_f \subset \mathbb{R}$ are definition domains in three dimensions. The function $f(\cdot)$ gives the value of data at any coordinate in $D_f := X_f \times Y_f \times Z_f$. We interpret $f(\cdot)$ as a *tensor function* since it maps a three-dimensional coordinate to the corresponding value, implicitly representing third-order tensor data. Compared with classical tensor formulation, the tensor function intrinsically allows us to process and analyse multi-dimensional data on meshgrid beyond the original resolution or even beyond meshgrid. When $D_f$ is a discrete set of some constants, the output form of $f(\cdot)$ degrades to the discrete case (i.e., tensors).

Based on tensor functions, we can naturally define the following sampled tensor set, which covers all tensors that can be sampled from the tensor function with different sampling coordinates.

**Definition 2.** *For a tensor function $f(\cdot) : D_f \to \mathbb{R}$, where $D_f \subset \mathbb{R}^3$, we define the sampled tensor set $S[f]$ as*

$$S[f] := \{\mathcal{T} | \mathcal{T}_{(i,j,k)} = f(\mathbf{x}_{(i)}, \mathbf{y}_{(j)}, \mathbf{z}_{(k)}), \mathbf{x} \in X_f^{n_1}, \mathbf{y} \in Y_f^{n_2}, \mathbf{z} \in Z_f^{n_3}, n_1, n_2, n_3 \in \mathbb{N}_+\}, \tag{2}$$

*where $\mathbf{x}$, $\mathbf{y}$, $\mathbf{z}$ denote the coordinate vector variables and $n_1$, $n_2$, $n_3$ are positive integer variables that determine the sizes of the sampled tensor $\mathcal{T}$.*

In Definition 2, the sampled tensor set $S[f]$ is defined as the set containing all discrete tensors sampled from $f(\cdot)$. Specifically, given the sampling coordinate vectors $\mathbf{x} \in X_f^{n_1}, \mathbf{y} \in Y_f^{n_2}, \mathbf{z} \in Z_f^{n_3}$, the corresponding sampled tensor $\mathcal{T} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$ is defined as $\mathcal{T}_{(i,j,k)} = f(\mathbf{x}_{(i)}, \mathbf{y}_{(j)}, \mathbf{z}_{(k)})$ for all $i, j, k$, where $\mathcal{T}_{(i,j,k)}$ denotes the $(i,j,k)$-th element of $\mathcal{T}$, $\mathbf{x}_{(i)}$ denotes the $i$-th element of the vector $\mathbf{x}$, $\mathbf{y}_{(j)}$ denotes the $j$-th element of the vector $\mathbf{y}$, and $\mathbf{z}_{(k)}$ denotes the $k$-th element of the vector $\mathbf{z}$. Given different coordinate vectors $\mathbf{x}$, $\mathbf{y}$, $\mathbf{z}$, we can obtain different discrete tesnors in the set of tensors $S[f]$.

Tensor function is a promising and potential tool for multi-dimensional data processing. An interesting and fundamental question is whether we can analogously define the "rank" and develop the "tensor factorization" for tensor functions. Regarding the rank definition, a reasonable expectation is that if $f(\cdot)$ is low-rank, then any tensor sampled from $S[f]$ is a low-rank tensor. Thus, we can naturally define the function rank (F-rank) of $f(\cdot)$ as the supremum of the tensor rank in $S[f]$.

**Definition 3.** *Given a tensor function $f : D_f = X_f \times Y_f \times Z_f \to \mathbb{R}$, we define a measure of its complexity, denoted by* F-rank$[f]$ *(function rank of $f(\cdot)$), as the supremum of Tucker rank in the sampled tensor set $S[f]$:*

$$\text{F-rank}[f] := (r_1, r_2, r_3), \text{ where } r_i = \sup_{\mathcal{T} \in S[f]} \text{rank}(\mathbf{T}^{(i)}). \tag{3}$$

*Here, given a tensor $\mathcal{T}$ in the sampled tensor set $S[f]$, $\mathbf{T}^{(i)}$ $(i = 1,2,3)$ denotes the mode-$i$ unfolding matrix of $\mathcal{T}$.*

We call a tensor function $f(\cdot)$ with F-rank$[f] = (r_1, r_2, r_3)$ ($r_i < \infty$ for $i = 1,2,3$) as a low-rank tensor function since the Tucker rank of any $\mathcal{T} \in S[f]$ is bounded by $(r_1, r_2, r_3)$. When $f(\cdot)$ is defined on certain discrete sets, the F-rank degenerates into the discrete case, i.e., the classical Tucker rank, as stated in Proposition 1[1].

**Proposition 1.** *Let $\mathcal{X} \in \mathbb{R}^{m_1 \times m_2 \times m_3}$ be an arbitrary tensor. Let $X_f = \{1, 2, \cdots, m_1\}, Y_f = \{1, 2, \cdots, m_2\}, Z_f = \{1, 2, \cdots, m_3\}$ be three discrete sets, and denote $D_f = X_f \times Y_f \times Z_f$. Define the tensor function $f(\cdot) : D_f \to \mathbb{R}$ by $f(v_1, v_2, v_3) = \mathcal{X}_{(v_1, v_2, v_3)}$ for any $(v_1, v_2, v_3) \in D_f$. Then we have* F-rank$[f] = \text{rank}_T(\mathcal{X})$.

Proposition 1 establishes the connection between F-rank and the classical tensor rank in the discrete case. Otherwise, if the definition domains are continuous, e.g., $X_f = [1, n_1]$, $Y_f = [1, n_2]$, and $Z_f = [1, n_3]$ for some constants $n_i$s $(i = 1,2,3)$, $f(\cdot)$ can represent data beyond meshgrid with infinite resolution. Therefore, F-rank is an extension of Tucker rank from discrete tensors to tensor functions for continuous representations.

Similar to classical tensor representations, it is meaningful to think over whether a low-rank tensor function $f(\cdot)$ with $(\text{F-rank}[f])_{(i)} < \infty$ $(i = 1,2,3)$ can also have some tensor factorization strategies to encode the low-rankness. We present a positive answer that a tensor function $f(\cdot)$ with F-rank$[f] = (r_1, r_2, r_3)$ can be factorized as the product of a core tensor $\mathcal{C}$ and three factor functions $f_x(\cdot)$, $f_y(\cdot)$, and $f_z(\cdot)$, where their output dimensions are related to the F-rank $r_i$ $(i = 1,2,3)$. On the contrary, the products of a core tensor $\mathcal{C}$ and three factor functions $g_x(\cdot)$, $g_y(\cdot)$, and $g_z(\cdot)$ form a low-rank representation $g(\cdot)$, where F-rank$[g]$ is bounded by the output dimensions of $g_x(\cdot)$, $g_y(\cdot)$, and $g_z(\cdot)$. The theory is formally given as follows.

**Theorem 2.** *(Low-rank tensor function factorization) Let $f(\cdot) : D_f = X_f \times Y_f \times Z_f \to \mathbb{R}$ be a bounded tensor function, where $X_f, Y_f, Z_f \subset \mathbb{R}$. Then*

**(i)** *If* F-rank$[f] = (r_1, r_2, r_3)$*, then there exists a tensor $\mathcal{C} \in \mathbb{R}^{r_1 \times r_2 \times r_3}$ and three bounded functions $f_x(\cdot) : X_f \to$*

1. The proof is shown in supplementary files due to page limitation.

$\mathbb{R}^{r_1}$, $f_y(\cdot) : Y_f \to \mathbb{R}^{r_2}$, *and* $f_z(\cdot) : Z_f \to \mathbb{R}^{r_3}$ *such that for any* $(v_1, v_2, v_3) \in D_f$, $f(v_1, v_2, v_3) = \mathcal{C} \times_1 f_x(v_1) \times_2 f_y(v_2) \times_3 f_z(v_3)$.

**(ii)** *On the other hand, let* $\mathcal{C} \in \mathbb{R}^{r_1 \times r_2 \times r_3}$ *be an arbitrary tensor and* $g_x(\cdot) : X_g \to \mathbb{R}^{r_1}$, $g_y(\cdot) : Y_g \to \mathbb{R}^{r_2}$, *and* $g_z(\cdot) : Z_g \to \mathbb{R}^{r_3}$ *be arbitrary bounded functions defined on* $X_g$, $Y_g$, $Z_g \subset \mathbb{R}$. *Then we have* $(\text{F-rank}[g])_{(i)} \leq r_i$ $(i = 1, 2, 3)$, *where* $g(\cdot) : D_g = X_g \times Y_g \times Z_g \to \mathbb{R}$ *is defined by* $g(v_1, v_2, v_3) = \mathcal{C} \times_1 g_x(v_1) \times_2 g_y(v_2) \times_3 g_z(v_3)$ *for any* $(v_1, v_2, v_3) \in D_g$.

Theorem 2 is a natural extension of Tucker factorization (Theorem 1) from discrete meshgrid to the continuous domain. It inherits the nice property of Tucker factorization that the low-rank tensor function $f(\cdot)$ can be factorized into a core tensor and three factor functions.

**Remark 1.** *Tucker factorization (Theorem 1) is a special case of our Theorem 2 when* $D_f$ *(or* $D_g$*) is a certain discrete set representing meshgrid. This can be easily derived by incorporating Proposition 1 into Theorem 2.*

Remark 1 establishes the connection between discrete tensor factorization and our continuous tensor function factorization. Based on the low-rank tensor function factorization, we can represent the multi-dimensional data by a low-rank tensor function formulated as

$$[\![\mathcal{C}; f_x, f_y, f_z]\!](\mathbf{v}) := \mathcal{C} \times_1 f_x(\mathbf{v}_{(1)}) \times_2 f_y(\mathbf{v}_{(2)}) \times_3 f_z(\mathbf{v}_{(3)}), \tag{4}$$

which is parameterized by the core tensor $\mathcal{C}$ and factor functions $f_x(\cdot)$, $f_y(\cdot)$, and $f_z(\cdot)$. The representation implicitly encodes the low-rankness of the tensor function by the low-rank function factorization, i.e., any tensor sampled from the tensor function representation must be a low-rank tensor, as stated in Theorem 2.

In the LRTFR (4), we further suggest to use the MLP to parameterize the factor functions due to its powerful universal approximation abilities [60]. Specifically, we employ three MLPs $f_{\theta_x}(\cdot)$, $f_{\theta_y}(\cdot)$, and $f_{\theta_z}(\cdot)$ with parameters $\theta_x$, $\theta_y$, and $\theta_z$ to parameterize the factor functions $f_x(\cdot)$, $f_y(\cdot)$, and $f_z(\cdot)$. Taking $f_{\theta_x}(\cdot)$ as an example, it is formulated as

$$f_{\theta_x}(\mathbf{x}) := \mathbf{H}_d(\sigma(\mathbf{H}_{d-1} \cdots \sigma(\mathbf{H}_1 \mathbf{x}))) : X_f \to \mathbb{R}^{r_1}, \tag{5}$$

where $X_f \subset \mathbb{R}$ is the definition domain in the first dimension, $\sigma(\cdot)$ is the nonlinear activation function, and $\theta_x := \{\mathbf{H}_i\}_{i=1}^d$ are learnable weight matrices of the MLP. With these in mind, the MLP-parameterized LRTFR is formulated as $[\![\mathcal{C}; f_{\theta_x}, f_{\theta_y}, f_{\theta_z}]\!](\mathbf{v}) := \mathcal{C} \times_1 f_{\theta_x}(\mathbf{v}_{(1)}) \times_2 f_{\theta_y}(\mathbf{v}_{(2)}) \times_3 f_{\theta_z}(\mathbf{v}_{(3)})$, which is parameterized by the core tensor $\mathcal{C}$ and MLP weights $\theta_x$, $\theta_y$, and $\theta_z$.

### 3.3 Implicit Smooth Regularization of LRTFR

Since smoothness is another common property in multi-dimensional data, e.g., the temporal smoothness of videos [61] and the spectral smoothness of hyperspectral images [13], it is interesting to explore the smooth property in LRTFR besides the low-rankness. Next, we theoretically justify that LRTFR encodes an implicit smooth regularization brought from the specific structures of MLPs.

**Theorem 3.** *Let* $\mathcal{C} \in \mathbb{R}^{r_1 \times r_2 \times r_3}$, *and* $f_{\theta_x}(\cdot) : X_f \to \mathbb{R}^{r_1}$, $f_{\theta_y}(\cdot) : Y_f \to \mathbb{R}^{r_2}$, $f_{\theta_z}(\cdot) : Z_f \to \mathbb{R}^{r_3}$ *be three MLPs structured as in (5) with parameters* $\theta_x$, $\theta_y$, $\theta_z$, *where* $X_f, Y_f, Z_f \subset \mathbb{R}$. *Suppose that the MLPs share the same activation function* $\sigma(\cdot)$ *and depth* $d$. *Besides, we assume that*

- $\sigma(\cdot)$ *is Lipschitz continuous with Lipschitz constant* $\kappa$.
- *The* $\ell_1$*-norm of* $\mathcal{C}$ *is bounded by* $\eta_1 > 0$.
- *The* $\ell_1$*-norm of each weight matrix* $\mathbf{H}_i$ *in the three MLPs is bounded by* $\eta_2 > 0$. *Let* $\eta = \max\{\eta_1, \eta_2\}$.

*Define a tensor function* $f(\cdot) : D_f = X_f \times Y_f \times Z_f \to \mathbb{R}$ *as* $f(\cdot) := [\![\mathcal{C}; f_{\theta_x}, f_{\theta_y}, f_{\theta_z}]\!](\cdot)$. *Then, the following inequalities hold for any* $x_1, x_2 \in X_f$, $y_1, y_2 \in Y_f$, *and* $z_1, z_2 \in Z_f$:

$$\begin{cases} |f(x_1, y_1, z_1) - f(x_2, y_1, z_1)| \leq \delta |x_1 - x_2| \\ |f(x_1, y_1, z_1) - f(x_1, y_2, z_1)| \leq \delta |y_1 - y_2| \\ |f(x_1, y_1, z_1) - f(x_1, y_1, z_2)| \leq \delta |z_1 - z_2|, \end{cases} \tag{6}$$

*where* $\delta = \eta^{3d+1} \kappa^{3d-3} \zeta^2$ *and* $\zeta = \max\{|x_1|, |y_1|, |z_1|\}$.

Theorem 3 gives a Lipschitz type smoothness guarantee of the MLP-parameterized LRTFR. The smoothness is implicitly encoded under mild assumptions of nonlinear activation function and weight matrices, which are easy to achieve in real implementation. For example, most widely-used activation functions are Lipschitz continuous, such as ReLU, LeakyReLU, Sine, and Tanh. Moreover, we can expediently control the degree of smoothness by controlling the structures of MLPs, as explained in the following remark.

