# Multiway K-Clustered Tensor Approximation: Toward High-Performance Photorealistic Data-Driven Rendering

YU-TING TSAI
Yuan Ze University

This article presents a generalized sparse multilinear model, namely *multiway K-clustered tensor approximation* (MK-CTA), for synthesizing photorealistic 3D images from large-scale multidimensional visual datasets. MK-CTA extends previous tensor approximation algorithms, particularly *K-clustered tensor approximation* (K-CTA) [Tsai and Shih 2012], to partition a multidimensional dataset along *more than one* dimension into overlapped clusters. On the contrary, K-CTA only sparsely clusters a dataset along just one dimension and often fails to efficiently approximate other unclustered dimensions. By generalizing K-CTA with multiway sparse clustering, MK-CTA can be regarded as a novel sparse tensor-based model that simultaneously exploits the *intra-* and *inter-cluster* coherence among different dimensions of an input dataset. Our experiments demonstrate that MK-CTA can accurately and compactly represent various multidimensional datasets with complex and sharp visual features, including *bidirectional texture functions* (BTFs) [Dana et al. 1999], *time-varying light fields* (TVLFs) [Bando et al. 2013], and *time-varying volume data* (TVVD) [Wang et al. 2010], while easily achieving high rendering rates in practical graphics applications.

Categories and Subject Descriptors: I.3.7 [**Computer Graphics**]: Three-Dimensional Graphics and Realism—*Color, Shading, Shadowing, and Texture*; E.4 [**Data**]: Coding and Information Theory—*Data Compaction and Compression*

General Terms: Algorithms

Additional Key Words and Phrases: Real-time rendering, multidimensional data analysis, tensor approximation, multiway clustering, sparse representation

This work was supported in part by the Ministry of Science and Technology of Taiwan under Grant Numbers NSC100-2221-E-155-078, NSC101-2221-E-155-065, and MOST103-2221-E-155-031.

Author's address: Y.-T. Tsai, Department of Computer Science and Engineering, Innovation Center for Big Data and Digital Convergence, Yuan Ze University, 135 Yuandong Road, Zhongli District, Taoyuan City, Taiwan 32003, R.O.C.; email: hieicis91@hotmail.com.


DOI:http://dx.doi.org/10.1145/2753756

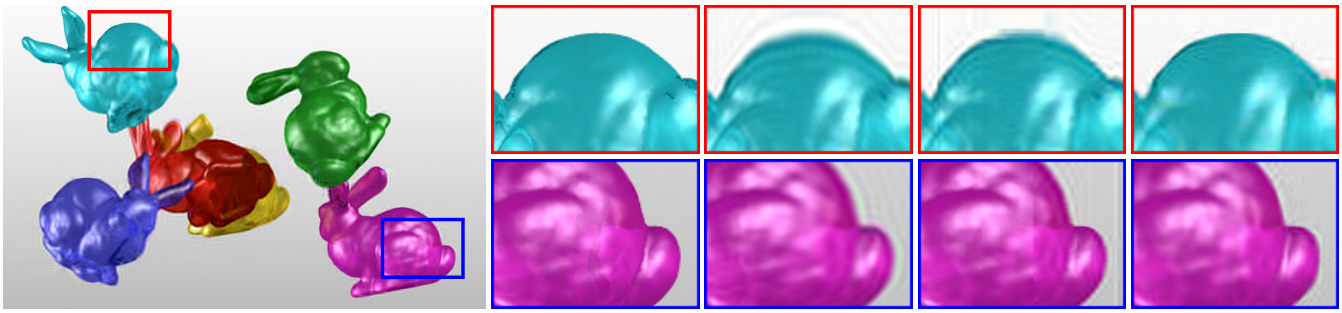
**ACM Reference Format:**
Yu-Ting Tsai. 2015. Multiway K-Clustered Tensor Approximation: Toward High-Performance Photorealistic Data-Driven Rendering. *ACM Trans. Graph.* 34, 5, Article 157 (October 2015), 15 pages.
DOI:http://dx.doi.org/10.1145/2753756

## 1. INTRODUCTION

During the last decades, data-driven rendering has shown many promising results and caught a lot of attention. Related methods not only avoid complex physically-based procedures at runtime, but also allow intuitive synthesis of photorealistic 3D images. One of the most famous and representative models may be image-based rendering [Gortler et al. 1996; Levoy and Hanrahan 1996; Shum et al. 2007]. Even the well-known texture mapping techniques can be regarded as a simple type of data-driven models. Nevertheless, high-quality results based on data-driven rendering require an enormous amount of raw visual datasets, which is usually over several gigabytes. A sophisticated approximation method for the raw data thus becomes a major research topic in data-driven rendering. Recently, there have been tremendous developments in this field [Filip and Haindl 2009; Ramamoorthi 2009], but it is still challenging to achieve a good tradeoff among fast rendering rates, high compression ratios, and low approximation errors for large-scale multidimensional visual datasets.

In this article, we propose a novel sparse multilinear model, namely MK-CTA, to solve this problem. MK-CTA seamlessly integrates multiway clustering, sparse representation, and tensor approximation. It overcomes the major drawback of previous tensor approximation algorithms to allow clustering data and analyzing coherence along more than one dimension. Specifically, MK-CTA partitions an input dataset along multiple dimensions into clusters, so that data variations within each cluster can be significantly reduced to improve intra-cluster coherence for efficient approximation. To further exploit inter-cluster coherence, MK-CTA sparsely mixes the approximated results of different clusters, particularly across different dimensions. It thus can be regarded as generalized vector quantization with overlapped blocks and varying block sizes. Elements in a block also can be transferred into another block through sparse clustering.

Note that the proposed method is not just a simple extension of previous multilinear models, such as K-CTA [Tsai and Shih 2012]. A trivial extension may be to partition the input dataset along each dimension, respectively, regardless of the correlations among different dimensions. Nevertheless, this approach only can obtain a suboptimal solution. The complex mixing effects of different dimensions particularly require generalized mathematical formulation and theorems that are different from previous multilinear models. MK-CTA also needs a *multiway* sparse clustering algorithm to account for cross-dimensional correlations and to derive a (locally) optimal solution.



(a) Rendered image of MK-CTA (b) Raw data (1.79 FPS) (c) $N$-SVD (29.03 dB, 0.05 FPS) (d) K-CTA (28.9 dB, 1.26 FPS) (e) MK-CTA (28.95 dB, 56.24 FPS)

Fig. 1. Comparison of approximated TVLFs ("AnimatedBunnies" ©Synthetic Light Field Archive) among $N$-SVD [De Lathauwer et al. 2000], K-CTA [Tsai and Shih 2012], and MK-CTA with similar compression ratios. In subfigures (b)–(e), signal-to-noise ratios and rendering performance are shown in parentheses. The rendering performance of raw data is slow due to an enormous data amount and frequent disk access operations, but still faster than that of $N$-SVD and K-CTA, while MK-CTA holds the highest rendering rate. Although the signal-to-noise ratios of the three algorithms are similar, there are strong ringing effects and some visible seams respectively in the results of $N$-SVD and K-CTA, which will be discussed more in Section 5.2. The visual quality of MK-CTA is thus still the best by manual inspection. Readers may refer to supplemental materials for full-size images and our accompanying video for animations of TVLFs.

Moreover, we demonstrate applications of MK-CTA to various multidimensional visual datasets, including BTFs [Dana et al. 1999], TVLFs [Bando et al. 2013], and TVVD [Wang et al. 2010]. Experimental results reveal that the approximated data of MK-CTA are compact and accurate. It is also simple for MK-CTA to achieve either faster rendering rates (with similar compression ratios) or higher visual quality (with similar runtime performance) than previous multilinear models, thus providing a better tradeoff among rendering performance, visual quality, and memory requirements.

## 2. RELATED WORK

### 2.1 Tensor Approximation

Recently, tensor approximation (or multilinear models) [De Lathauwer et al. 2000; Kolda and Bader 2009] has drawn a lot of attention in computer graphics. Due to its powerful approximation efficiency and flexibility, tensor approximation is suitable for representing large-scale multidimensional visual datasets. Vasilescu and Terzopoulos [2004] introduced *TensorTextures* to model BTFs based on $N$-SVD [De Lathauwer et al. 2000]. It was soon extended into an efficient out-of-core method by Wang et al. [2005].

Advanced multilinear models have also been developed by combining $N$-SVD with various sophisticated data analysis algorithms. *Clustered tensor approximation* (CTA) [Sun et al. 2007; Tsai and Shih 2006] relies on clustering to reduce intra-cluster data variations, so that the approximation efficiency of $N$-SVD within each cluster can be increased. K-CTA [Tsai and Shih 2012] further extends CTA with sparse representation to improve visual quality, while maintaining similar runtime performance. Moreover, Wu et al. [2008] proposed a hierarchical tensor representation by integrating multiresolution analysis with $N$-SVD to capture multiscale visual features in an input dataset. Suter et al. [2011] presented a GPU-based rendering framework for large-scale volume data using hierarchical brick tensor decomposition.

Nevertheless, the rendering performance of previous multilinear models may be very slow when data variations are large among multiple dimensions. In this case, CTA and K-CTA only can reduce variations along just one specified dimension by clustering, while still relying on $N$-SVD to analyze the coherence among other dimensions. Moreover, data with large variations often contain many high-frequency visual features. For hierarchical tensor approximation, these features are captured only at fine scales and may be lost at runtime when rendering performance is a major concern. By contrast, MK-CTA can intrinsically handle this challenging issue by multiway clustering to provide high rendering rates at runtime.

### 2.2 Sparse Representation

Sparse representation [Elad et al. 2010; Wright et al. 2010] targets at modeling data as linear combinations of just a few atoms in a dictionary. This simple and intuitive idea has been shown to work very well in practice, since real-world signals are often dominated by only a few factors. In computer graphics, sparse representation has been applied to solve many problems. Ruiters and Klein [2009] introduced sparse tensor decomposition to compress BTFs. Tsai and Shih [2012] proposed K-CTA as a multilinear generalization of K-SVD [Aharon et al. 2006] to approximate multidimensional visual datasets. Compressed sensing [Starck and Bobin 2010; Wright et al. 2010], which exploits the concept of sparsity to efficiently reconstruct signals, has also been employed to improve light transport acquisition [Peers et al. 2009], 3D model reconstruction [Avron et al. 2010], and ray tracing [Sen and Darabi 2011].

We believe that sparse representation is promising for data-driven rendering. Due to its merit that only a few atoms are related to an observation, the runtime process can be very efficient on GPUs. Nevertheless, previous sparse models are inadequate to high-performance rendering. While it is difficult for sparse tensor decomposition to achieve high rendering performance, K-CTA may need to sacrifice visual quality for real-time rendering rates when data variations are large among multiple dimensions. By contrast, MK-CTA can achieve a better tradeoff between visual quality and rendering performance than previous methods.

### 2.3 Multiway Clustering

Multiway clustering [Banerjee et al. 2007; Bekkerman et al. 2005; Shashua et al. 2006] generalizes traditional clustering methods to partition data based on correlations among different dimensions. It exploits these multiway correlations to form a high-quality and efficient clustering algorithm for large-scale multidimensional

datasets. Nevertheless, little attention has been paid to multiway clustering in computer graphics. Hašan et al. [2008] applied multiway clustering to solve global illumination issues for dynamic scenes, so that temporal coherence can be exploited to avoid flickering from sparsely sampled pixels and lights. Havran et al. [2010] proposed to approximate BTFs separately for individual dimensions based on multilevel vector quantization, which also can be considered as a variant of multiway clustering. Sun et al. [2011] introduced an all-frequency bi-scale radiance transfer algorithm by employing biclustering, namely two-way clustering, to compress precomputed transfer matrices.

Related articles in computer graphics generally focus on common multiway clustering approaches that linearly represent partitioned data. By contrast, we present a novel clustering algorithm that combines multilinear models and sparse representation with multiway clustering. MK-CTA thus can be regarded as a unified learning/analysis framework for large-scale multidimensional datasets.

## 3. MK-CTA ALGORITHM

### 3.1 Notation

In this article, we follow the basic definitions of tensor algebra in [De Lathauwer et al. 2000]. For simplicity, the in-core tensor notation is adopted, even if the out-of-core algorithm [Wang et al. 2005] was actually employed in our implementation.

An $N$-th order tensor $\mathcal{A} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ is a high order generalization of a vector (a first order tensor). It can be defined as an $N$-dimensional array, where each dimension corresponds to a distinct mode in tensor algebra. Figure 2(a) shows an example of a second order tensor (a matrix).