**Remark 2.** *In Theorem 3, we can see that the degree of smoothness, i.e.,* $\delta$, *is related to the Lipschitz constant* $\kappa$ *and the upper bound* $\eta$ *of weight matrices and core tensor. Thus, we can control two variables in practice to balance the implicit smoothness:*

1) *First, we use the sine function* $\sigma(\cdot) = \sin(\omega_0 \cdot)$ *as the nonlinear activation function in the MLPs. The sine function is Lipschitz continuous. An efficient way to adjust its Lipschitz constant* $\kappa$ *is to change the value of* $\omega_0$, *i.e., the smaller* $\omega_0$ *is, the smaller Lipschitz constant* $\kappa$ *can be got and the smoother result can be obtained.*
2) *Second, to control the upper bound* $\eta$ *of weight matrices and core tensor, we can tune the trade-off parameter of the energy regularization on MLP weights and the core tensor, known as the weight decay in modern deep learning optimizers. This strategy controls the intensity of* $\eta$.

Based on Theorem 3, we can deduce the following corollary concluding the smoothness of any sampled tensor on the continuous representation $f(\cdot)$.

**Corollary 1.** *Suppose the assumptions in Theorem 3 hold. Define* $f(\cdot) := [\![\mathcal{C}; f_{\theta_x}, f_{\theta_y}, f_{\theta_z}]\!](\cdot)$. *Then, for any* $\mathcal{T} \in S[f] \cap \mathbb{R}^{n_1 \times n_2 \times n_3}$ *(with* $n_i$*s being any positive numbers) sampled by coordinate vectors* $\mathbf{x} \in {X_f}^{n_1}$, $\mathbf{y} \in {Y_f}^{n_2}$, *and* $\mathbf{z} \in {Z_f}^{n_3}$, *where* $S[f]$ *denotes the sampled tensor set of* $f(\cdot)$ *introduced in Definition 2, the following inequalities hold for all* $(i, j, k)$*s* $(i = 1, 2, \cdots, n_1,\ j = 1, 2, \cdots, n_2,\ k = 1, 2, \cdots, n_3)$:

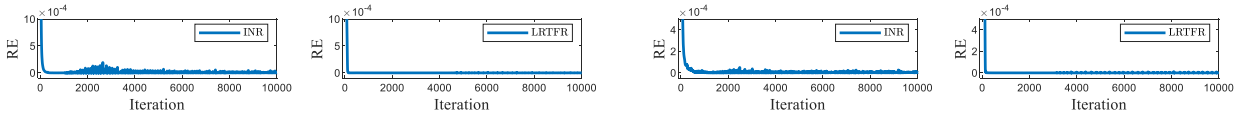
$$\begin{cases} |\mathcal{T}_{(\mathbf{x}_{(i)}, \mathbf{y}_{(j)}, \mathbf{z}_{(k)})} - \mathcal{T}_{(\mathbf{x}_{(i-1)}, \mathbf{y}_{(j)}, \mathbf{z}_{(k)})}| \leq \delta |\mathbf{x}_{(i)} - \mathbf{x}_{(i-1)}| \\ |\mathcal{T}_{(\mathbf{x}_{(i)}, \mathbf{y}_{(j)}, \mathbf{z}_{(k)})} - \mathcal{T}_{(\mathbf{x}_{(i)}, \mathbf{y}_{(j-1)}, \mathbf{z}_{(k)})}| \leq \delta |\mathbf{y}_{(j)} - \mathbf{y}_{(j-1)}| \\ |\mathcal{T}_{(\mathbf{x}_{(i)}, \mathbf{y}_{(j)}, \mathbf{z}_{(k)})} - \mathcal{T}_{(\mathbf{x}_{(i)}, \mathbf{y}_{(j)}, \mathbf{z}_{(k-1)})}| \leq \delta |\mathbf{z}_{(k)} - \mathbf{z}_{(k-1)}|, \end{cases}$$

*where* $\delta = \eta^{3d+1} \kappa^{3d-3} \tilde{\zeta}^2$ *and* $\tilde{\zeta} = \max\{\|\mathbf{x}\|_\infty, \|\mathbf{y}\|_\infty, \|\mathbf{z}\|_\infty\}$.

Corollary 1 claims that for any sampled tensor $\mathcal{T} \in S[f]$, the difference between its adjacent elements is bounded by

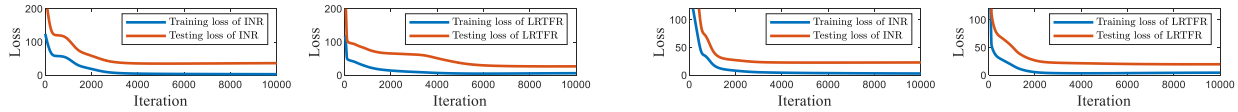
TABLE 2
The computational complexity, FLOPs, number of parameters, and average running time (second) of INR [44] and our LRTFR for image inpainting with image size $n_1 = 300$, $n_2 = 300$, and $n_3 = 3$.

| Method | Computational complexity | FLOPs | Number of parameters | Average running time |
|---|---|---|---|---|
| INR | $O(m^2 dn_1n_2n_3)$ | 1101.60 M | 41.20 K | 38.42 s |
| LRTFR | $O(m\hat{r}d(n_1+n_2+n_3)+\hat{r}n_1n_2n_3)$ | 42.24 M | 182.70 K | 7.78 s |




(a) *Peppers* (SR=0.2) (b) *Plane* (SR=0.2)

Fig. 2. The relative error of two successive solutions w.r.t. the iteration number by INR [44] and our LRTFR for image inpainting.




(a) *Peppers* (SR=0.2) (b) *Plane* (SR=0.2)

Fig. 3. The training loss and testing loss w.r.t. the iteration number by INR [44] and our LRTFR for image inpainting.

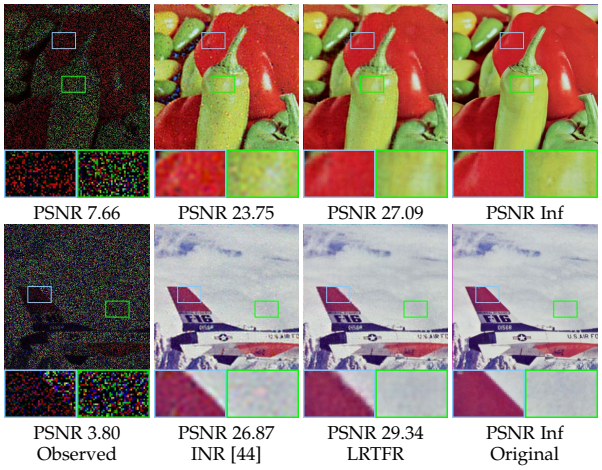



Fig. 4. The image inpainting results by INR [44] and our LRTFR on *Peppers* and *Plane* with sampling rate 0.2.

the distance between adjacent coordinates up to a constant. Hence, our LRTFR implicitly unifies the low-rankness and smoothness in all three dimensions, making it effective and efficient for continuous data representation.

### 3.4 Comparisons to Implicit Neural Representation

We next discuss the relationships and advantages of our method over the classical INR [44]. The classical INR learns an implicit function parameterized by deep neural networks $f_\theta(\cdot)$ to map a vector-form coordinate $(x, y, z) \in \mathbb{R}^3$ to the value of interest, i.e., $f_\theta(x, y, z)$. Our LRTFR (see illustrations in Fig. 1 (a)) maps some independent coordinates $x$, $y$, and $z$ to the corresponding value by disentangling the continuous representation into three simpler factor functions, implicitly encoding the low-rankness into the representation. Both INR and our method learn a continuous representation of data in the infinite real space. However, we additionally introduce the low-rank domain knowledge into the LRTFR, while the classical INR [44] ignores the intrinsic structure of data. By virtue of the introduced domain knowledge, our LRTFR enjoys the following intrinsic superiorities over the classical INR.

First, our method is more effective for multi-dimensional data recovery attributed to the low-rankness encoded in our LRTFR. As an example, we directly apply INR [44] and our LRTFR to the image inpainting task; see Fig. 4. Specifically, we directly use INR and LRTFR with the same MLP structure to fit the observed entries of the image and use the learned continuous representations to predict the unobserved entries. We have cropped the original image from $512 \times 512 \times 3$ to $300 \times 300 \times 3$ due to the large memory costs of INR. For fairness, we test INR and LRTFR with different hyperparameter values (i.e., the depth of MLP, the number of hidden units of MLP, and $\omega_0$, where $\omega_0$ is the hyperparameter of the sine activation function $\sin(\omega_0 \cdot)$), and select the best results for both INR and LRTFR for comparisons. The results in Fig. 4 show that our method is more effective for multi-dimensional data recovery than the classical INR, which can be attributed to the low-rankness encoded in our LRTFR.

Meanwhile, by exploiting the low-rankness of multi-dimensional data, our method has more steady numerical convergence behavior. To better demonstrate this, we have plotted the relative error (RE) curves of our method and the classical INR in Fig. 2 by calculating the RE of two successive solutions, i.e., $\text{RE} = \|\mathcal{T}^{t+1} - \mathcal{T}^t\|_F^2 / \|\mathcal{T}^t\|_F^2$, where $\mathcal{T}^t$ denotes the recovered image at the $t$-th iteration. We can observe that our method has steady numerical convergence behavior, while the RE curves of INR have vibrations. This should be rationally attributed to the low-rankness encoded in our LRTFR that constrains the solution space in a low-rank manifold.

Moreover, by exploiting the low-rankness of multi-dimensional data, our method tends to alleviates the over-fitting issue of classical INR. To intuitively depict this point, we have plotted the training loss and testing loss of our method and INR w.r.t. the iteration number on image inpainting problem; see Fig. 3. Here, the training loss refers to the loss on observed entries, i.e., Training Loss $= \|\mathcal{P}_\Omega(\mathcal{O} - \mathcal{T})\|_F^2$, where $\mathcal{O}$ denotes the observed image, $\mathcal{T}$ denotes the recovered image, $\Omega$ is the set of the observed indexes, and $\mathcal{P}_\Omega$ is the projection operator

that keeps the elements in $\Omega$ and makes others be zeros. Correspondingly, the testing loss is defined as Testing Loss = $\|\mathcal{P}_{\Omega^C}(\mathcal{G}-\mathcal{T})\|_F^2$, where $\mathcal{G}$ denotes the ground-truth image and $\Omega^C$ denotes the complementary set of $\Omega$. We can observe that the training losses of both our method and INR are monotonically decreasing. However, the testing loss of INR does not maintain such a consistent decreasing trend w.r.t. the iteration number, while our method still maintains consistent decreasing testing loss. These observations indicate that our method tends to alleviate the overfitting issue of INR attributed to the low-rankness encoded in our LRTFR.

Lastly, by virtue of the tensor function factorization, the parameters of our LRTFR can be efficiently trained with lower computational costs than classical INR. We take the image representation as an example. Given the observed image $\mathcal{O} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$, where $n_1, n_2, n_3$ respectively denote the lengths of the first, second, and third dimensions of the image, INR [44] needs an input coordinate matrix of size $n_1 n_2 n_3 \times 3$ to train the network, and the computational cost in each forward pass is $O(m^2 d n_1 n_2 n_3)$, where $m$ denotes the number of hiden units of the MLP and $d$ denotes the depth. In our LRTFR, the continuous representation (tensor function) is disentangled into three simpler factor functions, which could more efficiently represent the data. More specifically, we represent the image with size $n_1 \times n_2 \times n_3$ in Tucker format by using a core tensor $\mathcal{C} \in \mathbb{R}^{r_1 \times r_2 \times r_3}$ and three factor matrices $\mathbf{U} \in \mathbb{R}^{n_1 \times r_1}$, $\mathbf{V} \in \mathbb{R}^{n_2 \times r_2}$, and $\mathbf{W} \in \mathbb{R}^{n_3 \times r_3}$, where $r_1, r_2, r_3$ are preset ranks. In our LRTFR, we represent the factor matrix, e.g., $\mathbf{U}$, by a factor function $f_{\theta_x}(\cdot) : \mathbb{R} \to \mathbb{R}^{r_1}$. $f_{\theta_x}(\cdot)$ takes an $x$-coordinate scalar as the input and outputs a vector of size $1 \times r_1$. We apply the function $f_{\theta_x}(\cdot)$ to $n_1$ $x$-coordinate scalars respectively and output the corresponding $n_1$ vectors of size $1 \times r_1$. These output vectors form the factor matrix $\mathbf{U} \in \mathbb{R}^{n_1 \times r_1}$. Similarly, we apply $f_{\theta_y}(\cdot)$ to $n_2$ $y$-coordinates respectively to obtain the factor matrix $\mathbf{V}$ and apply $f_{\theta_z}(\cdot)$ to $n_3$ $z$-coordinates respectively to obtain the factor matrix $\mathbf{W}$. Therefore, the input size of our method for image representation is $n_1+n_2+n_3$, where $n_1, n_2, n_3$ are the lengths of the first, second, and third dimensions of the image, respectively. The computational cost of our method is $O(m\hat{r}d(n_1+n_2+n_3)+\hat{r}n_1n_2n_3)$ in each forward pass, where $\hat{r} = \max\{r_1, r_2, r_3\}$. This cost is much lower than that of INR, i.e., $O(m^2 d n_1 n_2 n_3)$, since $\hat{r}$ is much smaller than $m^2 d$ in practice. We summarize the computational complexity, floating point operations (FLOPs), number of parameters (i.e., model size), and average running time of our method and classical INR for image inpainting problem; see Table 2[2]. By virtue of the low-rank tensor function factorization, i.e., we use three MLPs and a core tensor to represent the image, our computational efficiency (in terms of computational complexity, FLOPs, and running time) is superior over INR by an evidently higher order of magnitude, except the model size (the model size of our method is relatively larger than that of INR while with similar magnitude).

Next, we discuss the relations and differences between our method and neural radiance fields (NeRF) [45]. The philosophy of INR is to use neural networks to continuously represent discrete data. Comparatively, NeRF [45] aims to use neural networks to continuously represent 3D scenes. Specifically, NeRF uses an MLP to process the input coordinate $(x, y, z)$, and outputs the volume density $\sigma$ and a feature vector. This feature vector is then concatenated with the camera ray's viewing direction and passed to one additional fully-connected layer that outputs the view-dependent RGB color. Therefore, NeRF and our method both essentially belong to the INR family, yet there are still substantial differences existed between NeRF and our method. Specifically, the idea of NeRF is to directly use neural networks to continuously represent 3D scenes. Our idea, however, is to integrate the advantages of domain knowledge (i.e., low-rankness) and neural networks to continuously represent multi-dimensional discrete data. By exploiting the low-rankness of multi-dimensional data, our method inclines to possess better stability and effectiveness for multi-dimensional data recovery tasks. Our idea can be extended to NeRF for 3D scene representation, but this is beyond the scope of this work and we will try to extend our method to NeRF in future work.

2. Here, the FLOPs and number of parameters are calculated by thop package (https://github.com/Lyken17/pytorch-OpCounter).

### 3.5 LRTFR for Multi-Dimensional Data Recovery

Since LRTFR can continuously represent data and capture the low-rank structure, it is versatile for data processing and analysis on or beyond meshgrid. In this work, we deploy the LRTFR in multi-dimensional data recovery problems to examinate its effectiveness. Specifically, we first establish a general data recovery model by using LRTFR and then more detailedly introduce four data recovery problems, including image inpainting and image denoising (on meshgrid with original resolution), hyperparameter optimization (on meshgrid beyond original resolution), and point cloud upsampling (beyond meshgrid).

#### *3.5.1 General Data Recovery Model*

In this section, we introduce a general model for multi-dimensional data recovery by using LRTFR. Suppose that the observed multi-dimensional data is defined in a function form $h(\cdot) : D_h \to \mathbb{R}$, where $D_h \subset \mathbb{R}^3$ is the observed set. The observed multi-dimensional data is usually in a discrete manner on meshgrid (e.g., images) or even not on meshgrid (e.g., point clouds). In both situations, the observed function $h(\cdot)$ is defined on a certain discrete set $D_h$. We assume that the underlying real low-rank tensor function has F-rank bounded by $(r_1, r_2, r_3)$. Using the MLP-parameterized LRTFR, we can formulate the following multi-dimensional data recovery model

$$\min_{\substack{\mathcal{C}\in\mathbb{R}^{r_1\times r_2\times r_3},\\ \theta_x,\theta_y,\theta_z}} \sum_{\mathbf{v}\in D_h} \left|h(\mathbf{v}) - [\![\mathcal{C}; f_{\theta_x}, f_{\theta_y}, f_{\theta_z}]\!](\mathbf{v})\right|^2, \tag{7}$$

where $f_{\theta_x}(\cdot) : X_f \to \mathbb{R}^{r_1}$, $f_{\theta_y}(\cdot) : Y_f \to \mathbb{R}^{r_2}$, and $f_{\theta_z}(\cdot) : Z_f \to \mathbb{R}^{r_3}$ are three factor MLPs, and $\mathcal{C}$ is the core tensor. The recovered low-rank tensor function is $f(\cdot) := [\![\mathcal{C}; f_{\theta_x}, f_{\theta_y}, f_{\theta_z}]\!](\cdot)$. According to Theorem 2, the real low-rank continuous tensor function must lies in the optimization space of (7) and the recovered tensor function $f(\cdot)$ must be a low-rank tensor function with $(\text{F-rank}[f])_{(i)} \leq r_i$. In the classical Tucker factorization, the factor matrices are usually required to be orthogonal [11] to constrain the solution space. Different from the discrete setting, it is difficult

to implement the orthogonal constraint on the continuous factor functions. The pioneer Tucker factorization method for multivariate function approximation [59] also does not consider the orthogonal constraint. Hence, we do not consider the orthogonal constraint in our LRTFR.

In model (7), the optimization variables are the core tensor $\mathcal{C}$ and the weights of MLPs, and the objective function is a square error term, which is differentiable w.r.t. all MLP weights and the core tensor. Hence, we can use easily attachable deep learning optimizers based on the gradient descent to tackle model (7). In this work, we consistently use the efficient adaptive moment estimation (Adam) algorithm [62]. Next, we more detailedly introduce four applications in multi-dimensional data recovery, which are some specific examples of the general model (7).

#### 3.5.2 Multi-Dimensional Image Inpainting

The multi-dimensional image inpainting [21], [63], which is a classical problem on the original meshgrid, aims to recover the underlying image from the observed image. Given an observed image $\mathcal{O} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$ with observed set $\Omega \subset \Psi$, where $\Psi := \{(i,j,k)|i = 1,2,\cdots,n_1, j = 1,2,\cdots,n_2, k = 1,2,\cdots,n_3\}$, the optimization model of our LRTFR for multi-dimensional image inpainting is formulated as

$$\begin{aligned} &\min_{\mathcal{C},\theta_x,\theta_y,\theta_z} \|\mathcal{P}_\Omega(\mathcal{O}-\mathcal{T})\|_F^2, \\ &\mathcal{T}_{(i,j,k)} = [\![\mathcal{C}; f_{\theta_x}, f_{\theta_y}, f_{\theta_z}]\!](i,j,k),\ \forall\ (i,j,k) \in \Psi. \end{aligned} \tag{8}$$

Here, $f_{\theta_x}(\cdot) : X_f \to \mathbb{R}^{r_1}$, $f_{\theta_y}(\cdot) : Y_f \to \mathbb{R}^{r_2}$, and $f_{\theta_z}(\cdot) : Z_f \to \mathbb{R}^{r_3}$ are three factor MLPs, $\mathcal{C} \in \mathbb{R}^{r_1 \times r_2 \times r_3}$ is the core tensor, and $r_i$s ($i = 1,2,3$) are preset F-ranks. $\mathcal{P}_\Omega(\cdot)$ denotes the projection operator that keeps the elements in $\Omega$ and makes others be zeros. The recovered result is obtained by $\mathcal{P}_\Omega(\mathcal{O}) + \mathcal{P}_{\Omega^C}(\mathcal{T})$, where $\Omega^C$ denotes the complementary set of $\Omega$ in $\Psi$. We adopt the Adam optimizer to address the image inpainting model (8) by optimizing the MLP parameters and core tensor. To better illustrate the proposed method, we summarize the steps of our method for multi-dimensional image inpainting in Algorithm 1.

#### 3.5.3 Multispectral Image Denoising

Multispectral image (MSI) denoising [10], [13] aims to recover a clean image from the noisy observation. It is another typical problem on the original meshgrid. In practice, MSI is corrupted with mixed noise, such as Gaussian noise, sparse noise, stripe noise, and deadlines (missing columns). Given the observed noisy MSI $\mathcal{O} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$, the optimization model of LRTFR for MSI denoising is

$$\begin{aligned} &\min_{\substack{\mathcal{C},\mathcal{S},\\ \theta_x,\theta_y,\theta_z}} \|\mathcal{O}-\mathcal{T}-\mathcal{S}\|_F^2 + \gamma_1\|\mathcal{S}\|_{\ell_1} + \gamma_2\|\mathcal{T}\|_{\mathrm{TV}}, \\ &\mathcal{T}_{(i,j,k)} = [\![\mathcal{C}; f_{\theta_x}, f_{\theta_y}, f_{\theta_z}]\!](i,j,k),\ \forall\ (i,j,k) \in \Psi. \end{aligned} \tag{9}$$

In this model, $f_{\theta_x}(\cdot) : X_f \to \mathbb{R}^{r_1}$, $f_{\theta_y}(\cdot) : Y_f \to \mathbb{R}^{r_2}$, and $f_{\theta_z}(\cdot) : Z_f \to \mathbb{R}^{r_3}$ are three factor MLPs, $\mathcal{C} \in \mathbb{R}^{r_1 \times r_2 \times r_3}$ is the core tensor, $\mathcal{S} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$ represents the sparse noise, $\mathcal{T} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$ denotes the recovered result, and $\gamma_i$s ($i = 1,2$) are trade-off parameters. We introduce a simple spatial TV regularization $\|\mathcal{T}\|_{\mathrm{TV}} := \sum_{k=1}^{n_3}(\sum_{i=1}^{n_1-1}\sum_{j=1}^{n_2}|\mathcal{T}_{(i+1,j,k)} - \mathcal{T}_{(i,j,k)}| + \sum_{i=1}^{n_1}\sum_{j=1}^{n_2-1}|\mathcal{T}_{(i,j+1,k)} - \mathcal{T}_{(i,j,k)}|)$ to more faithfully remove the noise. Here, the Lipchitz smoothness in Theorem 3 and the TV regularization reveal different types of smoothness: The Lipchitz smoothness delivers *global* smooth structures that the gradient is bounded everywhere, while TV considers *local* smoothness of adjacent pixels. The global and local smooth regularizations are complementary to each other to yield more promising denoising results.

**Algorithm 1** MLP-Parameterized LRTFR for Multi-Dimensional Image Inpainting

**Input:** Observed image $\mathcal{O} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$, observed set $\Omega$, F-ranks $r_1, r_2, r_3$, hyperparameters $\omega_0$;
**Initialization:** Initialize core tensor $\mathcal{C} \in \mathbb{R}^{r_1 \times r_2 \times r_3}$ and factor MLPs $f_{\theta_x}(\cdot), f_{\theta_y}(\cdot), f_{\theta_z}(\cdot)$ with activation function $\sigma(\cdot) = \sin(\omega_0 \cdot)$;
1: **while** not converge **do**
2: Compute the recovered tensor $\mathcal{T} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$ via $\mathcal{T}_{(i,j,k)} = \mathcal{C} \times_1 f_{\theta_x}(i) \times_2 f_{\theta_y}(j) \times_3 f_{\theta_z}(k)$;
3: Compute the loss in (8), i.e., $\|\mathcal{P}_\Omega(\mathcal{O}-\mathcal{T})\|_F^2$;
4: Update $\mathcal{C}$ and MLP parameters $\theta_x, \theta_y, \theta_z$ with the loss using the Adam algorithm;
5: **end while**
6: Compute $\mathcal{T} \leftarrow \mathcal{P}_\Omega(\mathcal{O}) + \mathcal{P}_{\Omega^C}(\mathcal{T})$;
**Output:** The recovered multi-dimensional image $\mathcal{T}$;

We utilize the alternating minimization algorithm to tackle the denoising model. Specifically, we tackle the following sub-problems in the $t$-th iteration:

$$\begin{aligned} &\min_{\mathcal{C},\theta_x,\theta_y,\theta_z} \|\mathcal{O}-\mathcal{T}-\mathcal{S}^t\|_F^2 + \gamma_2\|\mathcal{T}\|_{\mathrm{TV}}, \\ &\min_{\mathcal{S}}\ \ \|\mathcal{O}^t-\mathcal{T}^t-\mathcal{S}\|_F^2 + \gamma_1\|\mathcal{S}\|_{\ell_1}, \\ &\mathcal{T}_{(i,j,k)} = [\![\mathcal{C}; f_{\theta_x}, f_{\theta_y}, f_{\theta_z}]\!](i,j,k),\ \forall\ (i,j,k) \in \Psi. \end{aligned} \tag{10}$$

We utilize the Adam algorithm to tackle the $\{\mathcal{C}, \theta_x, \theta_y, \theta_z\}$ sub-problem. In each iteration of the alternating minimization, we employ one step of the Adam algorithm to update $\{\mathcal{C}, \theta_x, \theta_y, \theta_z\}$. The $\mathcal{S}$ sub-problem can be exactly solved by $\mathcal{S} = Soft_{\frac{\gamma_1}{2}}(\mathcal{O}^t - \mathcal{T}^t)$, where $Soft_v(\cdot) := sgn(\cdot)\max\{|\cdot| - v, 0\}$ denotes the soft-thersholding operator applied on each element of the input.

#### 3.5.4 Hyperparameter Optimization

Hyperparameter optimization (HPO) [64], [65], or hyperparameters searching, is a critical step in machine learning. Recent studies [64] cleverly modeled HPO as the classical low-rank tensor completion problem. It is even more interesting to explore the benefits of searching hyperparameter values in the continuous domain, and investigate what is the superiority of continuous representation over classical discrete tensor completion methods in HPO.

Specifically, the tensor completion-based HPO [64] formulates the hyperparameter search as the completion problem of higher-order tensors, whose elements represent the performances of the algorithm with different hyperparameter values. In practice, some partial observations have been given, i.e., there is an incompleted tensor $\mathcal{O} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$ that gives the real performances under some configurations. We aim to complete the tensor to predict the performances among all configurations. This problem can be equivalently formulated as the tensor completion problem (8). With the

completion result $\mathcal{T}$, we select the configuration corresponding to the best predicted performance (the maximum value in $\mathcal{T}$) as the recommended hyperparameter values.

Since our method predicts a tensor function on a continuous domain, it is interesting to explore the benefits of seeking hyperparameter values beyond meshgrid. Intuitively, seeking hyperparameter values in a continuous domain is expected to obtain a better result, because the optimal configuration probably does not lie in the fixed meshgrid. To illustrate this claim, we sample $\times 2$ and $\times 4$ super-resolution results using the learned tensor function with evenly spaced sampling, which offers more candidate configurations and gives the corresponding predictions. Results show this strategy could suggest a better configuration than meshgrid methods in most cases; see Sec. 4.3.

#### 3.5.5 Point Cloud Upsampling

We further apply our LRTFR for point cloud representation to test the effectiveness of our method beyond meshgrid. Point cloud representation is a challenging task due to the unstructured and unordered nature of point clouds. It is difficult to use classical meshgrid-based low-rank representations for point cloud representation, since these unordered point clouds are defined beyond meshgrid. As compared, our LRTFR can represent point clouds by using the continuous representation, which validates its versatility as compared with classical low-rank representations.

Specifically, we consider the point cloud upsampling task [66], [67], which refers to upsampling a sparse point cloud into a dense point cloud that will benefit subsequent applications [68]. Suppose that we are given a sparse point cloud $\mathbf{P} \in \mathbb{R}^{p\times 3}$, where $p$ denotes the number of points. We use $\Omega = \{\mathbf{P}_{(m,:)}\}_{m=1}^{p}$ to denote the observed set. We borrow the signed distance function (SDF) [42] to learn a continuous representation from sparse point cloud. The loss function to learn the SDF is formulated as

$$\begin{aligned}\min_{\mathcal{C},\theta_x,\theta_y,\theta_z} \sum_{\mathbf{v}\in\Omega} |s(\mathbf{v})| + \gamma_1 \int_{\mathbb{R}^3} \big|\|\frac{\partial s(\mathbf{v})}{\partial(\mathbf{v})}\|_F^2 - 1\big| d\mathbf{v} \\ + \gamma_2 \int_{\mathbb{R}^3\setminus\Omega} \exp(-|s(\mathbf{v})|) d\mathbf{v}, \\ s(\mathbf{v}) := \mathcal{C}\times_1 f_{\theta_x}(\mathbf{v}_{(1)})\times_2 f_{\theta_y}(\mathbf{v}_{(2)})\times_3 f_{\theta_z}(\mathbf{v}_{(3)}),\end{aligned} \tag{11}$$

where $s(\cdot) : \mathbb{R}^3 \to \mathbb{R}$ denotes the SDF represented by LRTFR, and $\gamma_i$s $(i = 1, 2)$ are trade-off parameters. The first term enforces the SDF to be zero on observed points. The second term enforces the SDF gradients to be one everywhere. The third term restricts the SDF value to be far from zero outside the observed set [44]. In practice, we approximate the integral by randomly sampling large number of points in the space. The loss function can be minimized by using the Adam algorithm. The solution of $s(\mathbf{v}) = 0$ forms a surface, which represents the underlying shape of the point cloud. We sample dense points $\mathbf{v}$ in the space such that $|s(\mathbf{v})| < \tau$ by evenly spaced sampling, where $\tau$ is a pre-defined threshold. These points represent the desired upsampling result.