The transpose of a matrix $\mathbf{U} \in \mathbb{R}^{I \times J}$ is written as $\mathbf{U}^T$, and the row vector $(\mathbf{U})_{i*}$ denotes the $i$-th row of $\mathbf{U}$. The scalar $(\mathbf{U})_{ij}$ denotes the element in row $i$ and column $j$ of $\mathbf{U}$; similarly, $(\mathcal{A})_{i_1 \cdots i_N}$ is an element of $\mathcal{A}$. The $i$-th mode-$n$ sub-tensor [Tsai and Shih 2012] of $\mathcal{A}$ is denoted by $\mathcal{A}_{\langle n,i \rangle} \in \mathbb{R}^{I_1 \times \cdots \times I_{n-1} \times 1 \times I_{n+1} \times \cdots \times I_N}$, whose elements are

$$\left(\mathcal{A}_{\langle n,i \rangle}\right)_{i_1 \cdots i_{n-1}\, 1\, i_{n+1} \cdots i_N} = (\mathcal{A})_{i_1 \cdots i_{n-1}\, i\, i_{n+1} \cdots i_N}. \qquad (1)$$

Let $\langle \mathcal{A}, \mathcal{B} \rangle$ represent the scalar product of two $N$-th order tensors $\mathcal{A}, \mathcal{B} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$, and $\|\mathcal{A}\|_F = \sqrt{\langle \mathcal{A}, \mathcal{A} \rangle}$ be the Frobenius norm of $\mathcal{A}$. The symbol $uf_n(\mathcal{A}) \in \mathbb{R}^{I_n \times (I_{n+1} \cdots I_N I_1 \cdots I_{n-1})}$ denotes the mode-$n$ unfolded matrix of $\mathcal{A}$, which results from retaining the $n$-th mode of $\mathcal{A}$ and flattening the others (refer to [De Lathauwer et al. 2000, Figure 2.1]). The mode-$n$ product of $\mathcal{A}$ and a matrix $\mathbf{U} \in \mathbb{R}^{J_n \times I_n}$ is denoted by $\mathcal{B} = \mathcal{A} \times_n \mathbf{U}$. A series of mode-$n$ products is defined as

$$\mathcal{B} = \mathcal{A} \bigtimes_{n=1}^{N} \mathbf{U}_n = \mathcal{A} \times_1 \mathbf{U}_1 \cdots \times_N \mathbf{U}_N. \qquad (2)$$

### 3.2 Key Ideas

K-CTA combines tensor approximation, clustering, and sparse representation to improve rendering performance over $N$-SVD and visual quality over CTA in real-time applications. Along the clustered mode $m$, it sparsely classifies each mode-$m$ sub-tensor of an $N$-th order tensor $\mathcal{A}$ into more than one cluster, and then decomposes sub-tensors within a cluster using $N$-SVD. For example, Figure 2(c) illustrates that each mode-1 sub-tensor, namely each row, of a second order tensor is respectively classified along the first mode into two clusters, resulting in three overlapped clusters. In this way, the data coherence within each cluster and among different clusters can be simultaneously exploited.

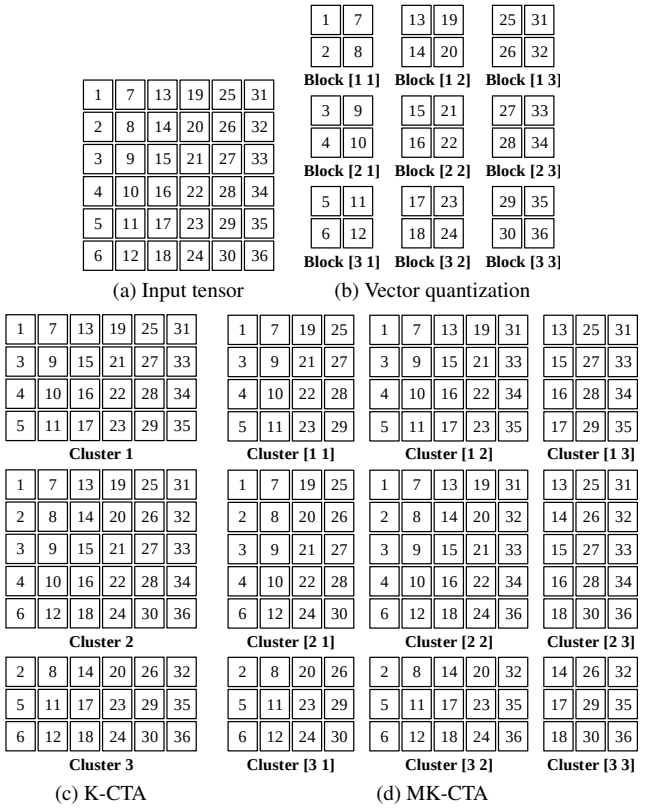



Fig. 2. Comparison of traditional vector quantization, K-CTA, and MK-CTA for a second order tensor (also as a matrix or a 2D grayscale image). In each subfigure, a box is a single tensor element, and the number inside specifies its identity. Vector quantization partitions the input tensor into disjoint blocks/clusters with fixed membership and the same block size, for instance, 2×2 in (b). K-CTA instead allows overlapped clusters with varying sizes and members, but only one mode can be clustered, such as the first mode in (c). By contrast, MK-CTA can split the tensor along multiple modes and associate a tensor element with clusters across different modes, for example in (d), the element "1" with clusters $\begin{bmatrix}1 & 1\end{bmatrix}$, $\begin{bmatrix}1 & 2\end{bmatrix}$, $\begin{bmatrix}2 & 1\end{bmatrix}$, and $\begin{bmatrix}2 & 2\end{bmatrix}$.

Although K-CTA can exploit both intra- and inter-cluster coherence to reduce approximation errors and reconstruction costs, it only partitions $\mathcal{A}$ along just one user-specified clustered mode. If there are large data variations among more than one mode, K-CTA will fail to efficiently approximate sub-tensors within each cluster. To solve this problem, an intuitive way is to sequentially partition $\mathcal{A}$ along multiple modes into sparsely overlapped clusters for subsequent tensor decomposition. Nevertheless, this simple idea is far from easy to achieve. If the mixing correlations among different modes and clusters are ignored, the derived solution is actually not (locally) optimal. Thus, we develop a multiway sparse clustering algorithm based on multilinear models to account for the complex mixing correlations.

To intuitively explain our key ideas, Figure 2 compares MK-CTA to traditional vector quantization and K-CTA for a second order

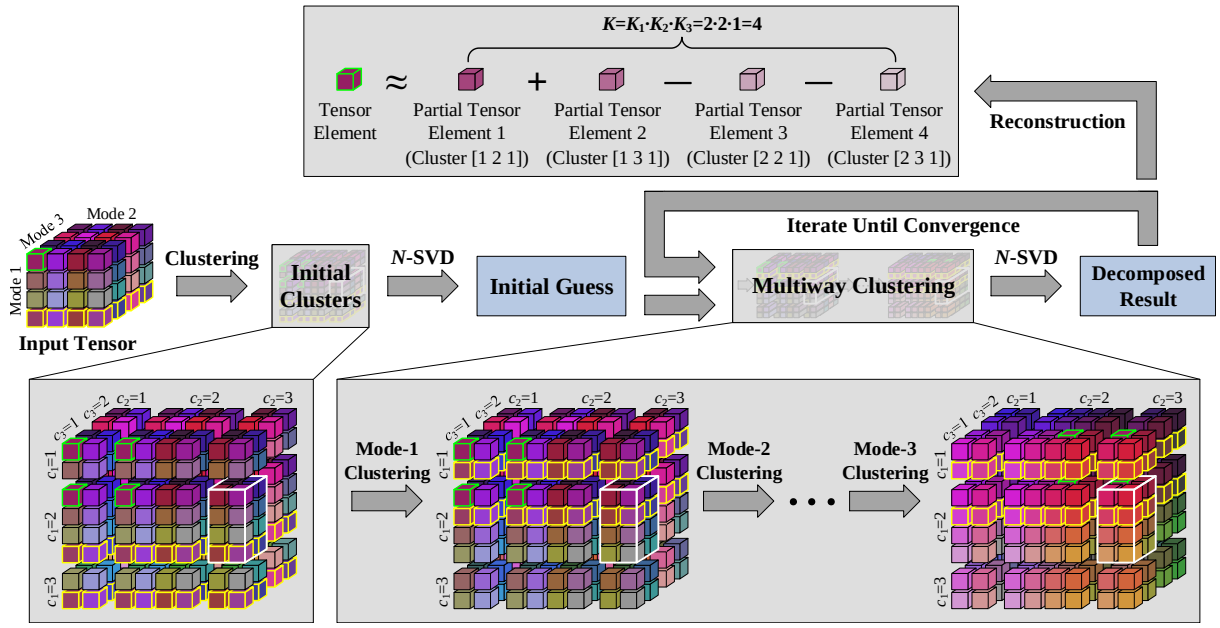



Fig. 3. An example of MK-CTA for a third order tensor. The input tensor is respectively partitioned along its three modes into $C_1 = 3$, $C_2 = 3$, and $C_3 = 2$ clusters, with $K_1 = 2$, $K_2 = 2$, and $K_3 = 1$ mixture clusters for each mode. Thus, the total number of clusters is $C_1 \cdot C_2 \cdot C_3 = 18$, and each cluster is indexed by a vector $\begin{bmatrix} c_1 & c_2 & c_3 \end{bmatrix}$ (highlighted in white for the cluster $\begin{bmatrix} 2 & 3 & 1 \end{bmatrix}$). In our framework, all the elements in a mode-$n$ sub-tensor should be assigned to the same $K_n$ mode-$n$ cluster subsets. For example, the elements in the last mode-1 sub-tensor (highlighted in yellow) are assigned to the first and the second mode-1 cluster subsets after mode-1 clustering (in the middle bottom), where the $j$-th mode-1 cluster subset is the set of all the cluster index vectors with $c_1 = j$, namely clusters $\begin{bmatrix} j & 1 & 1 \end{bmatrix}$, $\begin{bmatrix} j & 1 & 2 \end{bmatrix}$, $\begin{bmatrix} j & 2 & 1 \end{bmatrix}$, $\begin{bmatrix} j & 2 & 2 \end{bmatrix}$, $\begin{bmatrix} j & 3 & 1 \end{bmatrix}$, and $\begin{bmatrix} j & 3 & 2 \end{bmatrix}$. Note that clusters may be overlapped if $\exists n, K_n > 1$, such as clusters $\begin{bmatrix} 1 & 1 & 1 \end{bmatrix}$ and $\begin{bmatrix} 2 & 1 & 1 \end{bmatrix}$ after mode-1 clustering. Each element in the tensor (highlighted in green for an element) is respectively related to total $K_1 \cdot K_2 \cdot K_3 = 4$ clusters, and can be approximated by a linear combination of the reconstructed results of these clusters (partial tensor elements in the top).

tensor. Figure 3 further illustrates an example of MK-CTA for a third order tensor.

## 3.3 Mathematical Formulation

To allow sparse clustering along multiple modes, MK-CTA is formulated as the following constrained least-squares optimization problem:

$$\min_{\{\mathcal{Z}_{\mathbf{c}}, \{\mathbf{U}_{n,\mathbf{c}}\}_{n=1}^{N}\}_{\mathbf{c}\in\mathbf{C}}} \left\| \mathcal{A} - \sum_{\mathbf{c}\in\mathbf{C}} \left( \mathcal{Z}_{\mathbf{c}} \times_{n=1}^{N} \mathbf{U}_{n,\mathbf{c}} \right) \right\|_F^2, \text{ s. t.}$$

$$\begin{cases} \forall i, \forall n, \forall c_1, \cdots, \forall c_{n-1}, \forall c_{n+1}, \cdots, \forall c_N, \\ \qquad \sum_{c_n=1}^{C_n} \left\| \left(\mathbf{U}_{n,\mathbf{c}}\right)_{i*} \right\|_0 = K_n R_n, \\ \forall i, \forall n, \forall \mathbf{c}, \ \left\| \left(\mathbf{U}_{n,\mathbf{c}}\right)_{i*} \right\|_0 \in \{0, R_n\}, \\ \forall i, \forall j, \forall n, \forall \mathbf{c}, \mathbf{c}' \in \mathbf{C}_{n,j}, \ \left\| \left(\mathbf{U}_{n,\mathbf{c}}\right)_{i*} \right\|_0 = \left\| \left(\mathbf{U}_{n,\mathbf{c}'}\right)_{i*} \right\|_0, \\ \forall n, \forall \mathbf{c}, \ \mathbf{U}_{n,\mathbf{c}}^T \mathbf{U}_{n,\mathbf{c}} = \mathbf{I}_{R_n}, \end{cases} \quad (3)$$

where $C_n \in \mathbb{Z}^+$ and $K_n \in \mathbb{Z}^+$ are the total numbers of clusters and mixture clusters for the $n$-th mode of $\mathcal{A}$, $R_n \in \mathbb{Z}^+$ represents the mode-$n$ reduced rank, $\mathcal{Z}_{\mathbf{c}} \in \mathbb{R}^{R_1 \times \cdots \times R_N}$ and $\mathbf{U}_{n,\mathbf{c}} \in \mathbb{R}^{I_n \times R_n}$ respectively denote the decomposed core tensor and the mode-$n$ basis matrix of a cluster $\mathbf{c}$, $\mathbf{I}_{R_n} \in \mathbb{R}^{R_n \times R_n}$ is the identity matrix of size $R_n \times R_n$, and $\|\cdot\|_0$ specifies the $\ell^0$ norm of a vector. Note that in our formulation, a cluster is indexed by a vector $\mathbf{c} = \begin{bmatrix} c_1 & \cdots & c_N \end{bmatrix}$, where $c_n \in \{1, \ldots, C_n\}$ is the mode-$n$ cluster index. For example, the index vector of the cluster highlighted in white in Figure 3 is $\begin{bmatrix} 2 & 3 & 1 \end{bmatrix}$. Moreover, $\mathbf{C}$ represents the set of all possible cluster index vectors, and $\mathbf{C}_{n,j}$ denotes the $j$-th mode-$n$ cluster subset, which includes all the cluster index vectors whose $n$-th elements are $j$, namely $c_n = j$.

Eq. (3) is a generalization of the K-CTA formulation [Tsai and Shih 2012]. The first, second, and fourth constraints similarly enforce that all the mode-$n$ basis matrices are column-orthonormal and sparse, containing the nonzero mixing coefficients of each mode-$n$ sub-tensor. Specifically, the mixing coefficients of the $i$-th mode-$n$ sub-tensor for a cluster $\mathbf{c}$ corresponds to the elements in row $i$ of $\mathbf{U}_{n,\mathbf{c}}$. Unlike K-CTA, the third constraint further restricts that all the elements in a mode-$n$ sub-tensor should be classified into the same $K_n$ mode-$n$ cluster subsets. This means that we do not apply hierarchical clustering in order to simplify the proposed algorithm. As a result, each tensor element belongs to total $\prod_{n=1}^{N} K_n$ clusters that are the intersection of all its associated cluster subsets. For example, the tensor element highlighted in green in Figure 3 is associated with cluster subsets $\mathbf{C}_{1,1}$, $\mathbf{C}_{1,2}$, $\mathbf{C}_{2,2}$, $\mathbf{C}_{2,3}$, and $\mathbf{C}_{3,1}$ after mode-3 clustering (in the right bottom). It thus belongs to clusters $\begin{bmatrix} 1 & 2 & 1 \end{bmatrix}$, $\begin{bmatrix} 1 & 3 & 1 \end{bmatrix}$, $\begin{bmatrix} 2 & 2 & 1 \end{bmatrix}$, and $\begin{bmatrix} 2 & 3 & 1 \end{bmatrix}$.

## 3.4 Algorithm Overview

The proposed MK-CTA algorithm iteratively alternates between two stages, namely *clustering* and *decomposition* stages, until convergence to derive a (locally) optimal solution to Eq. (3).

In the clustering stage (Section 3.5), the input tensor $\mathcal{A}$ is partitioned along each mode. Since it would be numerically intractable to simultaneously partition all the modes of $\mathcal{A}$, we instead sequen-

tially handle each mode one by one to simplify the proposed algorithm, while fixing the results of other modes and taking them into account. To cluster the $n$-th mode of $\mathcal{A}$, all variables in Eq. (3) are fixed, except for the mode-$n$ basis matrix of each cluster. Then, each mode-$n$ sub-tensor is assigned to the best $K_n$ mode-$n$ cluster subsets using either optimal search (Section 3.5.1) or greedy search (Section 3.5.2), while the nonzero elements in the mode-$n$ basis matrices are solved by minimizing the approximation error of each mode-$n$ sub-tensor.

In the decomposition stage (Section 3.6), the core tensor and basis matrices of each cluster are decomposed using $N$-SVD without altering cluster membership. Since members of different clusters may be overlapped, we decompose one cluster at a time, while leaving others unchanged. Readers may refer to Algorithm 1 for the pseudocode of MK-CTA.

**Algorithm 1:** The MK-CTA algorithm.

**Input**: $\mathcal{A}$, $\{R_n\}_{n=1}^N$, $\{C_n\}_{n=1}^N$, $\{K_n\}_{n=1}^N$, initial guess for $\left\{\mathcal{Z}_{\mathbf{c}}, \{\mathbf{U}_{n,\mathbf{c}}\}_{n=1}^N\right\}_{\mathbf{c}\in\mathbf{C}}$, and convergence threshold $\epsilon$
**Output**: $\left\{\mathcal{Z}_{\mathbf{c}}, \{\mathbf{U}_{n,\mathbf{c}}\}_{n=1}^N\right\}_{\mathbf{c}\in\mathbf{C}}$
**repeat**
  **foreach** $\mathbf{c}\in\mathbf{C}$ **do** $\mathcal{Z}'_{\mathbf{c}} \leftarrow \mathcal{Z}_{\mathbf{c}}$
  *// Clustering stage*
  **for** $n \leftarrow 1$ **to** $N$ **do** *// Mode-$n$ clustering*
    **foreach** $\mathbf{c}\in\mathbf{C}$ **do** Initialize each element of $\mathbf{U}_{n,\mathbf{c}}$ to zero
    Update $\{\mathbf{U}_{n,\mathbf{c}}\}_{\mathbf{c}\in\mathbf{C}}$ based on optimal or greedy search
    **foreach** $\mathbf{c}\in\mathbf{C}$ **do** Perform post-process (Section 3.5.3)
  *// Decomposition stage*
  **foreach** $\mathbf{c}\in\mathbf{C}$ **do**
    Compute the residual tensor $\mathcal{R}_{\mathbf{c}}$ and membership matrices $\{\mathbf{M}_{n,\mathbf{c}}\}_{n=1}^N$ of cluster $\mathbf{c}$ (Section 3.6)
    Apply $N$-SVD to $\mathcal{R}_{\mathbf{c}}$ for updating $\mathcal{Z}_{\mathbf{c}}$ and $\{\mathbf{U}_{n,\mathbf{c}}\}_{n=1}^N$
    **for** $n \leftarrow 1$ **to** $N$ **do** $\mathbf{U}_{n,\mathbf{c}} \leftarrow \mathbf{M}_{n,\mathbf{c}}\mathbf{U}_{n,\mathbf{c}}$
**until** $\sum_{\mathbf{c}\in\mathbf{C}} \frac{\left|\|\mathcal{Z}_{\mathbf{c}}\|_F^2 - \|\mathcal{Z}'_{\mathbf{c}}\|_F^2\right|}{\|\mathcal{A}\|_F^2} < \epsilon$

## 3.5 Clustering Stage

In this stage, sparse clustering is sequentially performed along each mode of $\mathcal{A}$, from the first mode to the $N$-th one. To cluster the $n$-th mode, we respectively determine exactly $K_n$ mode-$n$ cluster indices for each mode-$n$ sub-tensor, and compute the corresponding nonzero elements in the mode-$n$ basis matrix of each cluster by minimizing the approximation errors of mode-$n$ sub-tensors in the least-squares sense.

As discussed in previous work [Tsai and Shih 2012], sparse clustering along the $n$-th mode can be considered as a pursuit problem that is NP-hard [Davis et al. 1997]. In addition, the complex correlations among different modes make the pursuit problem even more difficult. We therefore propose two approaches to solve this issue. If the optimal solution is desired, optimal search (Section 3.5.1) should be applied to update the cluster membership and mode-$n$ basis matrices. If an approximate solution is acceptable, or one would like to incrementally classify a mode-$n$ sub-tensor into additional mode-$n$ cluster subsets, greedy search (Section 3.5.2) should be employed instead. Note that the convergence problem may occur when greedy search is adopted, which will be further discussed in Section 5.4.

3.5.1 *Optimal Search.* To cluster the $n$-th mode, optimal search derives the mode-$n$ basis matrix of each cluster using a brute-force approach. By fixing all the variables in Eq. (3) other than the mode-$n$ basis matrices, we first find the best $K_n$ mode-$n$ cluster indices for each mode-$n$ sub-tensor, and then respectively solve the corresponding rows of each mode-$n$ sub-tensor in the mode-$n$ basis matrices.

More formally, the best $K_n$ mode-$n$ cluster indices for the $i$-th mode-$n$ sub-tensor $\mathcal{A}_{\langle n,i\rangle}$, namely $c_{n,i}^{(1)}, \ldots, c_{n,i}^{(K_n)}$, are determined from all possible $K_n$-combinations of $\{1, \ldots, C_n\}$ by solving the following constrained integer optimization problem:

$$\max_{\left\{c_{n,i}^{(k)}\right\}_{k=1}^{K_n}} uf_n\left(\mathcal{A}_{\langle n,i\rangle}\right)\mathbf{Z}_{n,i}^{(K_n)}\left(\mathbf{Z}_{n,i}^{(K_n)}\right)^{\dagger} uf_n\left(\mathcal{A}_{\langle n,i\rangle}\right)^T, \quad \text{s. t. } \forall k, c_{n,i}^{(k)} \in \{1, \ldots, C_n\}, \tag{4}$$

where

$$\mathbf{Z}_{n,i}^{(k)} = \left[\hat{\mathbf{Z}}_n^{(c_{n,i}^{(1)})} \ \cdots \ \hat{\mathbf{Z}}_n^{(c_{n,i}^{(k)})}\right], \tag{5}$$

$$\hat{\mathbf{Z}}_n^{(j)} = \left[uf_n\left(\hat{\mathcal{Z}}_{(\mathbf{C}_{n,j})_1}\right)^T \ \cdots \ uf_n\left(\hat{\mathcal{Z}}_{(\mathbf{C}_{n,j})_{|\mathbf{C}_{n,j}|}}\right)^T\right], \tag{6}$$

$$\hat{\mathcal{Z}}_{\mathbf{c}} = \mathcal{Z}_{\mathbf{c}} \bigtimes_{\substack{n'=1 \\ n'\neq n}}^{N} \mathbf{U}_{n',\mathbf{c}}, \tag{7}$$

the superscript "†" specifies the Moore-Penrose pseudoinverse, $(\cdot)_i$ represents the $i$-th element of a set, and $|\cdot|$ denotes the cardinality of a set.

Moreover, the corresponding rows of $\mathcal{A}_{\langle n,i\rangle}$ in the mode-$n$ basis matrices are updated by

$$\mathbf{u}_{n,i}^{(K_n)} = \left(\mathbf{Z}_{n,i}^{(K_n)}\right)^{\dagger} uf_n\left(\mathcal{A}_{\langle n,i\rangle}\right)^T, \tag{8}$$

where

$$\mathbf{u}_{n,i}^{(k)} = \left[\hat{\mathbf{u}}_{n,i}^{(c_{n,i}^{(1)})} \ \cdots \ \hat{\mathbf{u}}_{n,i}^{(c_{n,i}^{(k)})}\right]^T, \tag{9}$$

$$\hat{\mathbf{u}}_{n,i}^{(j)} = \left[\left(\mathbf{U}_{n,(\mathbf{C}_{n,j})_1}\right)_{i*} \ \cdots \ \left(\mathbf{U}_{n,(\mathbf{C}_{n,j})_{|\mathbf{C}_{n,j}|}}\right)_{i*}\right]. \tag{10}$$

Readers may further refer to Algorithm 2 for the procedure of optimal search.

**Algorithm 2:** The optimal search algorithm.

**Input**: $\mathcal{A}$, $n$, $C_n$, $K_n$, and $\left\{\mathcal{Z}_{\mathbf{c}}, \left\{\mathbf{U}_{n',\mathbf{c}}\right\}_{n'=1}^N\right\}_{\mathbf{c}\in\mathbf{C}}$
**Output**: $\{\mathbf{U}_{n,\mathbf{c}}\}_{\mathbf{c}\in\mathbf{C}}$
**for** $i \leftarrow 1$ **to** $I_n$ **do**
  Obtain $c_{n,i}^{(1)}, \ldots, c_{n,i}^{(K_n)}$ of $\mathcal{A}_{\langle n,i\rangle}$ by solving Eq. (4)
  **foreach** $\mathbf{c} \in \bigcup_{k=1}^{K_n} \mathbf{C}_{n,c_{n,i}^{(k)}}$ **do** Update $\left(\mathbf{U}_{n,\mathbf{c}}\right)_{i*}$ as Eq. (8)

In the appendix, we show the mathematical proofs of Eq. (8) and the objective function in Eq. (4). Readers may find out that Eq. (8) is similar to non-orthogonal projection in linear algebra or the optimal projection method in K-CTA, particularly Eq. (15) in [Tsai

and Shih 2012]. The matrix $\left(\mathbf{Z}_{n,i}^{(K_n)}\right)^{\dagger}$ plays the same role as a projection operator that accounts for the correlations between the subspaces of any two clusters in the selected $K_n$ mode-$n$ cluster subsets. Unlike K-CTA, MK-CTA must further consider the clustered results of other modes in order to compute the optimal solution to $c_{n,i}^{(1)}, \ldots, c_{n,i}^{(K_n)}$. In other words, to assign a mode-$n$ sub-tensor into a mode-$n$ cluster subset, elements in the sub-tensor should be partitioned according to the clustered results of other modes. For $\boldsymbol{\mathcal{A}}_{\langle n,i\rangle}$, this can be implicitly handled by the multiplication of $uf_n\big(\boldsymbol{\mathcal{A}}_{\langle n,i\rangle}\big)$ with $\mathbf{Z}_{n,i}^{(K_n)}$, as shown in the objective function in Eq. (4).

3.5.2 *Greedy Search.* For the $n$-th mode, when either the number of clusters $C_n$ or the number of mixture clusters $K_n$ is large, applying optimal search to update the mode-$n$ basis matrix of each cluster may be impractically time-consuming. Fortunately, greedy approaches for the pursuit problem often provide appropriate approximate solutions [Aharon et al. 2006; Davis et al. 1997]. In addition to optimal search, we thus propose a greedy algorithm to derive an approximate solution to the mode-$n$ basis matrices. For each mode-$n$ sub-tensor, the proposed greedy algorithm relies on iteratively searching for the best $K_n$ mode-$n$ cluster indices. At each iteration, we only resolve one of them, while keeping previously selected cluster indices unaltered, but their corresponding rows in the mode-$n$ basis matrices are allowed to change values.

Specifically, for the $i$-th mode-$n$ sub-tensor $\boldsymbol{\mathcal{A}}_{\langle n,i\rangle}$, we sequentially determine $c_{n,i}^{(1)}, \ldots, c_{n,i}^{(K_n)}$ at each iteration. The $k$-th mode-$n$ cluster index $c_{n,i}^{(k)}$ is resolved at the $k$-th iteration from previously selected $k-1$ mode-$n$ cluster indices $c_{n,i}^{(1)}, \ldots, c_{n,i}^{(k-1)}$. Namely, $c_{n,i}^{(k)}$ is determined by solving the following constrained integer optimization problem:

$$\begin{aligned} \max_{c_{n,i}^{(k)}} \; & uf_n\big(\boldsymbol{\mathcal{A}}_{\langle n,i\rangle}\big)\mathbf{Z}_{n,i}^{(k)}\big(\mathbf{Z}_{n,i}^{(k)}\big)^{\dagger} uf_n\big(\boldsymbol{\mathcal{A}}_{\langle n,i\rangle}\big)^T, \\ & \text{s. t. } c_{n,i}^{(k)} \in \{1,\ldots,C_n\},\; c_{n,i}^{(k)} \notin \big\{c_{n,i}^{(1)},\ldots,c_{n,i}^{(k-1)}\big\}. \end{aligned} \tag{11}$$

Moreover, readers may refer to Algorithm 3 for the pseudocode of greedy search.