## 4 Experiments

In experiments, we have conducted comparison experiments and analysis on all the introduced tasks. We first introduce some important experimental settings. Then, we introduce baselines, datasets, and results for different tasks. Our method is implemented on Pytorch 1.7.0. with an i5-9400f CPU and an RTX 3060 GPU (12 GB GPU memory).

**Evaluation metrics** For inpainting and denoising, we use peak-signal-to-noisy ratio (PSNR), structural similarity (SSIM), and normalized root mean square error (NRMSE) for evaluations. For HPO, we report the average classification accuracy (ACA) and average recommendation accuracy (ARA) [64], [69] of different methods. For point cloud upsampling, we adopt the widely-used Chamfer distance (CD) [70] and F-Score [71] as evaluation metrics.

**Hyperparameters settings** For all tasks, we search the values of F-rank $(r_1, r_2, r_3)$ in the following set

$$\{(\lfloor n_1/s \rfloor, \lfloor n_2/s \rfloor, \lfloor n_3/s_3 \rfloor) | s, s_3 = 1, 2, 4, 8, 16, 32\}, \tag{12}$$

where $n_i$s $(i = 1, 2, 3)$ denote the sizes of the observed data. Meanwhile, we adopt the sine activation function $\sigma(\cdot) = \sin(\omega_0 \cdot)$ in the MLPs to learn the LRTFR, where $\omega_0$ is a hyperparameter. It was thoroughly demonstrated in literatures [44], [48], [72] that the sine function has superior representation abilities than other activations (e.g., ReLU and RBF) for continuous representation. We search the hyperparameter $\omega_0$ in $\{1, 2, 4, 8, 16, 32\}$. The goal of the above hyperparameters search is to obtain the best PSNR (for inpainting and denoising), ACA (for HPO), and CD (for point cloud upsampling) values for different samples. The weight decay of Adam is set to 1, 0.1, 0.5, and 0.5 for inpainting, denoising, HPO, and point cloud upsampling. In the denoising model (9), we set $\gamma_2 = 10^{-4}$ for MSIs and $\gamma_2 = 10^{-5}$ for hyperspectral images. $\gamma_1$ is set to 0.1 (with sparse noise) or 10 (without sparse noise). In the point cloud upsampling model (11), we set $\gamma_1 = 10^{-5}$ and $\gamma_2 = 10^{-2}$ for all samples. The threshold $\tau$ is tuned such that the recovered dense point cloud has at least $10^4$ points. The number of hidden units of MLPs is set to $\max\{n_1, n_2, n_3\}$ for image inpainting, denoising, and hyperparameter optimization tasks, where $n_1, n_2, n_3$ denote the dimensions of the observed multi-dimensional data. The number of hidden units is set to the number of observed points for point cloud upsampling. The depth of MLPs is set to 3 for image inpainting, denoising, and hyperparameter optimization and 4 for point cloud upsampling. For all comparison methods, we have selected their hyperparameter values to make them possibly with optimal performance by following the recommended configurations provided by authors.

### 4.1 Multi-Dimensional Image Inpainting Results

The multi-dimensional image inpainting is a typical data recovery problem on meshgrid with the original resolution. To validate the effectiveness of LRTFR on the original meshgrid, we compare it with state-of-the-art low-rank tensor-based methods TRLRF [73], FTNN [63], FCTN [21], and HLRTF [31], and a recent INR-based method INRR [74]. The testing data include color images[3], MSIs in the CAVE dataset[4] [77], and videos[5]. We consider random missing with sampling rates (SRs) 0.1, 0.15, 0.2, 0.25, and 0.3.

3. http://sipi.usc.edu/database/database.php
4. https://www.cs.columbia.edu/CAVE/databases/multispectral/
5. http://trace.eas.asu.edu/yuv/

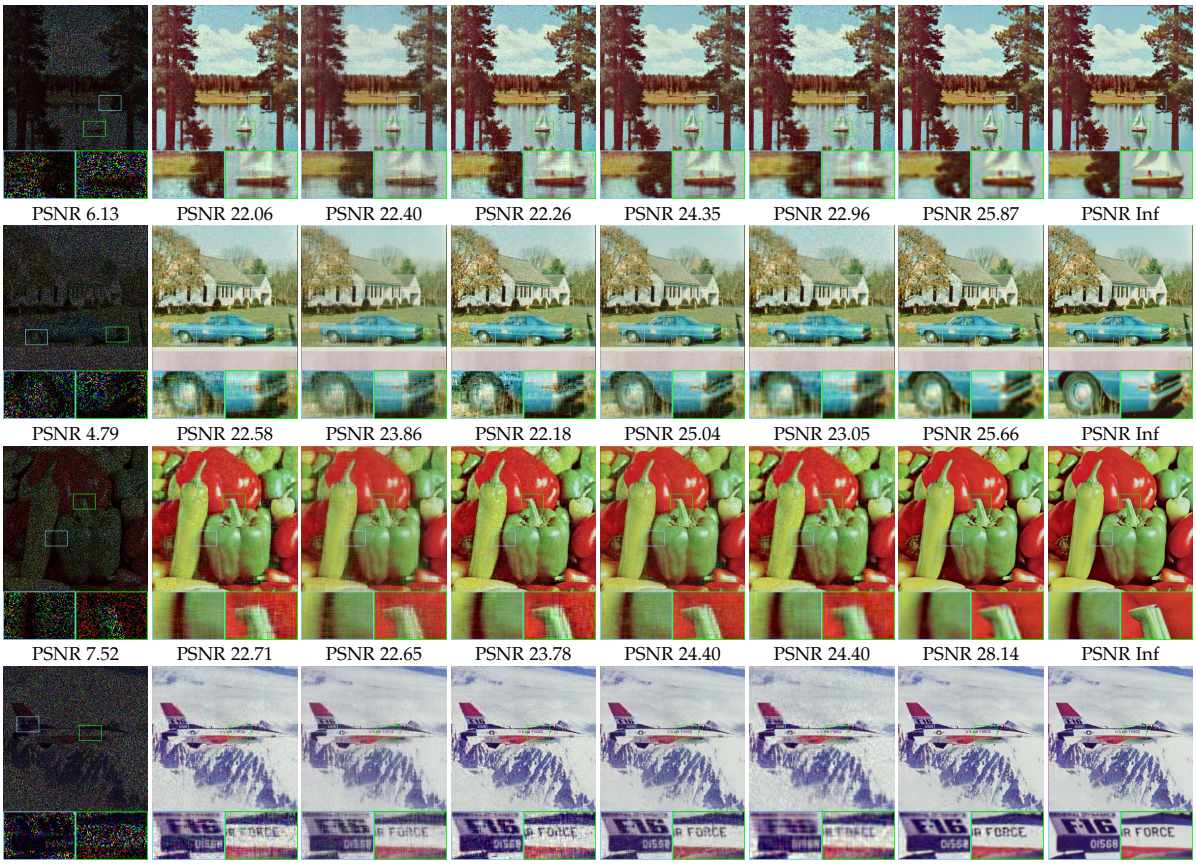



Fig. 5. The results of multi-dimensional image inpainting by different methods on color images *Sailboat*, *House*, *Peppers*, and *Plane* (SR = 0.2).

TABLE 3
The average quantitative results by different methods for multi-dimensional image inpainting. The **best** and second-best values are highlighted. (PSNR ↑, SSIM ↑, and NRMSE ↓)

| Sampling rate | | 0.1 | | | 0.15 | | | 0.2 | | | 0.25 | | | 0.3 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Data | Method | PSNR | SSIM | NRMSE | PSNR | SSIM | NRMSE | PSNR | SSIM | NRMSE | PSNR | SSIM | NRMSE | PSNR | SSIM | NRMSE |
| Color images *Sailboat* *House* *Peppers* *Plane* (512×512×3) | Observed | 5.01 | 0.043 | 3.003 | 5.26 | 0.054 | 2.381 | 5.52 | 0.065 | 2.001 | 5.80 | 0.075 | 1.734 | 6.10 | 0.086 | 1.529 |
| | TRLRF | 17.03 | 0.351 | 0.253 | 20.37 | 0.559 | 0.172 | 23.05 | 0.708 | 0.125 | 24.75 | 0.788 | 0.104 | 25.38 | 0.819 | 0.099 |
| | FTNN | 19.64 | 0.590 | 0.198 | 21.96 | 0.698 | 0.150 | 23.64 | 0.764 | 0.122 | 25.07 | 0.815 | 0.103 | 26.27 | 0.851 | 0.090 |
| | FCTN | 18.79 | 0.463 | 0.207 | 21.26 | 0.621 | 0.159 | 23.45 | 0.732 | 0.128 | 25.01 | 0.801 | 0.107 | 26.07 | 0.841 | 0.096 |
| | HLRTF | 22.49 | 0.700 | 0.136 | 24.41 | 0.779 | 0.110 | 25.39 | 0.811 | 0.097 | 26.34 | 0.842 | 0.086 | 27.17 | 0.868 | 0.078 |
| | INRR | 22.28 | 0.670 | 0.136 | 23.30 | 0.724 | 0.121 | 23.81 | 0.749 | 0.114 | 24.03 | 0.759 | 0.111 | 24.19 | 0.767 | 0.109 |
| | LRTFR | **24.88** | **0.827** | **0.102** | **26.00** | **0.849** | **0.089** | **27.35** | **0.892** | **0.079** | **27.65** | **0.895** | **0.075** | **28.42** | **0.916** | **0.071** |
| MSIs *Toys* *Flowers* (256×256×31) | Observed | 12.32 | 0.442 | 2.999 | 12.57 | 0.472 | 2.386 | 12.83 | 0.500 | 1.999 | 13.10 | 0.527 | 1.740 | 13.42 | 0.554 | 1.526 |
| | TRLRF | 27.83 | 0.797 | 0.165 | 35.60 | 0.966 | 0.070 | 37.03 | 0.975 | 0.060 | 37.85 | 0.979 | 0.054 | 38.49 | 0.982 | 0.050 |
| | FTNN | 35.08 | 0.973 | 0.079 | 38.09 | 0.985 | 0.057 | 40.61 | 0.990 | 0.045 | 42.65 | 0.994 | 0.036 | 44.58 | 0.995 | 0.030 |
| | FCTN | 36.72 | 0.973 | 0.064 | 40.18 | 0.986 | 0.044 | 42.08 | 0.990 | 0.036 | 43.32 | 0.992 | 0.031 | 44.73 | 0.994 | 0.026 |
| | HLRTF | 38.32 | 0.985 | 0.051 | 41.64 | 0.992 | 0.036 | 44.19 | 0.995 | 0.027 | 46.17 | **0.997** | 0.021 | 46.70 | **0.997** | 0.020 |
| | INRR | 34.74 | 0.953 | 0.087 | 37.16 | 0.974 | 0.063 | 38.09 | 0.979 | 0.054 | 38.41 | 0.981 | 0.051 | 38.75 | 0.983 | 0.048 |
| | LRTFR | **39.65** | **0.988** | **0.043** | **43.46** | **0.994** | **0.028** | **45.21** | **0.996** | **0.023** | **47.16** | **0.997** | **0.019** | **47.58** | **0.997** | **0.018** |
| Videos *Foreman* *Carphone* (144×176×100) | Observed | 5.20 | 0.044 | 3.004 | 5.45 | 0.059 | 2.381 | 5.72 | 0.073 | 1.998 | 6.03 | 0.087 | 1.730 | 6.30 | 0.101 | 1.527 |
| | TRLRF | 25.07 | 0.818 | 0.098 | 26.08 | 0.853 | 0.087 | 26.66 | 0.869 | 0.081 | 27.07 | 0.880 | 0.078 | 27.37 | 0.890 | 0.075 |
| | FTNN | 21.37 | 0.723 | 0.156 | 23.59 | 0.810 | 0.119 | 25.38 | 0.864 | 0.096 | 26.89 | 0.899 | 0.081 | 28.19 | 0.922 | 0.069 |
| | FCTN | 23.99 | 0.766 | 0.115 | 27.09 | 0.867 | 0.079 | 28.87 | 0.906 | 0.064 | 29.99 | 0.925 | 0.056 | 30.71 | 0.936 | 0.051 |
| | HLRTF | 24.66 | 0.768 | 0.104 | 26.49 | 0.830 | 0.085 | 28.10 | 0.877 | 0.071 | 29.52 | 0.908 | 0.060 | 30.80 | 0.932 | 0.052 |
| | INRR | 22.09 | 0.692 | 0.146 | 26.02 | 0.847 | 0.091 | 27.64 | 0.890 | 0.074 | 28.38 | 0.906 | 0.068 | 28.79 | 0.913 | 0.064 |
| | LRTFR | **27.56** | **0.901** | **0.074** | **29.29** | **0.918** | **0.060** | **30.14** | **0.930** | **0.054** | **30.96** | **0.942** | **0.050** | **32.05** | **0.959** | **0.044** |

The quantitative and qualitative results of multi-dimensional image inpainting are shown in Table 3 and Figs. 5-6. It can be seen that our LRTFR obtains the best results both quantitatively and qualitatively, which reveals the superiority of our LRTFR over classical low-rank tensor representations. The promising results of our method can be attributed to that our LRTFR concurrently encodes the low-rankness and smoothness into the representation. Meanwhile, we can observe that the recovered images of LRTFR are cleaner and smoother than other compared methods, which is mainly because our LRTFR implicitly encodes the Lipschitz smoothness into the continuous representation, making the recovered images have better visual qualities.