**Algorithm 3:** The greedy search algorithm.

**Input**: $\boldsymbol{\mathcal{A}}$, $n$, $C_n$, $K_n$, and $\big\{\boldsymbol{\mathcal{Z}}_{\mathbf{c}}, \big\{\mathbf{U}_{n',\mathbf{c}}\big\}_{n'=1}^{N}\big\}_{\mathbf{c}\in\mathbf{C}}$
**Output**: $\big\{\mathbf{U}_{n,\mathbf{c}}\big\}_{\mathbf{c}\in\mathbf{C}}$
**for** $i \leftarrow 1$ **to** $I_n$ **do**
  **for** $k \leftarrow 1$ **to** $K_n$ **do** Obtain $c_{n,i}^{(k)}$ of $\boldsymbol{\mathcal{A}}_{\langle n,i\rangle}$ by solving Eq. (11)
  **foreach** $\mathbf{c} \in \bigcup_{k=1}^{K_n} \mathbf{C}_{n,c_{n,i}^{(k)}}$ **do** Update $\big(\mathbf{U}_{n,\mathbf{c}}\big)_{i*}$ as Eq. (8)

It is worth noted that the objective function in Eq. (11) is in fact a straightforward modification of that in Eq. (4), but quite different from the objective function of deriving remaining mixture clusters in K-CTA, namely Theorem 3 in [Tsai and Shih 2012]. Like K-CTA, MK-CTA greedily selects the $k$-th cluster index by simultaneously maximizing intra-cluster coherence and minimizing inter-cluster coherence. Nevertheless, MK-CTA resolves each cluster index by minimizing the approximation error of $\boldsymbol{\mathcal{A}}_{\langle n,i\rangle}$, while K-CTA only minimizes the *residual* error of $\boldsymbol{\mathcal{A}}_{\langle n,i\rangle}$. In other words, when determining the $k$-th cluster index, K-CTA fixes the corresponding rows of previously selected cluster indices in the mode-$n$ basis matrices, but MK-CTA allows them to change values. This is similar to the difference between *orthogonal matching pursuit* (OMP) and *orthogonal least squares* (OLS) for the pursuit problem[1] [Blumensath and Davies 2007; Soussen et al. 2013]. The proposed greedy algorithm thus can be regarded as a multilinear generalization of OLS, in contrast to the search algorithm of K-CTA as generalized OMP. Since it was reported that OLS is usually more robust and accurate than OMP [Soussen et al. 2013], MK-CTA also holds the merit of OLS. In addition, MK-CTA further takes the clustered results of other modes into account, by the matrix $\mathbf{Z}_{n,i}^{(k)}$ and its Moore-Penrose pseudoinverse, in order to prevent a poor solution.

3.5.3 *Post-Process.* The derived mode-$n$ basis matrices by optimal (or greedy) search are not guaranteed to be column-orthonormal after clustering the $n$-th mode of $\boldsymbol{\mathcal{A}}$, so that the orthonormal constraints on basis matrices in Eq. (3) may be violated. To orthonormalize the mode-$n$ basis matrix of each cluster, we perform an additional post-process using singular value decomposition, which is the same as that described in [Tsai and Shih 2012, Section 4.2.3]. Note that this post-process only removes the scale ambiguity and has no other effects on the derived solution.

## 3.6 Decomposition Stage

After sequentially clustering each mode of $\boldsymbol{\mathcal{A}}$, we would like to further update the core tensor and basis matrices of each cluster using $N$-SVD, while fixing the cluster membership of each tensor element. Nevertheless, different clusters may be overlapped with each other, and decomposing overlapped ones at the same time will lead to a (locally) suboptimal solution. Only clusters that do not contain the same tensor element can be simultaneously decomposed. We thus alternate among all clusters to respectively decompose one cluster at each iteration, while leaving the decomposed results of other clusters unchanged.

More formally, suppose that a cluster $\mathbf{c} = \begin{bmatrix} c_1 & \cdots & c_N \end{bmatrix}$ is selected for decomposition at the current iteration. Its core tensor $\boldsymbol{\mathcal{Z}}_{\mathbf{c}}$ and basis matrices $\mathbf{U}_{1,\mathbf{c}}, \ldots, \mathbf{U}_{N,\mathbf{c}}$ will be updated at this iteration, while those of other clusters and the cluster membership of each tensor element remain unchanged. To achieve this goal, we first compute the residual tensor of cluster $\mathbf{c}$, namely $\boldsymbol{\mathcal{R}}_{\mathbf{c}}$, by

$$\boldsymbol{\mathcal{R}}_{\mathbf{c}} = \boldsymbol{\mathcal{A}} \bigtimes_{n=1}^{N}{}_{n} \mathbf{M}_{n,\mathbf{c}}^T - \sum_{\substack{\mathbf{c}'\in\mathbf{C} \\ \mathbf{c}'\neq\mathbf{c}}} \Big( \boldsymbol{\mathcal{Z}}_{\mathbf{c}'} \bigtimes_{n=1}^{N}{}_{n} \mathbf{M}_{n,\mathbf{c}}^T \mathbf{U}_{n,\mathbf{c}'} \Big), \tag{12}$$

where the elements in the membership matrix $\mathbf{M}_{n,\mathbf{c}}$ are defined as

$$\forall i_1,\; \forall i_2,\; \big(\mathbf{M}_{n,\mathbf{c}}\big)_{i_1 i_2} = \begin{cases} 1, & \text{if } i_1 = (M_{n,\mathbf{c}})_{i_2}, \\ 0, & \text{otherwise,} \end{cases} \tag{13}$$

and $(M_{n,\mathbf{c}})_{i_2}$ denotes the $i_2$-th element of the set $M_{n,\mathbf{c}}$ that is defined as

$$M_{n,\mathbf{c}} = \Big\{ i \in \{1,\ldots,I_n\} \,\Big|\, c_n \in \big\{c_{n,i}^{(k)}\big\}_{k=1}^{K_n} \Big\}. \tag{14}$$

The elements of $M_{n,\mathbf{c}}$ are in fact the indices of mode-$n$ sub-tensors that are assigned to the mode-$n$ cluster subset $\mathbf{C}_{n,c_n}$. In other

[1] In brief, OMP selects a nonzero element by minimizing the error with respect to the *current* residual, while OLS instead minimizes the *total* residual error. Therefore, OMP is likely to find a worse locally optimal solution, since the values of previously selected nonzero elements are not allowed to change values during the selection process.

words, if the mode-$n$ cluster indices for the $i$-th mode-$n$ sub-tensor include $c_n$, $i$ should be a member of $M_{n,\mathbf{c}}$. Therefore, decomposing $\mathcal{R}_\mathbf{c}$ using $N$-SVD will obtain new $\mathcal{Z}_\mathbf{c}$ and $\{\mathbf{U}_{n,\mathbf{c}}\}_{n=1}^N$, while keeping the zero elements in the basis matrices of cluster $\mathbf{c}$ as zeros and updating their nonzero elements at the same time.

## 4. IMPLEMENTATION ISSUES

This section discusses the practical issues of MK-CTA, including performance bottlenecks and how to improve them in the clustering and decomposition stages (Section 4.1), the initial guess of the decomposed results of each cluster (Section 4.2), and suggestions on how to set the parameters of MK-CTA and their limitations (Section 4.3).

### 4.1 Performance Improvement Techniques

The offline computational costs of MK-CTA are higher than previous multilinear models. In our implementation, the input tensor is partitioned into blocks and stored on the hard disk as [Wang et al. 2005], while the decomposed results of each cluster are instead stored in the main memory. Therefore, two of the major performance bottlenecks of MK-CTA are the pseudoinverse computation in the clustering stage and frequent disk read/write operations for out-of-core computation in both clustering and decomposition stages. In the following two subsections, we will discuss how to accelerate the pseudoinverse computation and briefly describe the proposed block partition and disk access schemes.

4.1.1 *Pseudoinverse Computation.* To solve Eq. (4) in the clustering stage, we should prevent computing the pseudoinverse of $\mathbf{Z}_{n,i}^{(K_n)}$, since its size is frequently very large. Fortunately, by employing the identity that $\big(\mathbf{Z}_{n,i}^{(K_n)}\big)^\dagger = \big(\big(\mathbf{Z}_{n,i}^{(K_n)}\big)^T\mathbf{Z}_{n,i}^{(K_n)}\big)^\dagger\big(\mathbf{Z}_{n,i}^{(K_n)}\big)^T$, the objective function in Eq. (4) can be reformulated as

$$uf_n\big(\mathcal{A}_{\langle n,i\rangle}\big)\mathbf{Z}_{n,i}^{(K_n)}\big(\big(\mathbf{Z}_{n,i}^{(K_n)}\big)^T\mathbf{Z}_{n,i}^{(K_n)}\big)^\dagger\big(\mathbf{Z}_{n,i}^{(K_n)}\big)^T uf_n\big(\mathcal{A}_{\langle n,i\rangle}\big)^T. \tag{15}$$

In this way, we only need to compute the pseudoinverse of a much smaller matrix $\big(\mathbf{Z}_{n,i}^{(K_n)}\big)^T\mathbf{Z}_{n,i}^{(K_n)}$ and can further reuse the result of $uf_n\big(\mathcal{A}_{\langle n,i\rangle}\big)\mathbf{Z}_{n,i}^{(K_n)}$.

For Eq. (15), we have to compute $\forall i, \forall k, uf_n\big(\mathcal{A}_{\langle n,i\rangle}\big)\hat{\mathbf{Z}}_n^{(k)}$ and $\forall k, \forall k', \big(\hat{\mathbf{Z}}_n^{(k)}\big)^T\hat{\mathbf{Z}}_n^{(k')}$, which in fact can be performed in the reduced tensor space. This means that $\hat{\mathbf{Z}}_n^{(1)}, \ldots, \hat{\mathbf{Z}}_n^{(C_n)}$ are not actually derived at all, since computing them requires to respectively reconstruct the contribution of each cluster and often results in a lot of large sparse matrices that may need to be stored out of core. We instead apply the identities that

$$uf_n\big(\mathcal{A}_{\langle n,i\rangle}\big)\,uf_n\big(\hat{\mathcal{Z}}_\mathbf{c}\big)^T = uf_n\Big(\mathcal{A}_{\langle n,i\rangle}\bigtimes_{\substack{n'=1\\n'\neq n}}^{N}\mathbf{U}_{n',\mathbf{c}}^T\Big)\,uf_n\big(\mathcal{Z}_\mathbf{c}\big)^T, \tag{16}$$

$$uf_n\big(\hat{\mathcal{Z}}_\mathbf{c}\big)\,uf_n\big(\hat{\mathcal{Z}}_{\mathbf{c}'}\big)^T = uf_n\Big(\mathcal{Z}_\mathbf{c}\bigtimes_{\substack{n'=1\\n'\neq n}}^{N}\mathbf{U}_{n',\mathbf{c}'}^T\mathbf{U}_{n',\mathbf{c}}\Big)\,uf_n\big(\mathcal{Z}_{\mathbf{c}'}\big)^T, \tag{17}$$

which can be easily verified by the relation between the mode-$n$ product and the Kronecker product [De Lathauwer et al. 2000]. Eqs. (16) and (17) particularly allow the computation of Eq. (15) to be executed in core as much as possible. Note that if $K_{n'} = 1$ for all $n' \neq n$, Eq. (17) can be directly evaluated as a zero matrix when $\mathbf{c} \neq \mathbf{c}'$, namely $\forall \mathbf{c} \neq \mathbf{c}'$, $uf_n\big(\hat{\mathcal{Z}}_\mathbf{c}\big)\,uf_n\big(\hat{\mathcal{Z}}_{\mathbf{c}'}\big)^T = \mathbf{0}$. This will lead to a more efficient process for MK-CTA if *hard* clustering is applied along all or most modes of $\mathcal{A}$.

For greedy search, the objective function in Eq. (11) can be similarly reformulated as Eq. (15). Thus, the results of $uf_n\big(\mathcal{A}_{\langle n,i\rangle}\big)\hat{\mathbf{Z}}_n^{(k)}$ and $\big(\hat{\mathbf{Z}}_n^{(k)}\big)^T\hat{\mathbf{Z}}_n^{(k')}$ also can be reused and only need to be computed once throughout the clustering process for the $n$-th mode.

4.1.2 *Block Partition and Disk Access Schemes.* Another most critical performance bottleneck of MK-CTA is frequent disk read/write operations. In order to maximize disk access efficiency as much as possible, tasks should be carefully scheduled. Recall that in our implementation, $\mathcal{A}$ is partitioned into blocks that are stored on the disk. It is highly likely that elements of a cluster reside in multiple blocks, while at the same time a single block contains elements of more than one cluster. Evaluating Eq. (16) for different clusters thus may require to repeatedly read the same block from the disk. In the decomposition stage, frequently reading/writing the same block is also unavoidable, since it is difficult to divide the decomposition process of a cluster into subprocesses that can be separately computed and then combined.

To reduce the number of disk access operations, we initially partition the input tensor into smaller blocks with an appropriate size before executing MK-CTA, and employ the most-recently-used replacement strategy in both clustering and decomposition stages to cache previously accessed blocks in the main memory. This allows a parallel or multithreaded process that can simultaneously compute the results of multiple clusters. To exploit parallel processing, note that only non-overlapped clusters can be decomposed at the same time in the decomposition stage. In our implementation, the block size is determined by

$$\forall n, B_n = \lceil \alpha I_n / C_n \rceil, \tag{18}$$

where $B_n$ denotes the mode-$n$ rank of each block, and $\alpha$ is a user-defined constant. We have found that this scheme tends to reduce the total number of disk access operations for each block. The constant $\alpha$ was set to 3 in our experiments, but other values in the interval $[2, 4]$ would also be good choices.

### 4.2 Initial Guess

In the clustering stage of MK-CTA, we need the core tensor and basis matrices of each cluster to iteratively update the basis matrices of clustered modes. Similar to many alternating least-squares algorithms, MK-CTA may suffer from a bad initial guess of the decomposed results of each cluster. We thus first obtain an initial guess by performing MK-CTA, with $K_n = 1$ for all $n$, for several iterations. As discussed in Section 4.1.1, this will quickly derive an initial solution. After that, we can further refine the core tensor and basis matrices of each cluster by setting $K_1, \ldots, K_N$ to their original values. In this way, the remaining problem is how to generate initial clustering seeds for clustered modes. In our experience, common initial clustering techniques, as those proposed in [Tsai and Shih 2012, Section 5.1], often work very well in practice.

### 4.3 Parameter Suggestions

Many parameters of MK-CTA need to be manually and empirically tuned for specific purposes. For $C_1, \ldots, C_N$, there is no limitation on their values. $C_n$ generally depends on the input data and needs to be fine-tuned. A large value mainly increases approximation quality, memory requirements, and offline computational costs, but has a small impact on rendering times.

For $K_1, \ldots, K_N$, there is also no limitation on their values. Typically, setting $K_n$ to 2 or 3 is adequate to generate high-quality images in practical real-time applications. A value larger than 3 usually leads to negligible quality improvement and memory overhead, but will substantially increase rendering times and offline computational costs.

For $R_1, \ldots, R_N$, their values are constrained to be the same for each cluster, but still can be different for each mode. Otherwise, MK-CTA cannot refine the basis matrices of each cluster in the clustering stage. It is suggested to set $R_n$ in the range of $[1, 4]$ for interactive or real-time rendering rates. A large value may slightly raise offline computational costs, but will considerably increase rendering times, approximation quality, and memory requirements.

Note that the parameters of $N$-SVD, particularly the reduced rank of each mode, can be adopted as a reference guide for the parameters of MK-CTA. It has been found that the approximation errors of $N$-SVD and MK-CTA tend to be close to each other, when $C_n \cdot R_n$ is equal to the mode-$n$ reduced rank for $N$-SVD. We thus first perform $N$-SVD on the input tensor and fine-tune the parameters of $N$-SVD until the desired approximation error is reached. For all $n$, we then select values of $K_n$ and $R_n$ from the suggested ranges and finally determine $C_n$ from the reduced ranks for $N$-SVD. If the decomposed result is not satisfying, $K_1, \ldots, K_N$ and $R_1, \ldots, R_N$ have to be further fine-tuned according to the requirements of an application.

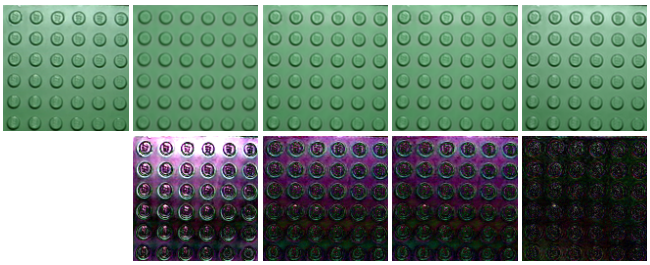
In summary, it is suggested to first set $R_1, \ldots, R_N$ in the range of $[1, 4]$ and $K_1, \ldots, K_N$ to 2 or 3 for high-performance rendering, since these parameters are the most important key to efficient runtime reconstruction. Then, determine $C_1, \ldots, C_N$ to reach a desired compression ratio or approximation quality for a good tradeoff. If the offline approximation time is really a concern, set $C_1, \ldots, C_N$ to small values instead and increase $R_1, \ldots, R_N$ for similar approximation quality, at the cost of a lower rendering rate.

## 5. EXPERIMENTAL RESULTS

In this section, we present the experimental results of MK-CTA on three common types of large-scale multidimensional visual datasets in computer graphics and visualization, including BTFs, TVLFs, and TVVD. We also compare MK-CTA to previous tensor-based methods: $N$-SVD and K-CTA. In the experiments of MK-CTA, we adopted optimal search to sparsely cluster the $n$-th mode of an input tensor when $K_n$ was less than 3. Otherwise, greedy search was instead employed for that mode. The raw and approximated data were respectively stored as 32-bit and 16-bit floating point numbers. Experiments and simulation timings were conducted and measured on a workstation with an Intel i7-3930K CPU, an NVIDIA GeForce GTX TITAN graphics card, and 16 GB main memory. The approximation quality is evaluated by the *signal-to-noise ratio* (SNR).

### 5.1 Bidirectional Texture Functions

*Overview.* BTFs [Dana et al. 1999] can be regarded as a generalization of BRDFs to faithfully capture spatial variations of appearance and reflectance on real-world object surfaces. Over the last decade, the enormous amount of BTF data has stimulated the development of modern approximation algorithms [Filip and Haindl 2009; Tsai et al. 2011], among which multilinear models [Tsai and Shih 2012; Vasilescu and Terzopoulos 2004; Wang et al. 2005; Wu et al. 2008] have been proved as one category of the most efficient analysis methods for BTFs.



(a) Raw data (1.98 GB) (b) $N$-SVD (3.8 MB) (c) $N$-SVD (12.06 MB) (d) K-CTA (12.4 MB) (e) MK-CTA (12.15 MB)

Fig. 4. Reconstructed images of approximated BTFs ("Lego" ©Volumetric Surface Texture Database) based on different multilinear models. The top row shows reconstructed images; the bottom row shows absolute difference images (scaled by a factor of 6).

*Experiment settings.* In our experiments, a BTF[2] is organized as a fourth order tensor[3] and approximated using $N$-SVD, K-CTA, and MK-CTA. The tensor is sparsely clustered along only the view mode for K-CTA, but along both illumination and view modes for MK-CTA.

*Rendering process.* Before rendering, we follow the suggestion in [Tsai and Shih 2012] to reconstruct the $x$ and $y$ modes of each cluster core tensor and organize the results into a core texture, so that hardware texture filtering on GPUs for the two modes can be exploited for efficient rendering. The nonzero elements of mode-$\omega_l$ and mode-$\omega_v$ basis matrices are packed into a basis texture, and another index texture is employed to store the cluster indices of each sampled illumination/view direction.

At runtime, the approximated BTFs based on the three multilinear models are reconstructed on GPUs for high-performance rendering. The BTF rendering process of MK-CTA is mostly the same as that of K-CTA. For a BTF texel, we simply access the index and basis textures in the pixel shader for corresponding elements in basis matrices according to current illumination and view directions, obtain associated cluster core tensors from the core texture, and then reconstruct the illumination and view modes of the BTF texel to determine the final shading color.

*Results.* Table I shows the statistics of BTF approximation for different multilinear models. From this table, MK-CTA outperforms $N$-SVD and K-CTA, especially for the BTF "Lego". With similar compression ratios, MK-CTA is superior to $N$-SVD in terms of approximation errors and runtime performance. At similar rendering rates, MK-CTA also can achieve higher approximation quality than $N$-SVD and K-CTA. Since there are large data variations among both illumination and view modes of the BTF "Lego", $N$-SVD cannot fully exploit the coherence among the two modes with small reduced ranks. Although K-CTA sparsely clusters the view mode to allow an accurate and efficient reconstruction for that mode, data variations along the illumination mode still cannot be effectively approximated for high-performance rendering.

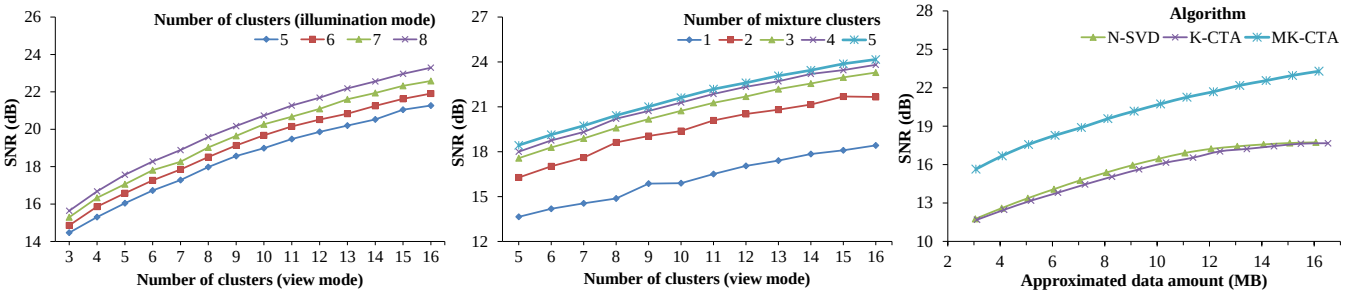
Figure 4 compares the reconstructed BTF images based on the three tensor-based methods. Readers may refer to supplemental

[2]The BTF datasets were provided in courtesy of Dr. Xin Tong and collected from the *Volumetric Surface Texture Database* [Koudelka et al. 2003], http://vision.ucsd.edu/kriegman-grp/research/vst/.

[3]The four modes include the illumination direction $\omega_l$, the view direction $\omega_v$, and the 2D spatial coordinates $(x, y)$ of a texel.

Table I. Statistics of BTF approximation. For the same BTF, we compare different multilinear models at similar rendering rates and/or with similar compression ratios. For $N$-SVD, two different parameter configurations are shown: one at a similar rendering rate to MK-CTA; the other with a similar compression ratio. As for K-CTA, only one parameter configuration is listed, since it satisfies both comparison conditions. The rendering rates were measured under 4 directional lights with a screen resolution of 1024×768.

| BTF | Hole | | | | Lego | | | |
|---|---|---|---|---|---|---|---|---|
| Object model | Bunny | | | | Teapot | | | |
| $I_{\omega_l} \times I_{\omega_v} \times I_x \times I_y$ | 81×81×128×128 | | | | 120×90×128×128 | | | |
| Raw data (GB) | 1.2 | | | | 1.98 | | | |
| Algorithm | $N$-SVD | $N$-SVD | K-CTA | MK-CTA | $N$-SVD | $N$-SVD | K-CTA | MK-CTA |
| $R_{\omega_l} \times R_{\omega_v} \times R_x \times R_y$ | 24×24×80×80 | 32×40×80×80 | 32×4×80×80 | 4×4×80×80 | 20×24×64×64 | 32×48×64×64 | 32×4×64×64 | 4×4×64×64 |
| $C_{\omega_l} \times C_{\omega_v} \times C_x \times C_y$ | 1×1×1×1 | 1×1×1×1 | 1×10×1×1 | 8×10×1×1 | 1×1×1×1 | 1×1×1×1 | 1×12×1×1 | 8×12×1×1 |
| $K_{\omega_l} \times K_{\omega_v} \times K_x \times K_y$ | 1×1×1×1 | 1×1×1×1 | 1×3×1×1 | 3×3×1×1 | 1×1×1×1 | 1×1×1×1 | 1×3×1×1 | 3×3×1×1 |
| Approximated data (MB) | 7.09 | 15.69 | 16.03 | 15.73 | 3.8 | 12.06 | 12.4 | 12.15 |
| SNR (dB) | 19.39 | 24.33 | 22.82 | 24.34 | 13.36 | 17.23 | 17.03 | 21.69 |
| Approximation time (min.) | 3.25 | 3.32 | 32.92 | 50.95 | 26.92 | 33.17 | 115.23 | 142.95 |
| Rendering rate (FPS) | 65.17 | 28.62 | 67.39 | 63.64 | 142.34 | 27.14 | 143.56 | 139.49 |




(a) $K_{\omega_l} = K_{\omega_v} = 3$

(b) $C_{\omega_l} = 8$, $K_{\omega_l} = K_{\omega_v}$ ($K_{\omega_l}, K_{\omega_v} \in \{1, \ldots, 5\}$)

(c) $N$-SVD ($R_{\omega_l} = 32$), K-CTA ($R_{\omega_l} = 32$, $K_{\omega_v} = 3$), MK-CTA ($C_{\omega_l} = 8$, $K_{\omega_l} = K_{\omega_v} = 3$)

Fig. 5. Reconstruction errors of approximated BTFs ("Lego" ©Volumetric Surface Texture Database) based on different multilinear models. Major parameters are shown under each subfigure. Readers may refer to Table I for other parameter settings. Subfigures (a), (b): different configurations of MK-CTA; subfigure (c): comparison of the SNR versus the data amount among different configurations of $N$-SVD, K-CTA, and MK-CTA.

materials for high-resolution images and more experimental results. The visual quality of reconstructed images for MK-CTA is higher than that for $N$-SVD and K-CTA. Among the three algorithms, only MK-CTA can faithfully capture the specular reflection in the top of the raw image. Figure 5(a) further plots the SNR of MK-CTA versus the number of clusters for the view mode, with which the SNR almost increases linearly. In Figure 5(b), we show the effects of changing the number of clusters for the view mode and/or the numbers of mixture clusters for both illumination and view modes. Note that the SNR quickly becomes saturated when the numbers of mixture clusters increase, making 2 or 3 a favorable choice. Moreover, Figure 5(c) compares the SNR versus the data amount among different configurations of the three multilinear models. It shows that MK-CTA can achieve lower approximation errors than $N$-SVD and K-CTA with similar compression ratios.