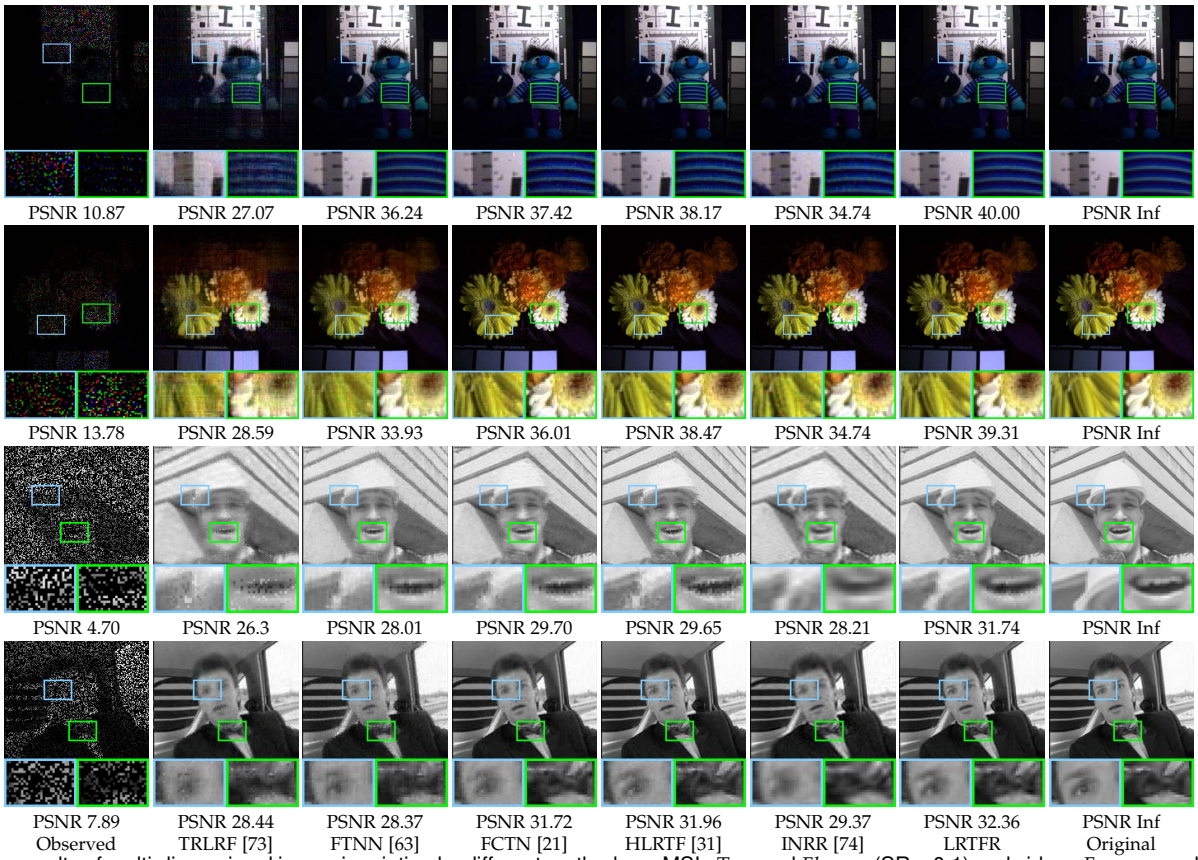



Fig. 6. The results of multi-dimensional image inpainting by different methods on MSIs *Toys* and *Flowers* (SR = 0.1) and videos *Foreman* and *Carphone* (SR = 0.3).

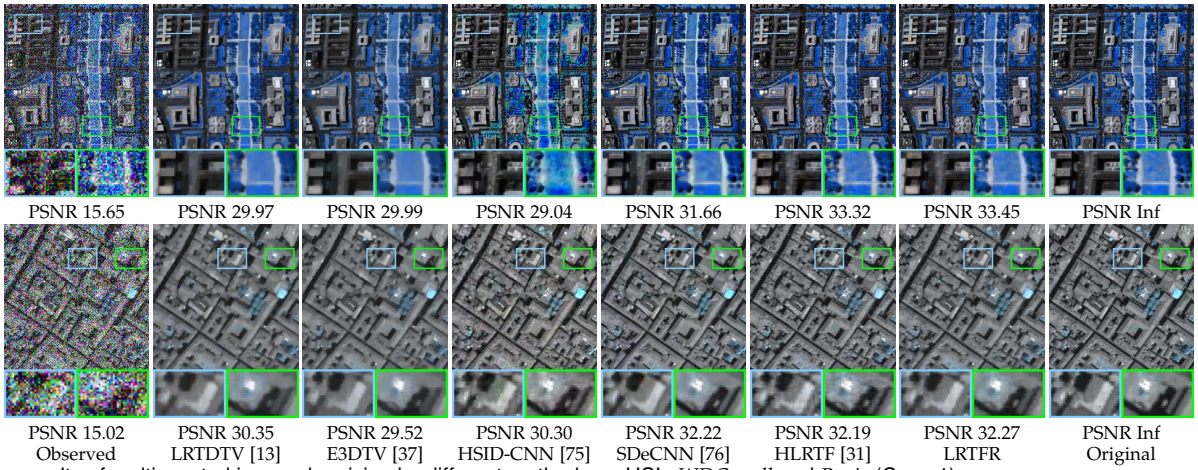



Fig. 7. The results of multispectral image denoising by different methods on HSIs *WDC mall* and *Pavia* (Case 1).

### 4.2 Multispectral Image Denoising Results

The MSI denoising is another challenging data recovery problem on the original meshgrid. We compare our LRTFR with low-rank matrix/tensor-based methods LRTDTV [13], E3DTV [37], and HLRTF [31]. Meanwhile, we include two supervised deep learning-based methods HSID-CNN [75] and SDeCNN [76] into comparisons. We use the best pre-trained models of HSID-CNN and SDeCNN for testing. The testing data includes four MSIs (*Balloons*, *Fruits* [77], *Pool*, and *Doll* [31]) and two hyperspectral images (HSIs)[6]. We consider different noisy cases. Case 1 contains Gaussian noise with standard deviation 0.2. Case 2 contains Gaussian noise with standard deviation 0.1 and sparse noise with SR 0.1. Case 3 contains the same noise of Case 2 plus deadlines in all spectral bands [78]. Case 4 contains the same noise of Case 2 plus stripe noise [78] in 40% of spectral bands. Case 5 contains the same noise of Case 3 plus stripe noise in 40% of spectral bands.

The results of MSI denoising are shown in Table 4 and Figs. 7-8. From Table 4, we can see that LRTFR is the most stable method among the tested algorithms in terms of different noisy cases and different data. Especially, LRTFR outperforms delicately designed TV-based methods LRTDTV and E3DTV, which validates the superiority of our combined global-local smooth regularizations. In Figs. 7-8, we can observe that our method can well remove complex

6. http://sipi.usc.edu/database/database.php

This article has been accepted for publication in IEEE Transactions on Pattern Analysis and Machine Intelligence. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2023.3341688



TABLE 4
The average quantitative results by different methods for multispectral image denoising. The **best** and second-best values are highlighted. (PSNR ↑, SSIM ↑, and NRMSE ↓)

| Case | | Case 1 | | | Case 2 | | | Case 3 | | | Case 4 | | | Case 5 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Data | Method | PSNR | SSIM | NRMSE | PSNR | SSIM | NRMSE | PSNR | SSIM | NRMSE | PSNR | SSIM | NRMSE | PSNR | SSIM | NRMSE |
| HSIs *WDC mall* (256×256×191) *Pavia* (200×200×80) | Observed | 15.34 | 0.290 | 0.634 | 14.02 | 0.270 | 0.632 | 13.87 | 0.254 | 0.664 | 13.80 | 0.260 | 0.635 | 13.64 | 0.245 | 0.666 |
| | LRTDTV | 30.16 | 0.915 | 0.136 | 32.81 | 0.953 | 0.099 | 23.97 | 0.826 | 0.301 | 32.57 | 0.950 | 0.101 | 23.94 | 0.823 | 0.299 |
| | E3DTV | 29.76 | 0.911 | 0.148 | 33.10 | 0.959 | 0.101 | 23.98 | 0.827 | 0.301 | 34.01 | **0.968** | 0.087 | 24.15 | 0.838 | 0.292 |
| | HSID-CNN | 29.67 | 0.919 | 0.146 | 25.93 | 0.858 | 0.198 | 22.26 | 0.749 | 0.315 | 24.54 | 0.834 | 0.229 | 21.58 | 0.726 | 0.329 |
| | SDeCNN | 31.94 | 0.947 | 0.107 | 26.77 | 0.892 | 0.176 | 22.45 | 0.766 | 0.304 | 25.22 | 0.866 | 0.210 | 21.73 | 0.741 | 0.319 |
| | HLRTF | 32.75 | **0.954** | 0.099 | 34.00 | **0.969** | 0.083 | 30.69 | 0.936 | 0.128 | 33.51 | 0.962 | 0.089 | 31.16 | 0.942 | 0.117 |
| | LRTFR | **32.86** | **0.954** | **0.097** | **34.60** | **0.969** | **0.079** | **32.01** | **0.951** | **0.108** | **34.10** | **0.968** | **0.083** | **31.65** | **0.947** | **0.114** |
| MSIs *Balloons* *Fruits* (256×256×31) | Observed | 16.16 | 0.149 | 0.745 | 13.48 | 0.124 | 0.766 | 13.53 | 0.127 | 0.789 | 13.17 | 0.118 | 0.772 | 13.22 | 0.121 | 0.794 |
| | LRTDTV | 33.58 | 0.941 | 0.143 | 34.22 | 0.914 | 0.128 | 25.55 | 0.852 | 0.330 | 34.06 | 0.916 | 0.127 | 25.49 | 0.845 | 0.330 |
| | E3DTV | 31.86 | 0.928 | 0.186 | 33.71 | 0.931 | 0.149 | 25.43 | 0.860 | 0.352 | 34.45 | 0.927 | 0.131 | 25.40 | 0.853 | 0.343 |
| | HSID-CNN | 29.59 | 0.864 | 0.222 | 24.25 | 0.599 | 0.330 | 22.70 | 0.564 | 0.405 | 23.14 | 0.563 | 0.367 | 21.61 | 0.516 | 0.445 |
| | SDeCNN | 34.80 | 0.942 | 0.118 | 25.29 | 0.656 | 0.286 | 22.85 | 0.594 | 0.386 | 23.53 | 0.603 | 0.339 | 21.77 | 0.539 | 0.423 |
| | HLRTF | 35.19 | 0.945 | 0.110 | 35.89 | 0.953 | 0.099 | 34.06 | 0.943 | 0.116 | 34.98 | 0.941 | 0.107 | 34.08 | 0.932 | 0.138 |
| | LRTFR | **35.63** | **0.964** | **0.103** | **36.76** | **0.958** | **0.090** | **35.67** | **0.954** | **0.101** | **35.69** | **0.947** | **0.103** | **34.36** | **0.937** | **0.126** |
| MSIs *Pool* *Doll* (256×288×49) | Observed | 15.47 | 0.150 | 0.615 | 13.91 | 0.137 | 0.626 | 13.75 | 0.135 | 0.654 | 13.69 | 0.133 | 0.631 | 13.53 | 0.130 | 0.658 |
| | LRTDTV | 31.47 | 0.943 | 0.118 | 33.31 | 0.957 | 0.094 | 24.61 | 0.855 | 0.273 | 32.94 | 0.952 | 0.097 | 24.56 | 0.852 | 0.273 |
| | E3DTV | 29.69 | 0.927 | 0.159 | 31.19 | 0.943 | 0.130 | 24.26 | 0.845 | 0.286 | 32.25 | 0.950 | 0.110 | 24.38 | 0.849 | 0.278 |
| | HSID-CNN | 28.95 | 0.859 | 0.152 | 24.93 | 0.744 | 0.208 | 22.13 | 0.669 | 0.303 | 23.87 | 0.716 | 0.234 | 21.43 | 0.640 | 0.319 |
| | SDeCNN | 31.57 | 0.941 | 0.109 | 26.19 | 0.823 | 0.181 | 22.73 | 0.730 | 0.286 | 24.90 | 0.790 | 0.210 | 22.01 | 0.696 | 0.302 |
| | HLRTF | 35.13 | 0.953 | 0.076 | 35.50 | 0.961 | 0.065 | 34.42 | 0.951 | 0.082 | 34.98 | 0.955 | 0.074 | 34.70 | 0.953 | 0.077 |
| | LRTFR | **35.30** | **0.958** | **0.075** | **36.66** | **0.966** | **0.062** | **36.38** | **0.964** | **0.064** | **35.66** | **0.958** | **0.071** | **35.29** | **0.955** | **0.069** |

TABLE 5
The quantitative results by different HPO methods. We report the results on eight binary classification tasks using Gaussian kernel-based SVM with hyperparameters suggested by different HPO methods. The **best** and second-best values are highlighted. (ACA ↑ and ARA ↑)

| Data variance | | 0.05 | | 0.1 | | 0.15 | | 0.2 | | 0.25 | | 0.3 | | 0.35 | | 0.4 | | Average | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Method | | ACA | ARA | ACA | ARA | ACA | ARA | ACA | ARA | ACA | ARA | ACA | ARA | ACA | ARA | ACA | ARA | ACA | ARA |
| Meshgrid | RS (Observed) | 93.68 | 94.53 | 88.69 | 95.22 | 84.98 | 93.88 | 83.07 | 94.61 | 74.88 | 90.39 | 76.04 | 94.27 | 73.57 | 89.91 | 70.39 | 89.74 | 80.66 | 92.82 |
| | DCTNN | 93.81 | 92.58 | 88.47 | 95.26 | 83.76 | 87.65 | 81.96 | 90.81 | 74.42 | 85.34 | 74.01 | 85.70 | 73.68 | 87.18 | 68.27 | 74.42 | 79.80 | 87.37 |
| | TRLRF | 94.06 | **96.90** | 87.64 | 93.80 | 84.96 | 93.43 | 80.70 | 86.15 | 74.49 | 85.22 | 74.89 | 90.37 | 73.73 | 88.30 | 70.69 | 89.32 | 80.15 | 90.44 |
| | FTNN | 93.74 | 94.56 | 88.48 | 93.91 | 83.82 | 87.88 | 81.86 | 88.93 | 75.26 | 89.97 | 75.96 | 93.84 | 72.19 | 86.15 | 70.56 | 89.18 | 80.23 | 90.55 |
| | FCTN | 91.46 | 90.52 | 83.61 | 80.87 | 82.84 | 87.15 | 75.23 | 66.05 | 72.08 | 77.47 | 72.24 | 78.08 | 71.96 | 80.81 | 67.01 | 72.13 | 77.05 | 79.13 |
| | HLRTF | 93.25 | 90.60 | 88.48 | 94.56 | 83.98 | 84.52 | 81.82 | 89.90 | 74.93 | 89.25 | 75.74 | 92.56 | 73.63 | 89.02 | 70.40 | 86.66 | 80.28 | 89.63 |
| Beyond meshgrid | LRTFR | 94.10 | 96.14 | 88.84 | 96.16 | 85.41 | 95.08 | 83.09 | **94.72** | 75.50 | 89.71 | 75.66 | 91.35 | 73.79 | 87.93 | 70.68 | 91.80 | 80.88 | 92.86 |
| | LRTFR (×2) | **94.14** | 96.35 | 88.79 | 95.92 | **85.46** | **95.76** | **83.18** | 94.48 | **76.06** | 91.88 | **76.18** | **94.89** | 74.05 | 90.04 | 70.77 | 91.94 | **81.08** | 93.91 |
| | LRTFR (×4) | **94.14** | 96.33 | **88.87** | **96.23** | 85.39 | 95.44 | 83.12 | 94.30 | 76.04 | **92.10** | 76.07 | 94.44 | **74.14** | **90.95** | **70.78** | **92.04** | 81.07 | **93.98** |
| Grid Search (Ground-Truth) | | 94.79 | 100.0 | 90.03 | 100.0 | 86.28 | 100.0 | 84.41 | 100.0 | 77.18 | 100.0 | 77.19 | 100.0 | 75.43 | 100.0 | 72.14 | 100.0 | 82.18 | 100.0 |

noise. As compared, other denoising methods sometimes do not completely remove the mixed noise. Moreover, from Fig. 7, we can see that other model-based methods (LRTDTV and E3DTV) may produce over-smoothness. In contrast, our method preserves the image details better. The deep learning methods HSID-CNN and SDeCNN have relatively good performances for Gaussian noise (Case 1), but suffer from domain gap between training and testing samples when dealing with mixed noise. As compared, our method is a model-based method that implicitly encodes low-rankness and smoothness, which delivers more stable performances for different types of noise.

### 4.3 Hyperparameter Optimization Results

The HPO can be elegently modeled as the low-rank tensor completion problem [64]. As aforementioned, it is even more interesting to exploit the benefits of conducting HPO in the continuous domain by using our LRTFR.

Following the experimental settings of previous SOTA work [64] along this research line, we consider the classification problem using Gaussian kernel-based support vector machine (SVM), which has two hyperparameters—the regularization parameter $C$ and the kernel parameter $\sigma$. The search sets are $\{3^i|i = -15 : 1 : 15\}$ for $C$ and $\{2^i|i = -15 : 1 : 15\}$ for $\sigma$, and thus there are $31 \times 31 = 961$ candidate configurations. We use Gaussian distribution to generate classification datasets. The variance of Gaussian distribution is traversed in $\{0.05, 0.1, 0.15, \cdots, 0.4\}$ to construct eight classification tasks with different difficulty levels. Each task contains 16 datasets. These datasets are generated by Gaussian distributions with the same variance and random means, and we divide the generated points into two groups to form a binary classification dataset. Each dataset has 100 training points and 500 testing points. Therefore, by conducting grid search (GS) for all configurations and datasets, we form a tensor of size $31 \times 31 \times 16$ for each task (variance). The first two dimensions ($31 \times 31$) indicate the number of candidate configurations and the third dimension ($16$) indicates the number of datasets. By following [64], we use a low SR of $0.01$ to sample the tensor slice corresponding to the new dataset and use the SR of $0.1$ to sample the tensor slices corresponding to historical datasets (see detailed explanations in [64]), which forms the incompleted tensor. We repeat the sampling process 16 times, where each dataset is chosen as the new dataset once and the others are historical datasets. It results in 16 incompleted tensors and we report the average HPO results on the new dataset.

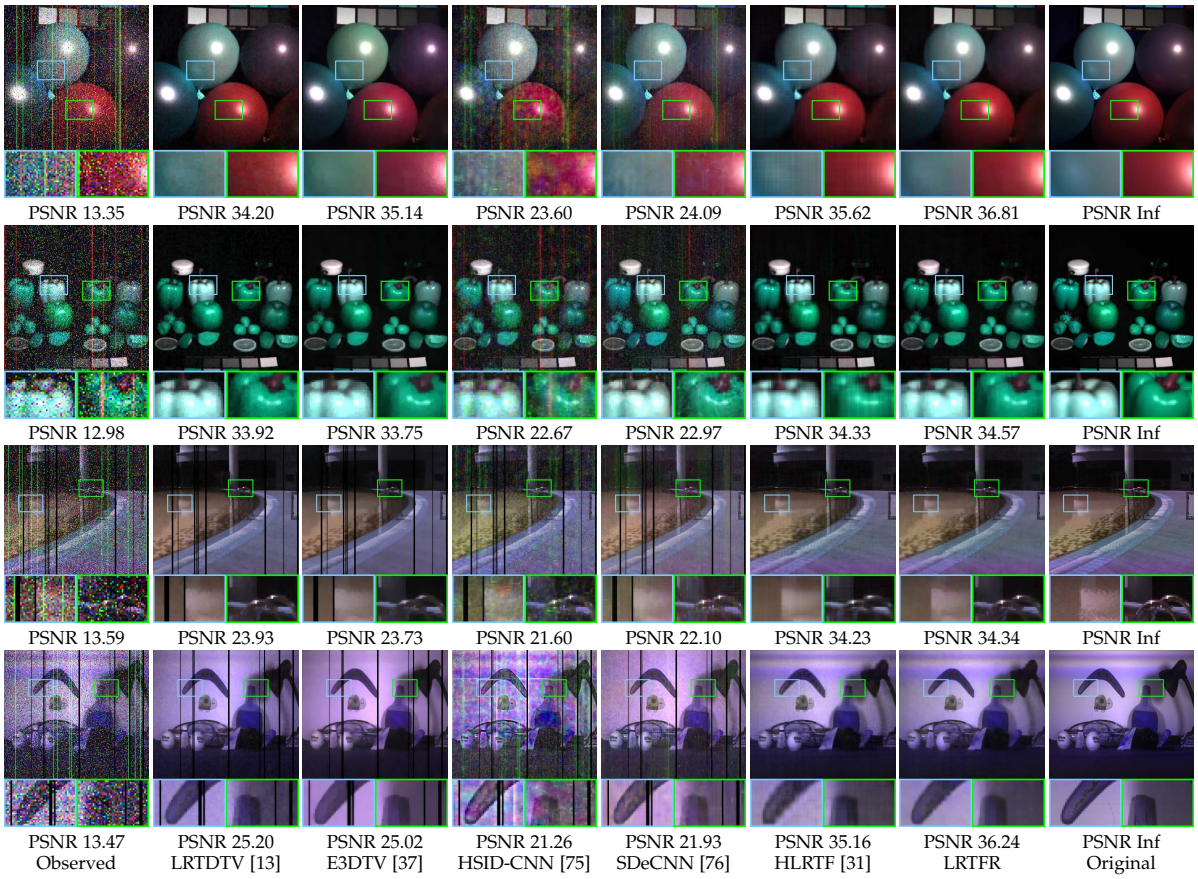



Fig. 8. The results of multispectral image denoising by different methods on *Balloons* (Case 4), *Fruits* (Case 4), *Pool* (Case 5), and *Doll* (Case 5).

We report three results of our method, including the standard tensor completion using LRTFR, and the super-resolution results using the learned continuous representation and evenly spaced sampling, termed as LRTFR $(\times 2)/(\times 4)$. The super-resolution results offer more candidate configurations and gives the predicted accuracy even though our method only sees the observed data on the original meshgrid. We compare our method with random search (RS) [65], which corresponds to the recommendation results using the observed tensor. Meanwhile, we use state-of-the-art tensor completion methods DCTNN [29], TRLRF [73], FTNN [63], FCTN [21], and HLRTF [31] as baselines.

The results of HPO are shown in Table 5. Compared to classical tensor completion methods, our LRTFR is more effective for suggesting a suitable configuration. Moreover, LRTFR $(\times 2)/(\times 4)$ incline to attain better performances than LRTFR, which reveals that searching the hyperparameter value beyond the original meshgrid resolution is helpful to obtain a better recommendation result. Meanwhile, we observe that LRTFR $(\times 2)$ and $(\times 4)$ obtain similar average performances. The similar performances of LRTFR $(\times 2)$ and $(\times 4)$ can be rationally explained by that the performance limit of the upsampling methods $(\times 2)$ and $(\times 4)$ is up to the given information contained in the original observed values on meshgrid. The reason why $(\times 2)$ and $(\times 4)$ obtain almost the same average results is that they use the same given information and should possibly have reached the performance limit of upsampling methods. Yet considering the consistent performance enhancement of LRTFR $(\times 2)/(\times 4)$ as compared with LRTFR, it is still rational to say that searching the hyperparameter values beyond the original meshgrid resolution tends to help obtain a better recommendation result, which reveals the superiority of the continuous representation.

### 4.4 Point Cloud Upsampling Results

Then, we consider the point cloud upsampling problem to show the effectiveness of our method beyond meshgrid. Standard low-rank tensor-based methods can not be applied to point cloud upsampling since they are not suitable for representing the unordered point cloud beyond meshgrid. As compared, our LRTFR is suitable to represent the point cloud since it learns a continuous representation of data.

We consider five deep learning-based methods as baselines, including MSN [79], SnowflakeNet [80], SMOG [81], NeuralPoints [82], and SAPCU [83]. Here, SAPCU [83] is an INR-based method. We use the best pre-trained models provided by authors. We adopt different datasets, including *Table*, *Sofa*, *Vessel*, and *Lamp* in the ShapeNet benchmark [84], the Stanford *Bunny*[7], and three hand-crafted shapes (*Doughnut*, *Sphere*, and *Heart*). We use random sampling to downsample the original point cloud to the observed point cloud with number of points less than 500.

The results for point cloud upsampling are shown in Table 6 and Figs. 9-10. Here, the relative improvement is calculated by $|m_1 - m_0|/m_0$, where $m_0$ and $m_1$ denote the metric values (i.e., DC/F-Score values) of the comparison method and our method, respectively. From Table 6, we can observe that our method gains a substantial margin in terms of the average metric value and the relative improvement

7. https://graphics.stanford.edu/data/3Dscanrep/

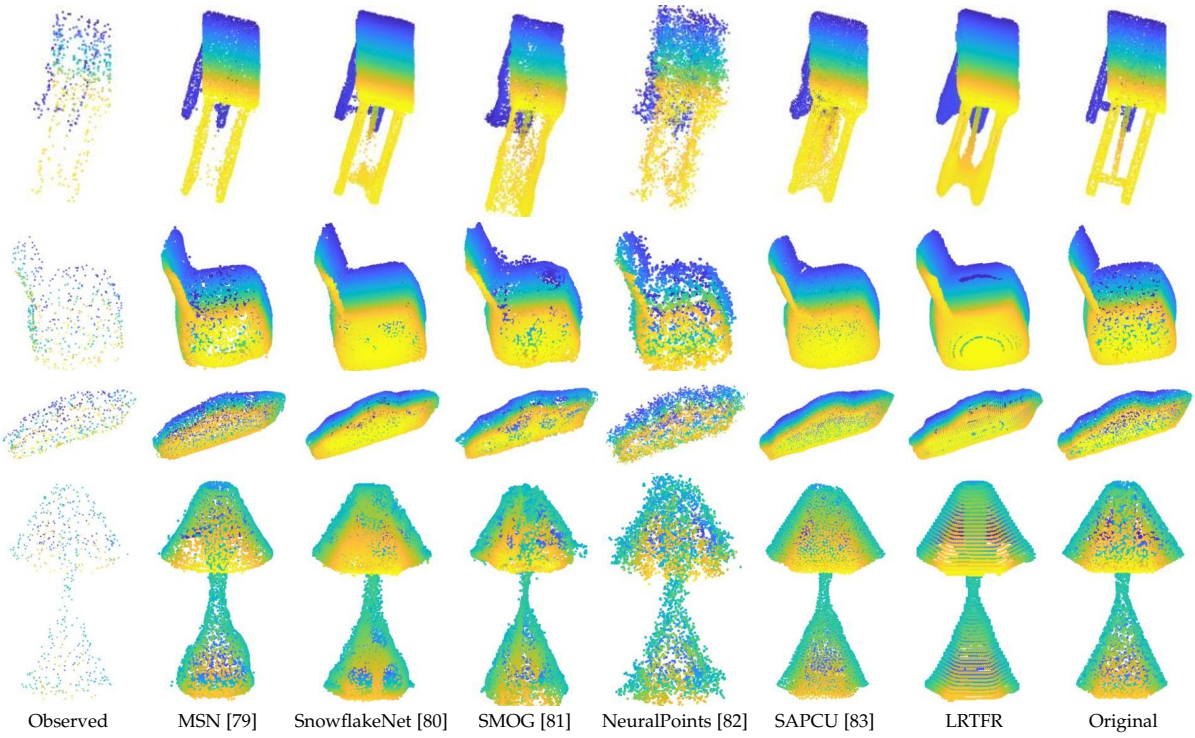



Fig. 9. The results of point cloud upsampling by different methods on *Table*, *Sofa*, *Vessel*, and *Lamp* in the ShapeNet dataset [84].

TABLE 6
The quantitative results by different methods for point cloud upsampling. The **best** and second-best values are highlighted. (CD ↓ and F-Score ↑)

| Data | *Table* | | *Sofa* | | *Vessel* | | *Lamp* | | *Bunny* | | *Doughnut* | | *Sphere* | | *Heart* | | Average | | Relative improvement | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Method | CD | F-Score | CD | F-Score | CD | F-Score | CD | F-Score | CD | F-Score | CD | F-Score | CD | F-Score | CD | F-Score | CD | F-Score | CD | F-Score |
| Observed | 0.0187 | 0.3448 | 0.0221 | 0.2976 | 0.0180 | 0.3866 | 0.0185 | 0.3677 | 0.1098 | 0.2086 | 0.1882 | 0.1863 | 0.0451 | 0.7379 | 0.0959 | 0.2201 | 0.0645 | 0.3437 | 48.37% | 159.6% |
| MSN | 0.0164 | 0.6918 | 0.0183 | 0.6267 | 0.0154 | 0.7768 | 0.0167 | 0.7188 | 0.1624 | 0.4157 | 0.3639 | 0.1408 | 0.1134 | 0.5529 | 0.2491 | 0.3145 | 0.1195 | 0.5298 | 72.13% | 68.45% |
| SnowflakeNet | **0.0106** | **0.9068** | **0.0125** | **0.8668** | 0.0096 | 0.9493 | 0.0126 | 0.8511 | 0.2017 | 0.3320 | 0.1248 | 0.5194 | 0.0228 | 0.9891 | 0.0729 | 0.8268 | 0.0584 | 0.7802 | 42.97% | 14.39% |
| SMOG | 0.0288 | 0.4819 | 0.0215 | 0.6163 | 0.0151 | 0.7769 | 0.0206 | 0.5783 | 0.2843 | 0.2380 | 0.1333 | 0.5036 | 0.0392 | 0.9642 | 0.1245 | 0.4662 | 0.0834 | 0.5782 | 60.07% | 54.35% |
| NeuralPoints | 0.0283 | 0.3900 | 0.0227 | 0.5083 | 0.0227 | 0.5272 | 0.0246 | 0.4800 | 0.0558 | 0.9045 | 0.2456 | 0.3316 | 0.0342 | 0.9559 | 0.0856 | 0.7326 | 0.0649 | 0.6038 | 48.69% | 47.81% |
| SAPCU | 0.0194 | 0.7234 | 0.0148 | 0.8085 | 0.0152 | 0.8848 | 0.0161 | 0.7719 | 0.0518 | 0.9187 | 0.1329 | 0.4142 | 0.0271 | 0.9974 | 0.0526 | 0.9130 | 0.0412 | 0.8040 | 19.17% | 11.01% |
| LRTFR | 0.0113 | 0.8453 | **0.0125** | 0.8387 | **0.0090** | **0.9723** | **0.0111** | **0.9360** | **0.0445** | **0.9784** | **0.1113** | **0.5871** | **0.0200** | **1.0000** | **0.0463** | **0.9824** | **0.0333** | **0.8925** | -- | -- |

of the average metric value. SnowflakeNet has favorable performances on *Table* and *Sofa*, which are included in the ShapeNet dataset [84]. This is because SnowflakeNet is trained on the same ShapeNet dataset. Our method has comparable performances on *Table* and *Sofa*, which are included in the ShapeNet dataset, and attains better performances than the other methods on the other datasets, especially on wild datasets, e.g., *Bunny*, *Doughnut*, *Sphere*, and *Heart*. This phenomenon can be rationally attributed to that our unsupervised method, which does not need any training dataset, encodes the low-rankness into the continuous representation and thus attains favorable generalization abilities over different datasets. From Figs. 9-10, we can observe that our LRTFR can generally obtain better recovered point clouds as compared with other methods, which validates the effectiveness of LRTFR for continuous representation.