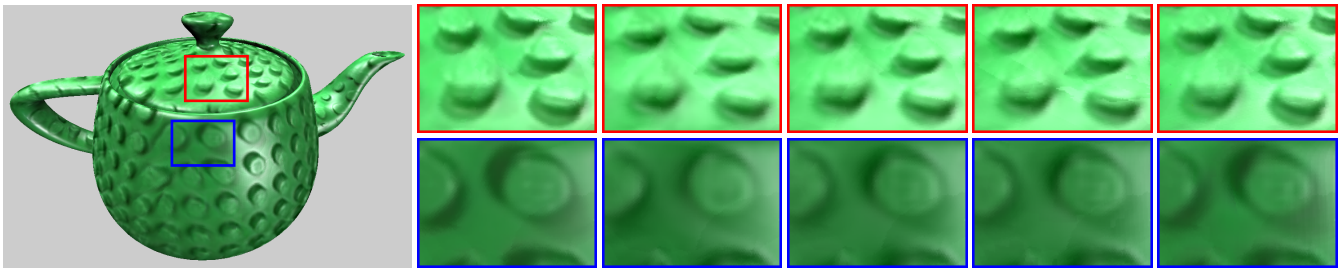
In Figure 6, we demonstrate the rendered images of the three tensor algorithms. Under 4 directional light sources, both K-CTA and MK-CTA can achieve rendering rates that are 3∼5 times faster than 30 *frames per second* (FPS), while the runtime performance of $N$-SVD is only near real-time with a similar compression ratio. As Figure 6(c) shows, we change the parameters of $N$-SVD to achieve a similar rendering rate, but at the cost of a much lower SNR. Furthermore, the rendered image of MK-CTA exhibits the highest visual quality among the three methods. The high-frequency visual features in the BTF, particularly at regions with sharp shadows and specular highlights, can be accurately preserved by MK-CTA.

*Environment lighting.* In addition to common light sources, approximated BTFs based on MK-CTA also can support complex *high-dynamic-range* (HDR) environment lighting[4] (Figure 7). Specifically, we precompute the convolution between an environment map and the mode-$\omega_l$ basis matrices of an approximated BTF. To allow interactively rotating the lighting environment, the convolution results for different orientations of the environment map have to be precomputed. We parameterize the orientation using the $yxz$ (yaw-pitch-roll) Euler angles and store the final precomputed results as a 6D table[5] (packed into a 3D texture for rendering on GPUs). In the experiments, we sample 32∼80 different values for each Euler angle, which usually consumes 100∼250 MB GPU memory for each pair of an approximated BTF and an environment map.

*Conclusion.* For BTF approximation, both K-CTA and MK-CTA are more favorable than $N$-SVD at similar rendering rates and/or with similar compression ratios. Note that K-CTA and MK-CTA are comparable under the two comparison conditions, but frequently K-CTA achieves slightly higher rendering rates and MK-CTA holds lower approximation errors in terms of the SNR.

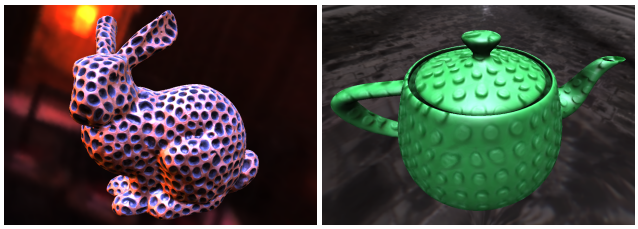
[4]The HDR environment maps were collected from the Light Probe Image Gallery [Debevec 1998], http://www.pauldebevec.com/Probes/.

[5]The 6 dimensions are respectively the color channel, the reduced rank index, the cluster index, and the Euler angles.



(a) Rendered image of MK-CTA (b) Raw data (1.98 GB, <0.01 FPS) (c) $N$-SVD (3.8 MB, 13.36 dB, 142.34 FPS) (d) $N$-SVD (12.06 MB, 17.23 dB, 27.14 FPS) (e) K-CTA (12.4 MB, 17.03 dB, 143.56 FPS) (f) MK-CTA (12.15 MB, 21.69 dB, 139.49 FPS)

Fig. 6. Rendered images of approximated BTFs ("Lego" ©Volumetric Surface Texture Database) based on different multilinear models at similar rendering rates (subfigures (c), (e), (f)) and/or with similar compression ratios (subfigures (d), (e), (f)). Readers may refer to supplemental materials for full-size images.



(a) Bunny with "Hole" (98.64 FPS) (b) Teapot with "Lego" (236.62 FPS)

Fig. 7. Rendered images of approximated BTFs based on MK-CTA with HDR environment lighting. (BTF "Hole" ©Dr. Xin Tong, BTF "Lego" ©Volumetric Surface Texture Database, model "Bunny" ©Stanford 3D Scanning Repository, HDR maps "Grace Cathedral" and "Uffizi Gallery" ©Light Probe Image Gallery)

## 5.2 Time-Varying Light Fields

*Overview.* Light fields [Levoy 2006; Levoy and Hanrahan 1996] have been widely used in image-based methods for rendering 3D images of a real-world static scene from arbitrary views. They provide a discrete representation for radiance along rays that enter/leave a free space at arbitrary positions and in different directions. To extend light fields for dynamic scenes, a TVLF [Bando et al. 2013] is often described as a 5D function of animated 2D image slices. A comprehensive survey on light fields is beyond the scope of this article. More information can be found in [Levoy 2006; Shum et al. 2007].

*Experiment settings.* In our experiments, a TVLF[6] is organized as a fourth order tensor[7] and approximated using $N$-SVD, K-CTA, and MK-CTA. We do not decompose the time mode for the three methods, since the temporal coherence in the adopted TVLFs is very low. Nevertheless, one can always choose to decompose the time mode when temporal coherence is high. For K-CTA, the tensor is sparsely clustered along the $x$ mode, while along the $x$, $y$, and slice modes for MK-CTA. To reduce offline computational costs, we apply MK-CTA (or K-CTA) to the spatially downsampled TVLF, upsample the decomposed basis matrices, and further refine the upsampled results by performing MK-CTA (or K-CTA) again on the input TVLF with the original spatial resolution.

Table II. Statistics of TVLF approximation. We compare different multilinear models with similar compression ratios. The rendering rates were measured with a screen resolution of 840×525.

| TVLF | AnimatedBunnies | | |
|---|---|---|---|
| $I_x \times I_y \times I_s \times I_t$ | 840×525×27×89 | | |
| Raw data (GB) | 11.84 | | |
| Algorithm | $N$-SVD | K-CTA | MK-CTA |
| $R_x \times R_y \times R_s \times R_t$ | 264×166×17×89 | 4×169×17×89 | 4×4×2×89 |
| $C_x \times C_y \times C_s \times C_t$ | 1×1×1×1 | 63×1×1×1 | 60×40×9×1 |
| $K_x \times K_y \times K_s \times K_t$ | 1×1×1×1 | 2×1×1×1 | 2×2×2×1 |
| Approximated data (MB) | 380.0 | 379.88 | 380.3 |
| SNR (dB) | 29.03 | 28.9 | 28.95 |
| Approximation time (min.) | 65.03 | 344.15 | 758.93 |
| Rendering rate (FPS) | 0.05 | 1.26 | 56.24 |

*Rendering process.* Similar to the BTF rendering process of MK-CTA, the core tensor of each cluster is packed into a core texture. If the time mode of a TVLF is decomposed, we first reconstruct that mode before packing for efficient temporal interpolation at runtime. The nonzero elements in mode-$x$, mode-$y$, and mode-$s$ basis matrices and the cluster indices of each sampled $x$, $y$, or $s$ coordinate are also stored in basis and index textures. At runtime, we first linearly interpolate the time mode of the core texture on GPUs to create a new core texture for a novel time from 5 nearby sampled frames. Then, the image of a novel view is obtained by linearly interpolating the reconstructed results of 4 nearest sampled views in the pixel shader, where the reconstruction for a sampled view is performed based on the basis, index, and time-interpolated core textures.

*Results.* Table II lists the statistics of TVLF approximation based on different tensor representations. Figure 1 also shows the rendered images of the three algorithms. With similar compression ratios, MK-CTA holds the highest rendering rate. Note that the rendering performance of $N$-SVD and K-CTA is even slower than that of raw data (1.79 FPS), which means that the two methods are inadequate to real-time high-quality TVLF rendering. Unlike BTF

[6]The TVLF datasets were collected from the *Synthetic Light Field Archive* [Wetzstein et al. 2011; Wetzstein et al. 2012], http://web.media.mit.edu/~gordonw/SyntheticLightFields/.

[7]The four modes respectively correspond to the 2D spatial coordinates $(x, y)$ of an image pixel, the slice index $s$, and the time (or frame index) $t$. Note that each slice is the resulting image from an observer position that is often parameterized by a 2D point on a plane for light fields. Nevertheless, we collapse the two modes into only one, namely the slice mode, since the number of observer positions is quite small for the utilized TVLFs.

approximation, the SNR of MK-CTA is similar to $N$-SVD and K-CTA, since it becomes more difficult to find a good (locally) optimal solution when the numbers of clusters are large. Nevertheless, the visual quality of MK-CTA is still the best by manually inspecting the rendered images in Figure 1. The result of $N$-SVD exhibits strong ringing effects around object boundaries, which are less perceptible in the result of MK-CTA. Since there are large variations along the slice mode of the TVLF, $N$-SVD fails to efficiently approximate image slices from different observer positions. This will lead to the ringing effects, where the reconstructed slice from an observer position implicitly shows the *ghosts* of other slices. By contrast, MK-CTA can sparsely cluster the slice mode to improve intra-cluster coherence and significantly reduce the ringing effects. As for K-CTA, there are some visible vertical seams in its result, since the $x$ mode of the TVLF is clustered but not accurately approximated. Although increasing the number of mixture clusters can alleviate the artifacts, it will moderately increase approximation and rendering times. Spatial filtering is another common technique to solve this issue, but also with an inevitable performance hit at runtime. Note that MK-CTA sometimes smoothes out sharp visual features in a few regions, while $N$-SVD and K-CTA can more accurately preserve them. For example, refer to the first row in Figure 1 for the highlights on the hip of the cyan bunny. Nevertheless, the strong ringing effects and visible seams are still more perceptible and annoying.

*Conclusion.* For TVLF approximation, MK-CTA can achieve significantly higher rendering rates than $N$-SVD and K-CTA with similar compression ratios. Nevertheless, the offline computational costs of MK-CTA are also the highest among the three algorithms.

## 5.3 Time-Varying Volume Data

*Overview.* TVVD [Wang et al. 2010] play an important role in studying the dynamic aspects of a scientific phenomenon. Visualizing TVVD would help scientists to effectively explore, analyze, and understand the underlying physical process behind the phenomenon. Since the amount of TVVD has drastically increased over the last decades, TVVD need to be compressed for efficient visualization at interactive rates. Traditional methods usually rely on vector quantization [Schneider and Westermann 2003] and predefined basis decomposition, such as wavelets [Guthe and Straßer 2001] and the discrete cosine transform [Lum et al. 2002], to efficiently compress and decompress TVVD. Recently, tensor-based algorithms have been applied to approximate large-scale volume data [Suter et al. 2011; Suter et al. 2013; Wang et al. 2005; Wu et al. 2008] and reported a lot of positive results and feedback due to their data dependent nature and flexibility. Moreover, the decomposed data based on multilinear models can be coherently stored without repacking to improve cache performance. Sparse representation [Gobbetti et al. 2012] is also another appropriate choice of volume data approximation. Nevertheless, MK-CTA additionally allows voxel block/cluster sizes to vary and integrates tensor approximation to exploit the coherence within a voxel block/cluster. Interested readers may find more information on volume data approximation in [Balsa Rodríguez et al. 2014; Wang et al. 2010].

*Experiment settings.* In this article, a TVVD[8] set is organized as a fourth order tensor[9] and approximated using $N$-SVD, K-CTA, and MK-CTA. For K-CTA and MK-CTA, we first employ $N$-SVD to extract a single mode-$t$ basis matrix and then perform K-CTA and MK-CTA on the decomposed core tensor to sparsely cluster the specified modes. This can significantly reduce offline computational costs, but only has a negligible impact on approximation errors. The spatial downsampling process, as described in Section 5.2, is also applied to further reduce offline costs. Note that one can always choose to sparsely cluster the time mode for lower approximation errors when using MK-CTA. Nevertheless, since both offline and runtime performance will also be substantially decreased and only a portion of the approximated TVVD are required to render a single frame, we do not see the need to sparsely cluster the time mode.

*Rendering process.* Like TVLFs, we first reconstruct the time mode of each cluster core tensor, pack the results into a core texture, and then create the basis and index textures for MK-CTA before rendering. At runtime, we adopt a GPU-based volume ray caster with pre-integration [Engel et al. 2001], early ray termination, and empty-space skipping[10] [Krüger and Westermann 2003]. For a non-empty sampled position, the corresponding data value is reconstructed from the basis, index, and time-interpolated core textures in the pixel shader.

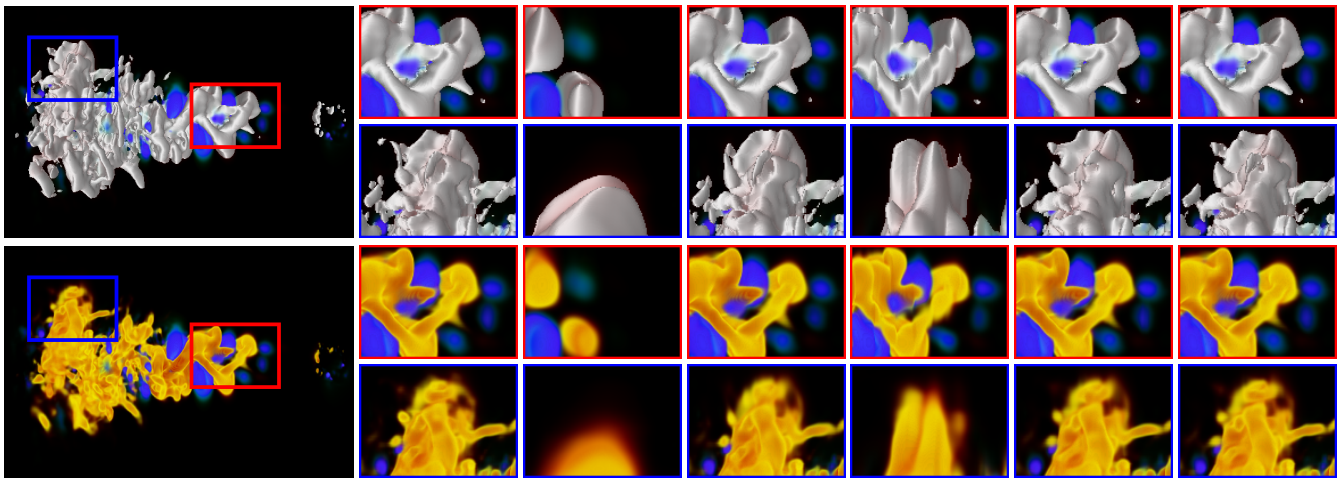
[8]The TVVD were provided in courtesy of Prof. Kwan-Liu Ma, from the *Time-Varying Volume Data Repository*, http://www.cs.ucdavis.edu/~ma/ITR/tvdr.html.

[9]The four modes are respectively the 3D spatial coordinates $(x, y, z)$ of a voxel and the time (or frame index) $t$.

[10]We first reconstruct the TVVD from the approximated data. The empty space is then determined from the reconstructed TVVD based on the adopted transfer function, and a separate data structure is precomputed for each time frame. When combined with pre-integration, we employ overlapped blocks, where there is an overlap of (at least) two voxels at each block boundary, to precompute the empty space.

Table III. Statistics of TVVD approximation. For the same TVVD, we compare different multilinear models at similar rendering rates or with similar compression ratios. The rendering rates were measured with a screen resolution of 640×480.

| TVVD | TurbJet | | | | | TurbVortex | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| $I_x \times I_y \times I_z \times I_t$ | 258×258×208×150 | | | | | 256×256×256×98 | | | | |
| Raw data (GB) | 7.74 | | | | | 6.13 | | | | |
| Algorithm | $N$-SVD | $N$-SVD | K-CTA | K-CTA | MK-CTA | $N$-SVD | $N$-SVD | K-CTA | K-CTA | MK-CTA |
| $R_x \times R_y \times R_z \times R_t$ | 4×4×6×96 | 38×38×68×96 | 6×6×2×96 | 38×38×2×96 | 2×2×2×96 | 6×6×6×64 | 64×64×64×64 | 6×2×6×64 | 64×2×64×64 | 2×2×2×64 |
| $C_x \times C_y \times C_z \times C_t$ | 1×1×1×1 | 1×1×1×1 | 1×1×32×1 | 1×1×32×1 | 18×18×32×1 | 1×1×1×1 | 1×1×1×1 | 1×30×1×1 | 1×30×1×1 | 30×30×30×1 |
| $K_x \times K_y \times K_z \times K_t$ | 1×1×1×1 | 1×1×1×1 | 1×1×2×1 | 1×1×2×1 | 2×2×2×1 | 1×1×1×1 | 1×1×1×1 | 1×2×1×1 | 1×2×1×1 | 2×2×2×1 |
| Approximated data (MB) | 0.05 | 18.07 | 0.64 | 18.15 | 18.0 | 0.05 | 32.1 | 0.45 | 31.89 | 31.65 |
| SNR (dB) | 2.72 | 19.42 | 8.18 | 18.41 | 19.33 | 7.94 | 40.59 | 10.77 | 40.01 | 40.49 |
| Approximation time (min.) | 5.12 | 30.5 | 116.78 | 200.07 | 329.77 | 2.52 | 16.47 | 139.85 | 205.73 | 413.55 |
| Rendering rate (FPS) | 96.05 | 0.28 | 94.98 | 4.16 | 93.26 | 31.05 | 0.02 | 29.79 | 0.39 | 30.87 |



(a) Rendered images of MK-CTA | (b) Raw data (7.74 GB, 3.79 FPS) | (c) $N$-SVD (0.05 MB, 2.72 dB, 96.05 FPS) | (d) $N$-SVD (18.07 MB, 19.42 dB, 0.28 FPS) | (e) K-CTA (0.64 MB, 8.18 dB, 94.98 FPS) | (f) K-CTA (18.15 MB, 18.41 dB, 4.16 FPS) | (g) MK-CTA (18.0 MB, 19.33 dB, 93.26 FPS)

Fig. 8. Rendered images of approximated TVVD ("TurbJet" ©Time-Varying Volume Data Repository) based on different multilinear models at similar rendering rates (subfigures (c), (e), (g)) or with similar compression ratios (subfigures (d), (f), (g)). The top and bottom rows respectively show rendered images with and without shaded isosurfaces. Readers may refer to supplemental materials for full-size images and our accompanying video for animations of TVVD.

*Results.* Table III shows the statistics of TVVD approximation for $N$-SVD, K-CTA, and MK-CTA. Figure 8 further compares the rendered images of the three algorithms at similar rendering rates or with similar compression ratios. Besides using transfer functions, we additionally extract isosurfaces with diffuse lighting in some rendered images. At similar rendering rates, both $N$-SVD and K-CTA fail to capture the important characteristics of a complex TVVD set, while MK-CTA provides the lowest approximation error and the best visual quality for smoothly navigating the TVVD through space and time. Moreover, the extracted isosurfaces of MK-CTA are accurate and smooth without sacrificing sharp visual features, further demonstrating its potentials for TVVD approximation. With similar compression ratios, the visual quality of the three algorithms is almost indistinguishable, but only MK-CTA can achieve real-time rendering performance, while it takes a few or more seconds for $N$-SVD and K-CTA to render a single frame. Although the SNR of $N$-SVD is the highest among the three methods, the difference over MK-CTA is small and negligible.

*Conclusion.* For TVVD approximation, $N$-SVD and K-CTA are inadequate to interactive visualization applications. Only MK-CTA can provide both real-time rendering rates and high visual quality, at the cost of longer approximation times.

## 5.4 Discussions and Limitations

MK-CTA generally allows a better tradeoff among visual quality, compression ratios, and rendering rates than previous tensor-based methods. With similar compression ratios, MK-CTA can outperform $N$-SVD and K-CTA in other two aspects, especially for the experimental results of TVLFs. At similar rendering rates, MK-CTA also can provide higher visual quality with an appropriate compression ratio. When there are large data variations among more than one dimension of an input dataset, $N$-SVD requires large reduced ranks for these dimensions to obtain high-quality results, which will significantly reduce runtime rendering performance. K-CTA alleviates this issue by clustering one of these dimensions, but the reduced ranks of other dimensions may be still too large to reach real-time rendering rates. By contrast, MK-CTA can cluster all dimensions and employ large numbers of clusters for high-quality results, rather than large reduced ranks. This particularly leads to efficient reconstruction of elements in the dataset and high-performance photorealistic rendering.

Although the parameters of MK-CTA need to be fine-tuned for each dataset, the suggested tuning guidelines and limitations of important parameters are described in Section 4.3. With similar compression ratios, we have found that the approximation errors of different parameter configurations of MK-CTA are close to each other in practice. By fixing the numbers of mixture clusters, changing the numbers of clusters and reduced ranks to reach similar compression ratios only has a small impact on approximation errors, but still significantly influences offline computational costs and rendering rates. This shows that the derived solutions are robust to parameter configurations. Users can freely change parameters for a specific purpose, without worrying about obtaining a poor solution.

The offline computational costs of MK-CTA are higher than previous multilinear models, but more efficient runtime performance can be achieved at the same time. As discussed in [Fout and Ma 2007; Balsa Rodríguez et al. 2014], this asymmetric rendering process is usually desired. To reduce the offline costs into a reasonable range, we also propose several performance improvement techniques. For in-core datasets, the decomposition stage is often the most critical performance bottleneck. When optimal search is employed and the number of mixture clusters is larger than 2, the clustering stage immediately becomes another major bottleneck. As for out-of-core datasets, disk access times usually dominate the offline performance. We believe that a GPU-accelerated implementation and a high-performance parallel disk system could further decrease the offline computational costs. Nevertheless, the scalability issue of MK-CTA is still left as our future work.

Compared to wavelet-based methods, MK-CTA can produce overlapped blocks/clusters with varying sizes and members. This allows the decomposed results of MK-CTA to be more data-dependent and compact, since intra-cluster coherence can be improved via clustering. Although the decomposed results of MK-CTA may lead to less spatial locality of memory accesses in the rendering process, we have found that nearby elements tend to be assigned into the same cluster if they are highly coherent. In the opposite case, wavelet-based methods still group nearby elements and often fail to efficiently approximate them. The benefit of better spatial locality instead may be overwhelmed by the increase in memory storage. Nevertheless, it needs a thorough analysis and comparison of wavelet- and tensor-based methods to reach a conclusion. Additionally, wavelet-based methods can capture multiscale visual features among data, while MK-CTA intrinsically does not support this characteristic. Integrating hierarchical models with MK-CTA is left as a future research direction.

For a multidimensional dataset, the data distributions along some dimensions may be different from those along other dimensions, but we treat all dimensions equally in MK-CTA to simplify the proposed algorithm. Since the employed multiway clustering method implicitly assumes the same type of distribution along each dimension, inappropriate cluster membership may be obtained when the assumption is violated. It is possible to extend MK-CTA to overcome this drawback, which is left as future work. Nevertheless, our experimental results show that MK-CTA works well in practice even if treating all dimensions equally. We believe that this is due to the merit of sparse representation, since inappropriate cluster membership can be moderately compensated by mixing the decomposed results of different clusters.

Note that MK-CTA does not always ensure a globally optimal solution to Eq. (3). Even a local optimum is guaranteed only when optimal search is applied in the clustering stage. According to our experience, MK-CTA often converges to a local optimum within 6 iterations. We have also observed that the increases in the numbers of clusters or mixture clusters have a small impact on the convergence rate, since the proposed method for the initial guess works very well in practice. Nevertheless, the increases in clustering parameters still substantially raise the computational costs of optimal search and the decomposition stage at each iteration, leading to a longer approximation time. If greedy search is employed instead, MK-CTA may not eventually converge at all. In this case, we can keep track of the approximation error at the end of the clustering stage and simply restore the decomposed results of the previous iteration when the approximation error increases.

## 6. CONCLUSIONS

This article presents a generalized multilinear model, namely MK-CTA, to allow a compact, efficient, and accurate representation for rendering large-scale multidimensional visual datasets. MK-CTA overcomes the major drawback of previous tensor-based methods to sparsely cluster a multidimensional dataset along more than one dimension. This novel multiway sparse clustering concept is the key to high-performance photorealistic data-driven rendering, since intra- and inter-cluster coherence among different dimensions of the input data can be well exploited. Moreover, the experimental results of the three common types of complex multidimensional visual datasets, including BTFs, TVLFs, and TVVD, further demonstrate the promising potentials of MK-CTA over $N$-SVD and K-CTA.

Although not shown and discussed in this article, we believe that MK-CTA is not limited to spatially- or time-varying visual datasets. It is actually a general and efficient learning/analysis method for various multidimensional scientific datasets in real-time applications, especially when data variations are large among multiple dimensions and only a portion of the whole dataset is needed at one time.

In the future, we would like to investigate the possibilities of applying MK-CTA to additional multidimensional data-driven applications beyond computer graphics, such as face recognition, image retrieval, and document classification. We are also interested in improving the scalability of MK-CTA to solve some issues about big data analysis.