## 5 Discussions

### 5.1 Influences of Tensor Factorization

The tensor Tucker factorization is a core building block in our LRTFR. However, our method can be readily extended to different tensor function factorizations. Here, we compare the CP factorization [11] and the Tucker factorization in our method for multi-dimensional image inpainting; see Fig. 11 (d)-(e). It can be seen that Tucker function factorization shows certain advantageous performances. This is possibly attributed to the better tensor structure preserving capability by the former Tucker representation manner. We will more deeply investigate this issue in future research.

### 5.2 Influences of Activation Functions

Since we use MLPs to parameterize the factor functions of LRTFR, the selection of activation function in the MLPs warrants discussion. Inspired by recent study of INR [44], which shows that the periodic sine activation function can capture natural signals' complex structures and fine details, we adopt the sine activation function in the MLP to learn the LRTFR to help obtain a more realistic continuous representation. To validate the effectiveness of the sine activation function, we compare it with ReLU, LeakyReLU, and Tanh activation functions; see Fig. 11 (a)-(c), (e). The results show that the sine activation function can help recover the signal much better than other activation functions, which are consistent with the results in existing literatures [44], [72].

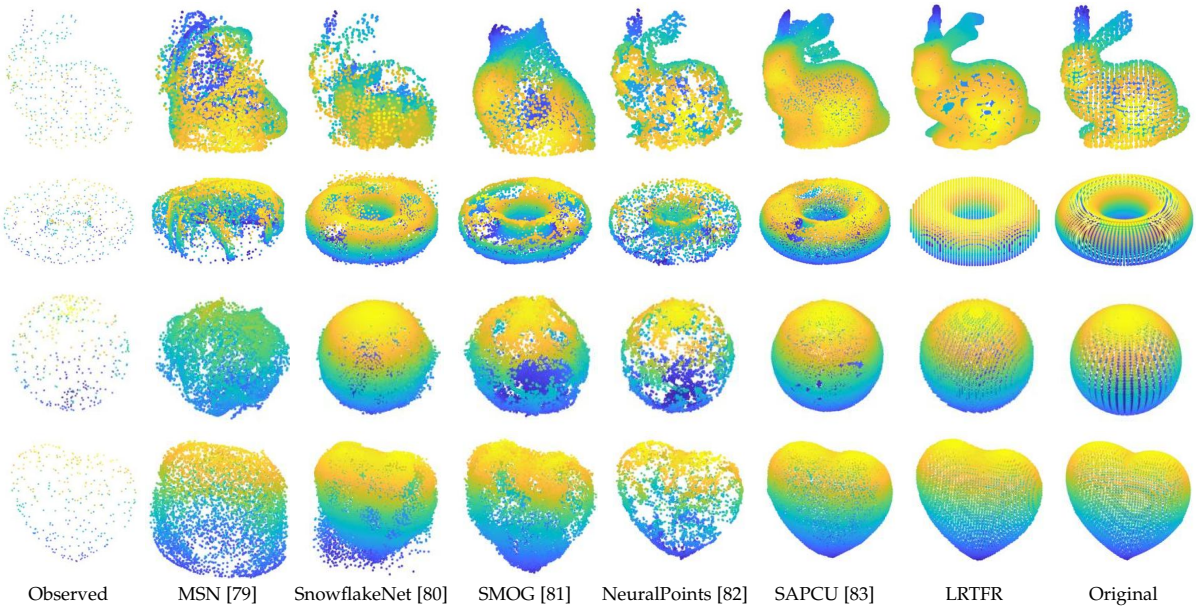



Fig. 10. The results of point cloud upsampling by different methods on *Bunny*, *Doughnut*, *Sphere*, and *Heart*.

### 5.3 Influences of Hyperparameters

Selecting suitable hyperparameter values is a necessary step in our method. These hyperparameters include the weight decay of the Adam optimizer (denoted by $w$), the hyperparameter of the sine activation $\omega_0$, the F-rank $(r_1, r_2, r_3)$, and the depth of the MLP (denoted by $d$). Meanwhile, there are two regularization parameters $\gamma_1, \gamma_2$ in the denoising model (9) and the point cloud upsampling model (11). To comprehensively analyze the influences of different hyperparameters on the performances of our method, we change the value of each hyperparameter and fix the others and report the corresponding results. The results are shown in Fig. 12 and Fig. 13. We remark that different tasks require different values of hyperparameters, and thus the testing ranges of a hyperparameter are inconsistent for different tasks. From the results we can see that our method is relatively robust w.r.t. these hyperparameters since it can obtain satisfactory performances for a wide range of values. This makes our method relatively easy to be applicable in real scenarios. Moreover, from Fig. 13 we can see that the adopted regularizations (e.g., the TV regularization in (9)) are effective to boost the performance of our method with suitable hyperparameter values, which reveals the compatibility of our method with other proven techniques to enhance performance. Anyway, how to build easy and automatic hyperparameter tuning strategies to make our method more flexible and adaptable to diverse scenarios still requires more endeavor of our future research.

## 6 Conclusions

In this work, we proposed the LRTFR for multi-dimensional data continuous representation. We formulated the continuous representation as a low-rank tensor function, and developed fundamental concepts including F-rank and tensor function factorization. We theoretically justified the low-rank and smooth regularizations hidden in LRTFR, which makes it effective and efficient for continuous representation. Extensive experiments on multi-dimensional data recovery problems including multi-dimensional image inpainting, denoising, HPO, and point cloud upsampling have validated the broad applicability and superiority of our method as compared with state-of-the-art methods. The suggested continuous representation is a potential tool for multi-dimensional data processing and analysis that can be applied to more tasks in the future, e.g., hyperspectral fusion and blind image super-resolution.

## References

[1] M. E. Kilmer, L. Horesh, H. Avron, and E. Newman, "Tensor-tensor algebra for optimal representation and compression of multiway data," *Proceedings of the National Academy of Sciences*, vol. 118, no. 28, 2021.

[2] X. Zhang and M. K. Ng, "Sparse nonnegative tensor factorization and completion with noisy observations," *IEEE Transactions on Information Theory*, vol. 68, no. 4, pp. 2551–2572, 2022.

[3] N. Vervliet, O. Debals, and L. De Lathauwer, "Exploiting efficient representations in large-scale tensor decompositions," *SIAM Journal on Scientific Computing*, vol. 41, no. 2, pp. A789–A815, 2019.

[4] J. Pan, M. K. Ng, Y. Liu, X. Zhang, and H. Yan, "Orthogonal nonnegative tucker decomposition," *SIAM Journal on Scientific Computing*, vol. 43, no. 1, pp. B55–B81, 2021.

[5] S. Gu, Q. Xie, D. Meng, W. Zuo, X. Feng, and L. Zhang, "Weighted nuclear norm minimization and its applications to low level vision," *International Journal of Computer Vision*, vol. 121, p. 183–208, 2017.

[6] J. A. Bengua, H. N. Phien, H. D. Tuan, and M. N. Do, "Efficient tensor completion for color image and video recovery: Low-rank tensor train," *IEEE Transactions on Image Processing*, vol. 26, no. 5, pp. 2466–2479, 2017.

[7] D. Zhu, H. Chen, W. Wang, H. Xie, G. Cheng, M. Wei, J. Wang, and F. L. Wang, "Nonlocal low-rank point cloud denoising for 3-D measurement surfaces," *IEEE Transactions on Instrumentation and Measurement*, 2022. doi=10.1109/TIM.2021.3139686.

[8] J. Liu, P. Musialski, P. Wonka, and J. Ye, "Tensor completion for estimating missing values in visual data," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 1, pp. 208–220, 2013.

[9] M. Yin, J. Gao, and Z. Lin, "Laplacian regularized low-rank representation and its applications," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 3, pp. 504–517, 2016.

[10] Q. Xie, Q. Zhao, D. Meng, and Z. Xu, "Kronecker-basis-representation based tensor sparsity and its applications to tensor recovery," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 8, pp. 1888–1902, 2018.

Observed PSNR 3.66 | (a) PSNR 18.64 | (b) PSNR 18.71 | (c) PSNR 20.07 | (d) PSNR 28.77 | (e) PSNR 29.75 | Original PSNR Inf

Fig. 11. The results of LRTFR with different MLP activation functions and factorization strategies for inpainting (*Plane* SR = 0.2). (a) ReLU + Tucker function factorization. (b) LeakyReLU + Tucker function factorization. (c) Tanh + Tucker function factorization. (d) Sine + CP function factorization. (e) Sine + Tucker function factorization (Our proposed).

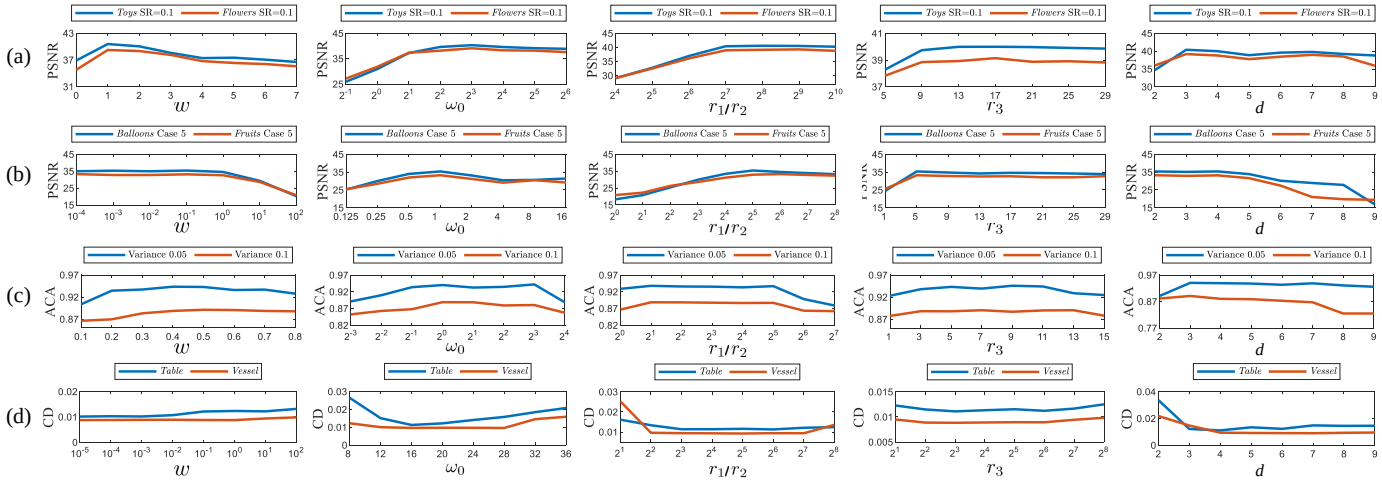



Fig. 12. The quantitative performances of LRTFR w.r.t. different values of hyperparameters (The weight decay $w$, the hyperparameter of the sine activation function $\omega_0$, the F-rank $(r_1, r_2, r_3)$, and the number of MLP layers $d$) for (a) multi-dimensional image inpainting, (b) MSI denoising, (c) hyperparameter optimization, and (d) point cloud upsampling.

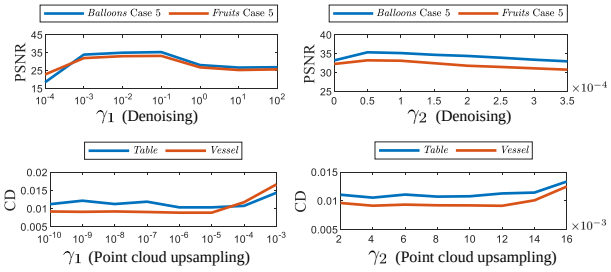



Fig. 13. The quantitative performances of LRTFR w.r.t. different values of regularization parameters in the denoising model (9) and point cloud upsampling model (11).


[11] T. G. Kolda and B. W. Bader, "Tensor decompositions and applications," *SIAM Review*, vol. 51, no. 3, pp. 455–500, 2009.

[12] N. D. Sidiropoulos, L. De Lathauwer, X. Fu, K. Huang, E. E. Papalexakis, and C. Faloutsos, "Tensor decomposition for signal processing and machine learning," *IEEE Transactions on Signal Processing*, vol. 65, no. 13, pp. 3551–3582, 2017.

[13] Y. Wang, J. Peng, Q. Zhao, Y. Leung, X.-L. Zhao, and D. Meng, "Hyperspectral image restoration via total variation regularized low-rank tensor decomposition," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 11, no. 4, pp. 1227–1243, 2018.

[14] Z. Chen, Z. Xu, and D. Wang, "Deep transfer tensor decomposition with orthogonal constraint for recommender systems," in *AAAI Conference on Artificial Intelligence (AAAI)*, pp. 4010–4018, 2021.

[15] J.-T. Chien and Y.-T. Bao, "Tensor-factorized neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 5, pp. 1998–2011, 2018.

[16] C. Li and Z. Sun, "Evolutionary topology search for tensor network decomposition," in *International Conference on Machine Learning (ICML)*, vol. 119, pp. 5947–5957, 2020.

[17] I. V. Oseledets, "Tensor-train decomposition," *SIAM Journal on Scientific Computing*, vol. 33, no. 5, pp. 2295–2317, 2011.

[18] X. P. Li and H. C. So, "Robust low-rank tensor completion based on tensor ring rank via $\ell_{p,\epsilon}$-norm," *IEEE Transactions on Signal Processing*, vol. 69, pp. 3685–3698, 2021.

[19] L. Grasedyck, "Hierarchical singular value decomposition of tensors," *SIAM Journal on Matrix Analysis and Applications*, vol. 31, no. 4, pp. 2029–2054, 2010.

[20] W. Wang, V. Aggarwal, and S. Aeron, "Efficient low rank tensor ring completion," in *IEEE International Conference on Computer Vision (ICCV)*, pp. 5698–5706, 2017.

[21] Y.-B. Zheng, T.-Z. Huang, X.-L. Zhao, Q. Zhao, and T.-X. Jiang, "Fully-connected tensor network decomposition and its application to higher-order tensor completion," in *AAAI Conference on Artificial Intelligence (AAAI)*, pp. 11071–11078, 2021.

[22] M. E. Kilmer, K. Braman, N. Hao, and R. C. Hoover, "Third-order tensors as operators on matrices: A theoretical and computational framework with applications in imaging," *SIAM Journal on Matrix Analysis and Applications*, vol. 34, no. 1, pp. 148–172, 2013.

[23] M. E. Kilmer and C. D. Martin, "Factorization strategies for third-order tensors," *Linear Algebra and its Applications*, vol. 435, no. 3, pp. 641–658, 2011.

[24] C. Lu, J. Feng, Y. Chen, W. Liu, Z. Lin, and S. Yan, "Tensor robust principal component analysis with a new tensor nuclear norm," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, no. 4, pp. 925–938, 2020.

[25] Y. Zhou and Y.-M. Cheung, "Bayesian low-tubal-rank robust tensor factorization with multi-rank determination," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, no. 1, pp. 62–76, 2021.

[26] F. Zhang, J. Wang, W. Wang, and C. Xu, "Low-tubal-rank plus sparse tensor recovery with prior subspace information," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, no. 10, pp. 3492–3507, 2021.

[27] J. Hou, F. Zhang, H. Qiu, J. Wang, Y. Wang, and D. Meng, "Robust low-tubal-rank tensor recovery from binary measurements," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 8, pp. 4355–4373, 2022.

[28] Z. Zhang, G. Ely, S. Aeron, N. Hao, and M. Kilmer, "Novel methods for multilinear data completion and de-noising based

on tensor-SVD," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3842–3849, 2014.

[29] C. Lu, X. Peng, and Y. Wei, "Low-rank tensor completion with a new tensor nuclear norm induced by invertible linear transforms," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5989–5997, 2019.

[30] P. Zhou, C. Lu, J. Feng, Z. Lin, and S. Yan, "Tensor low-rank representation for data recovery and clustering," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, no. 5, pp. 1718–1732, 2021.

[31] Y. Luo, X.-L. Zhao, D. Meng, and T.-X. Jiang, "HLRTF: Hierarchical low-rank tensor factorization for inverse problems in multi-dimensional imaging," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 19303–19312, 2022.

[32] J. Peng, Y. Wang, H. Zhang, J. Wang, and D. Meng, "Exact decomposition of joint low rankness and local smoothness plus sparse matrices," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022. doi=10.1109/TPAMI.2022.3204203.

[33] Y.-B. Zheng, T.-Z. Huang, X.-L. Zhao, Y. Chen, and W. He, "Double-factor-regularized low-rank tensor factorization for mixed noise removal in hyperspectral image," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 58, no. 12, pp. 8450–8464, 2020.

[34] M. Imaizumi and K. Hayashi, "Tensor decomposition with smoothness," in *International Conference on Machine Learning (ICML)*, vol. 70, pp. 1597–1606, 2017.

[35] Y.-S. Luo, X.-L. Zhao, T.-X. Jiang, Y. Chang, M. K. Ng, and C. Li, "Self-supervised nonlinear transform-based tensor nuclear norm for multi-dimensional image recovery," *IEEE Transactions on Image Processing*, vol. 31, pp. 3793–3808, 2022.

[36] X. Zhang and M. K. Ng, "Low rank tensor completion with poisson observations," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 8, pp. 4239–4251, 2022.

[37] J. Peng, Q. Xie, Q. Zhao, Y. Wang, L. Yee, and D. Meng, "Enhanced 3DTV regularization and its applications on hsi denoising and compressed sensing," *IEEE Transactions on Image Processing*, vol. 29, pp. 7889–7903, 2020.

[38] T. Yokota, R. Zdunek, A. Cichocki, and Y. Yamashita, "Smooth nonnegative matrix and tensor factorizations for robust multi-way data analysis," *Signal Processing*, vol. 113, pp. 234–249, 2015.

[39] O. Debals, M. Van Barel, and L. De Lathauwer, "Nonnegative matrix factorization using nonnegative polynomial approximations," *IEEE Signal Processing Letters*, vol. 24, no. 7, pp. 948–952, 2017.

[40] N. Kargas and N. D. Sidiropoulos, "Supervised learning and canonical decomposition of multivariate functions," *IEEE Transactions on Signal Processing*, vol. 69, pp. 1097–1107, 2021.

[41] Y. Chen, S. Liu, and X. Wang, "Learning continuous image representation with local implicit image function," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 8624–8634, 2021.

[42] J. J. Park, P. Florence, J. Straub, R. Newcombe, and S. Lovegrove, "DeepSDF: Learning continuous signed distance functions for shape representation," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 165–174, 2019.

[43] M. Michalkiewicz, J. K. Pontes, D. Jack, M. Baktashmotlagh, and A. Eriksson, "Implicit surface representations as layers in neural networks," in *IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 4742–4751, 2019.

[44] V. Sitzmann, J. Martel, A. Bergman, D. Lindell, and G. Wetzstein, "Implicit neural representations with periodic activation functions," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 33, pp. 7462–7473, 2020.

[45] 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 *European Conference on Computer Vision (ECCV)*, pp. 405–421, 2020.

[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 *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11353–11362, 2021.

[47] L. Yariv, J. Gu, Y. Kasten, and Y. Lipman, "Volume rendering of neural implicit surfaces," in *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 4805–4815, 2021.

[48] I. Skorokhodov, S. Ignatyev, and M. Elhoseiny, "Adversarial generation of continuous images," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10748–10759, 2021.

[49] Z. Chen and H. Zhang, "Learning implicit fields for generative shape modeling," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5932–5941, 2019.

[50] V. Sitzmann, M. Zollhöfer, and G. Wetzstein, "Scene representation networks: Continuous 3D-structure-aware neural scene representations," in *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 1121–1132, 2019.

[51] Z. Huang, S. Bai, and J. Z. Kolter, "(implicit)$^2$: Implicit layers for implicit representations," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 34, pp. 9639–9650, 2021.

[52] M. Tancik, P. P. Srinivasan, B. Mildenhall, S. Fridovich-Keil, N. Raghavan, U. Singhal, R. Ramamoorthi, J. T. Barron, and R. Ng, "Fourier features let networks learn high frequency functions in low dimensional domains," *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 7537–7547, 2020.

[53] B. Ma, Z. Han, Y.-S. Liu, and M. Zwicker, "Neural-pull: Learning signed distance function from point clouds by learning to pull space onto surface," in *International Conference on Machine Learning (ICML)*, vol. 139, pp. 7246–7257, 2021.

[54] J. Yang, S. Shen, H. Yue, and K. Li, "Implicit transformer network for screen content image continuous super-resolution," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 34, pp. 13304–13315, 2021.

[55] C. Ma, P. Yu, J. Lu, and J. Zhou, "Recovering realistic details for magnification-arbitrary image super-resolution," *IEEE Transactions on Image Processing*, vol. 31, pp. 3669–3683, 2022.

[56] X. Hu, H. Mu, X. Zhang, Z. Wang, T. Tan, and J. Sun, "Meta-SR: A magnification-arbitrary network for super-resolution," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1575–1584, 2019.

[57] C. Kim, J. Lee, and J. Shin, "Zero-shot blind image denoising via implicit neural representations," *arXiv:2204.02405*, 2022.

[58] I. V. Oseledets, "Constructive representation of functions in low-rank tensor formats," *Constructive Approximation*, vol. 37, pp. 1–18, 2013.

[59] B. Hashemi and L. N. Trefethen, "Chebfun in three dimensions," *SIAM Journal on Scientific Computing*, vol. 39, no. 5, pp. C341–C363, 2017.

[60] A. Kratsios, "The universal approximation property: Characterization, construction, representation, and existence," *Annals of Mathematics and Artificial Intelligence*, vol. 89, no. 5-6, pp. 435–469, 2021.

[61] L. Sun, W. Dong, X. Li, J. Wu, L. Li, and G. Shi, "Deep maximum a posterior estimator for video denoising," *International Journal of Computer Vision*, vol. 129, pp. 2827–2845, 2021.

[62] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *International Conference on Learning Representations (ICLR)*, 2015.

[63] T.-X. Jiang, M. K. Ng, X.-L. Zhao, and T.-Z. Huang, "Framelet representation of tensor nuclear norm for third-order tensor completion," *IEEE Transactions on Image Processing*, vol. 29, pp. 7233–7244, 2020.

[64] L. Deng and M. Xiao, "A new automatic hyperparameter recommendation approach under low-rank tensor completion framework," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022. doi=10.1109/TPAMI.2022.3195658.

[65] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *Journal of Machine Learning Research*, vol. 13, pp. 281–305, 2012.

[66] L. Yu, X. Li, C.-W. Fu, D. Cohen-Or, and P.-A. Heng, "PU-Net: Point cloud upsampling network," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2790–2799, 2018.

[67] R. Li, X. Li, C.-W. Fu, D. Cohen-Or, and P.-A. Heng, "PU-GAN: A point cloud upsampling adversarial network," in *IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 7202–7211, 2019.

[68] R. Li, X. Li, P.-A. Heng, and C.-W. Fu, "Point cloud upsampling via disentangled refinement," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 344–353, 2021.

[69] Q. Song, G. Wang, and C. Wang, "Automatic recommendation of classification algorithms based on data set characteristics," *Pattern Recognition*, vol. 45, no. 7, pp. 2672–2689, 2012.

[70] H. Fan, H. Su, and L. Guibas, "A point set generation network for 3D object reconstruction from a single image," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2463–2471, 2017.

[71] A. Knapitsch, J. Park, Q.-Y. Zhou, and V. Koltun, "Tanks and temples: Benchmarking large-scale scene reconstruction," *ACM Transactions on Graphics*, vol. 36, no. 4, 2017.

[72] 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 *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5795–5805, 2021.

[73] L. Yuan, C. Li, D. P. Mandic, J. Cao, and Q. Zhao, "Tensor ring decomposition with rank minimization on latent space: An efficient approach for tensor completion," in *AAAI Conference on Artificial Intelligence (AAAI)*, pp. 9151–9158, 2019.

[74] Z. Li, H. Wang, and D. Meng, "Regularize implicit neural representation by itself," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10280–10288, 2023.

[75] Q. Yuan, Q. Zhang, J. Li, H. Shen, and L. Zhang, "Hyperspectral image denoising employing a spatial–spectral deep residual convolutional neural network," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 57, no. 2, pp. 1205–1218, 2019.

[76] A. Maffei, J. M. Haut, M. E. Paoletti, J. Plaza, L. Bruzzone, and A. Plaza, "A single model CNN for hyperspectral image denoising," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 58, no. 4, pp. 2516–2529, 2020.

[77] F. Yasuma, T. Mitsunaga, D. Iso, and S. K. Nayar, "Generalized assorted pixel camera: Postcapture control of resolution, dynamic range, and spectrum," *IEEE Transactions on Image Processing*, vol. 19, no. 9, pp. 2241–2253, 2010.

[78] N. Liu, W. Li, R. Tao, and J. E. Fowler, "Wavelet-domain low-rank/group-sparse destriping for hyperspectral imagery," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 57, no. 12, pp. 10310–10321, 2019.

[79] M. Liu, L. Sheng, S. Yang, J. Shao, and S.-M. Hu, "Morphing and sampling network for dense point cloud completion," in *AAAI Conference on Artificial Intelligence (AAAI)*, vol. 34, pp. 11596–11603, 2020.

[80] P. Xiang, X. Wen, Y.-S. Liu, Y.-P. Cao, P. Wan, W. Zheng, and Z. Han, "SnowflakeNet: Point cloud completion by snowflake point deconvolution with skip-transformer," in *IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 5479–5489, 2021.

[81] A. Dell'Eva, M. Orsingher, and M. Bertozzi, "Arbitrary point cloud upsampling with spherical mixture of gaussians," in *International Conference on 3D Vision (3DV)*, 2022.

[82] W. Feng, J. Li, H. Cai, X. Luo, and J. Zhang, "Neural points: Point cloud representation with neural fields for arbitrary upsampling," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 18612–18621, 2022.

[83] W. Zhao, X. Liu, Z. Zhong, J. Jiang, W. Gao, G. Li, and X. Ji, "Self-supervised arbitrary-scale point clouds upsampling via implicit neural representation," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1989–1997, 2022.

[84] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, J. Xiao, L. Yi, and F. Yu, "ShapeNet: An information-rich 3D model repository," *arXiv:1512.03012*, 2015.


**Yisi Luo** received B.Sc. degree from School of Mathematical Sciences, University of Electronic Science and Technology of China, Chengdu, China, and is currently pursuing the M.Sc. degree at School of Mathematics and Statistics, Xi'an Jiaotong University, Xi'an, China. His research interests include image processing, machine learning, and multi-dimensional data analysis. He served as a regular reviewer or program committee for CVPR, ICCV, ECCV, AAAI, ACM MM, etc. More information can be found on his homepage at: https://yisiluo.github.io/

**Xile Zhao** received the M.S. and Ph.D. degrees from the University of Electronic Science and Technology of China (UESTC), Chengdu, China, in 2009 and 2012, respectively. He worked as a Postdoctoral Researcher with Prof. Michael K. Ng at Hong Kong Baptist University from 2013 to 2014. He worked as a Visiting Scholar with Prof. Jose Bioucas Dias at the University of Lisbon from 2016 to 2017. He is currently a Professor with the School of Mathematical Sciences, UESTC. His research interests include image processing, machine learning, and scientific computing. More information can be found on his homepage at: https://zhaoxile.github.io/

**Zhemin Li** is currently a Ph.D. student at the National University of Defense Technology (NUDT), Changsha, China. His research interests include image processing, machine learning, and scientific computing.

**Michael K. Ng** received the B.Sc. and M.Phil. degrees from The University of Hong Kong, Hong Kong, in 1990 and 1992, respectively, and the Ph.D. degree from The Chinese University of Hong Kong, Hong Kong, in 1995. From 1995 to 1997, he was a Research Fellow with the Computer Sciences Laboratory, The Australian National University, Canberra, ACT, Australia. From 1997 to 2005, he was an Assistant Professor/an Associate Professor with The University of Hong Kong. From 2006 to 2019, he was a Professor/Chair Professor with the Department of Mathematics, Hong Kong Baptist University, Hong Kong. He is currently a Chair Professor with the Department of Mathematics, The University of Hong Kong. His research interests include bioinformatics, image processing, scientific computing, and data mining. He serves as an editorial board member for several international journals. He was selected for the 2017 Class of Fellows of the Society for Industrial and Applied Mathematics. He received the Feng Kang Prize for his significant contributions to scientific computing.

**Deyu Meng** received the B.Sc., M.Sc., and Ph.D. degrees in 2001, 2004, and 2008, respectively, from Xi'an Jiaotong University, Xi'an, China. He is currently a professor with School of Mathematics and Statistics, Xi'an Jiaotong University, and adjunct professor with Faculty of Information Technology, The Macau University of Science and Technology. He currently serves as an associate editor of the IEEE Transactions on Pattern Analysis and Machine Intelligence, Science China-Information Sciences, and Frontiers of Computer Science. His current research interests include model-based deep learning, variational networks, and meta-learning.