## APPENDIX

To sparsely cluster the $n$-th mode, each mode-$n$ sub-tensor is separately processed by fixing the core tensor and basis matrices (other than the mode-$n$ one) of each cluster. For the $i$-th mode-$n$ sub-tensor, Eq. (3) can be simplified into the following constrained least-squares optimization subproblem:

$$\min_{\left\{\left(\mathbf{U}_{n,\mathbf{c}}\right)_{i*}\right\}_{\mathbf{c}\in\mathbf{C}}} \left\| \boldsymbol{\mathcal{A}}_{\langle n,i\rangle} - \sum_{\mathbf{c}\in\mathbf{C}} \Big( \boldsymbol{\mathcal{Z}}_{\mathbf{c}} \times_n \left(\mathbf{U}_{n,\mathbf{c}}\right)_{i*} \bigtimes_{\substack{n'=1 \\ n'\neq n}}^{N}{}_{n'} \mathbf{U}_{n',\mathbf{c}} \Big) \right\|_F^2,$$

$$\text{s. t.} \begin{cases} \forall c_1,\cdots,\forall c_{n-1},\forall c_{n+1},\cdots,\forall c_N, \\ \qquad \sum_{c_n=1}^{C_n} \left\| \left(\mathbf{U}_{n,\mathbf{c}}\right)_{i*} \right\|_0 = K_n R_n, \\ \forall \mathbf{c},\ \left\| \left(\mathbf{U}_{n,\mathbf{c}}\right)_{i*} \right\|_0 \in \{0, R_n\}, \\ \forall j, \forall \mathbf{c},\mathbf{c}' \in \mathbf{C}_{n,j},\ \left\| \left(\mathbf{U}_{n,\mathbf{c}}\right)_{i*} \right\|_0 = \left\| \left(\mathbf{U}_{n,\mathbf{c}'}\right)_{i*} \right\|_0, \end{cases} \tag{19}$$

where the orthonormal constraints on the mode-$n$ basis matrix of each cluster are omitted, since they can be satisfied by the post-process in Section 3.5.3.

To solve Eq. (19), optimal search employs a brute-force method to derive an optimal solution from all possible $K_n$-combinations of $\{1,\ldots,C_n\}$. For a possible case of the $K_n$-combinations, for example $c_{n,i}^{(1)},\ldots,c_{n,i}^{(K_n)}$, the constraints in Eq. (19) can be satisfied by setting the $i$-th row of $\mathbf{U}_{n,\mathbf{c}}$ to zeros for all $\mathbf{c} \notin \bigcup_{k=1}^{K_n} \mathbf{C}_{n,c_{n,i}^{(k)}}$. In this way, Eq. (19) can be rewritten in the matrix form as the following unconstrained least-squares optimization problem:

$$\min_{\left\{\left(\mathbf{U}_{n,\mathbf{c}}\right)_{i*}\right\}_{\mathbf{c}\in\mathbf{C}'}} \left\| uf_n\big(\boldsymbol{\mathcal{A}}_{\langle n,i\rangle}\big)^T - \mathbf{Z}_{n,i}^{(K_n)} \mathbf{u}_{n,i}^{(K_n)} \right\|_2^2, \tag{20}$$

which comes from the equivalence between the Frobenius norm of a mode-$n$ sub-tensor and the $\ell^2$ norm of the mode-$n$ unfolded *vector* of that sub-tensor [Tsai and Shih 2012, Appendix A.1]. By identifying that the optimal solution to $\mathbf{u}_{n,i}^{(K_n)}$ is the least-squares estimation of $\mathbf{u}_{n,i}^{(K_n)}$ for $\mathbf{Z}_{n,i}^{(K_n)}\mathbf{u}_{n,i}^{(K_n)} = uf_n\big(\boldsymbol{\mathcal{A}}_{\langle n,i\rangle}\big)^T$, Eq. (8) thus is proved.

Then, by substituting $\mathbf{u}_{n,i}^{(K_n)}$ with its optimal solution, namely Eq. (8), the objective function in Eq. (20) becomes

$$\begin{aligned} &\left\| uf_n\big(\boldsymbol{\mathcal{A}}_{\langle n,i\rangle}\big)^T - \mathbf{Z}_{n,i}^{(K_n)} \big(\mathbf{Z}_{n,i}^{(K_n)}\big)^{\dagger} uf_n\big(\boldsymbol{\mathcal{A}}_{\langle n,i\rangle}\big)^T \right\|_2^2 \qquad (21) \\ =& \left\| uf_n\big(\boldsymbol{\mathcal{A}}_{\langle n,i\rangle}\big) \right\|_2^2 - 2 \cdot uf_n\big(\boldsymbol{\mathcal{A}}_{\langle n,i\rangle}\big) \mathbf{Z}_{n,i}^{(K_n)} \big(\mathbf{Z}_{n,i}^{(K_n)}\big)^{\dagger} uf_n\big(\boldsymbol{\mathcal{A}}_{\langle n,i\rangle}\big)^T \\ &+ \left\| \mathbf{Z}_{n,i}^{(K_n)} \big(\mathbf{Z}_{n,i}^{(K_n)}\big)^{\dagger} uf_n\big(\boldsymbol{\mathcal{A}}_{\langle n,i\rangle}\big)^T \right\|_2^2 \end{aligned}$$

$$= \left\| uf_n\left(\boldsymbol{\mathcal{A}}_{\langle n,i\rangle}\right) \right\|_2^2 - uf_n\left(\boldsymbol{\mathcal{A}}_{\langle n,i\rangle}\right) \mathbf{Z}_{n,i}^{(K_n)} \left(\mathbf{Z}_{n,i}^{(K_n)}\right)^{\dagger} uf_n\left(\boldsymbol{\mathcal{A}}_{\langle n,i\rangle}\right)^T, \quad (22)$$

where we expand the $\ell^2$ norm into the matrix form and apply the identity that $\mathbf{Z}_{n,i}^{(K_n)} = \left(\left(\mathbf{Z}_{n,i}^{(K_n)}\right)^{\dagger}\right)^T \left(\mathbf{Z}_{n,i}^{(K_n)}\right)^T \mathbf{Z}_{n,i}^{(K_n)}$. Since the first term in Eq. (22) is a constant, minimizing Eq. (21) is equivalent to maximizing the second term in Eq. (22). The objective function in Eq. (4) thus is proved.

## ACKNOWLEDGMENTS

The author would like to thank anonymous reviewers for their profound comments and suggestions. The model "Bunny" was provided in courtesy of the Stanford Computer Graphics Laboratory, from the Stanford 3D Scanning Repository, http://graphics.stanford.edu/data/3Dscanrep/.

## REFERENCES

Michal Aharon, Michael Elad, and Alfred M. Bruckstein. 2006. K-SVD: An Algorithm for Designing Overcomplete Dictionaries for Sparse Representation. *IEEE Trans. Signal Process.* 54, 11 (2006), 4311–4322.

Haim Avron, Andrei Sharf, Chen Greif, and Daniel Cohen-Or. 2010. $\ell_1$-Sparse Reconstruction of Sharp Point Set Surfaces. *ACM Trans. Graph.* 29, 5 (2010).

Marcos Balsa Rodríguez, Enrico Gobbetti, José Antonio Iglesias Guitián, Maxim Makhinya, Fabio Marton, Renato B. Pajarola, and Susanne K. Suter. 2014. State-of-the-Art in Compressed GPU-Based Direct Volume Rendering. *Comput. Graph. Forum* 33, 6 (2014), 77–100.

Yosuke Bando, Henry Holtzman, and Ramesh Raskar. 2013. Near-Invariant Blur for Depth and 2D Motion via Time-Varying Light Field Analysis. *ACM Trans. Graph.* 32, 2 (2013).

Arindam Banerjee, Sugato Basu, and Srujana Merugu. 2007. Multiway Clustering on Relation Graphs. In *Proc. of SDM '07*. 145–156.

Ron Bekkerman, Ran El-Yaniv, and Andrew Kachites McCallum. 2005. Multiway Distributional Clustering via Pairwise Interactions. In *Proc. of ICML '05*. 41–48.

Thomas Blumensath and Mike E. Davies. 2007. *On the Difference between Orthogonal Matching Pursuit and Orthogonal Least Squares*. Technical Report.

Kristin J. Dana, Bram Van Ginneken, Shree Kumar Nayar, and Jan J. Koenderink. 1999. Reflectance and Texture of Real-World Surfaces. *ACM Trans. Graph.* 18, 1 (1999), 1–34.

Geoffrey M. Davis, Stéphane Georges Mallat, and Marco Avellaneda. 1997. Adaptive Greedy Approximations. *Constr. Approx.* 13, 1 (1997), 57–98.

Lieven De Lathauwer, Bart De Moor, and Joos Vandewalle. 2000. On the Best Rank-1 and Rank-$(R_1, R_2, \ldots, R_n)$ Approximation of Higher-Order Tensors. *SIAM J. Matrix Anal. Appl.* 21, 4 (2000), 1324–1342.

Paul Ernest Debevec. 1998. Rendering Synthetic Objects into Real Scenes: Bridging Traditional and Image-based Graphics with Global Illumination and High Dynamic Range Photography. In *Proc. of ACM SIGGRAPH '98*. 189–198.

Michael Elad, Mário A. T. Figueiredo, and Yi Ma. 2010. On the Role of Sparse and Redundant Representations in Image Processing. *Proc. of the IEEE* 98, 6 (2010), 972–982.

Klaus Jürgen Engel, Martin Kraus, and Thomas Ertl. 2001. High-Quality Pre-Integrated Volume Rendering Using Hardware-Accelerated Pixel Shading. In *Proc. of EGGH '01*. 9–16.

Jiří Filip and Michal Haindl. 2009. Bidirectional Texture Function Modeling: A State of the Art Survey. *IEEE Trans. Pattern Anal. Mach. Intell.* 31, 11 (2009), 1921–1940.

Nathaniel Richard Fout and Kwan-Liu Ma. 2007. Transform Coding for Hardware-Accelerated Volume Rendering. *IEEE Trans. Vis. Comput. Graph.* 13, 6 (2007), 1600–1607.

Enrico Gobbetti, José Antonio Iglesias Guitián, and Fabio Marton. 2012. COVRA: A Compression-Domain Output-Sensitive Volume Rendering Architecture Based on a Sparse Representation of Voxel Blocks. *Comput. Graph. Forum* 31, 3 (2012), 1315–1324.

Steven J. Gortler, Radek Grzeszczuk, Richard Szeliski, and Michael F. Cohen. 1996. The Lumigraph. In *Proc. of ACM SIGGRAPH '96*. 43–54. DOI:http://dx.doi.org/10.1145/237170.237200

Stefan Guthe and Wolfgang Straßer. 2001. Real-Time Decompression and Visualization of Animated Volume Data. In *Proc. of IEEE VIS '01*. 349–356.

Miloš Hašan, Edgar Velázquez-Armendáriz, Fabio Pellacini, and Kavita Bala. 2008. Tensor Clustering for Rendering Many-Light Animations. *Comput. Graph. Forum* 27, 4 (2008), 1105–1114.

Vlastimil Havran, Jiří Filip, and Karol Myszkowski. 2010. Bidirectional Texture Function Compression Based on Multi-Level Vector Quantization. *Comput. Graph. Forum* 29, 1 (2010), 175–190.

Tamara G. Kolda and Brett W. Bader. 2009. Tensor Decompositions and Applications. *SIAM Rev.* 51, 3 (2009), 455–500.

Melissa Linae Koudelka, Sebastian Magda, Peter Neil Belhumeur, and David Jay Kriegman. 2003. Acquisition, Compression, and Synthesis of Bidirectional Texture Functions. In *Proc. of Texture '03*. 59–64.

Jens Krüger and Rüdiger Westermann. 2003. Acceleration Techniques for GPU-based Volume Rendering. In *Proc. of IEEE VIS '03*. 287–292.

Marc Levoy. 2006. Light Fields and Computational Imaging. *IEEE Computer* 39, 8 (2006), 46–55.

Marc Levoy and Pat Hanrahan. 1996. Light Field Rendering. In *Proc. of ACM SIGGRAPH '96*. 31–42.

Eric Brian Lum, Kwan-Liu Ma, and John Clyne. 2002. A Hardware-Assisted Scalable Solution for Interactive Volume Rendering of Time-Varying Data. *IEEE Trans. Vis. Comput. Graph.* 8, 3 (2002), 286–301.

Pieter Peers, Dhruv Mahajan, Bruce Lamond, Abhijeet Ghosh, Wojciech Matusik, Ravi Ramamoorthi, and Paul Ernest Debevec. 2009. Compressive Light Transport Sensing. *ACM Trans. Graph.* 28, 1 (2009).

Ravi Ramamoorthi. 2009. Precomputation-Based Rendering. *Found. Trends Comput. Graph. Vis.* 3, 4 (2009), 281–369.

Roland Ruiters and Reinhard Klein. 2009. BTF Compression via Sparse Tensor Decomposition. *Comput. Graph. Forum* 28, 4 (2009), 1181–1188.

Jens Schneider and Rüdiger Westermann. 2003. Compression Domain Volume Rendering. In *Proc. of IEEE VIS '03*. 293–300.

Pradeep Sen and Soheil Darabi. 2011. Compressive Rendering: A Rendering Application of Compressed Sensing. *IEEE Trans. Vis. Comput. Graph.* 17, 4 (2011), 487–499.

Amnon Shashua, Ron Zass, and Tamir Hazan. 2006. Multiway Clustering Using Super-Symmetric Non-Negative Tensor Factorization. In *Proc. of ECCV '06, Part IV*. 595–608.

Heung-Yeung Shum, Shing-Chow Chan, and Sing Bing Kang. 2007. *Image-Based Rendering*. Springer-Verlag Press.

Charles Soussen, Rémi Gribonval, Jérôme Idier, and Cédric Herzet. 2013. Joint $k$-Step Analysis of Orthogonal Matching Pursuit and Orthogonal Least Squares. *IEEE Trans. Inf. Theory* 59, 5 (2013), 3158–3174.

Jean-Luc Starck and Jérôme Bobin. 2010. Astronomical Data Analysis and Sparsity: From Wavelets to Compressed Sensing. *Proc. of the IEEE* 98, 6 (2010), 1021–1030.

Xin Sun, Qiming Hou, Zhong Ren, Kun Zhou, and Baining Guo. 2011. Radiance Transfer Biclustering for Real-Time All-Frequency Biscale Rendering. *IEEE Trans. Vis. Comput. Graph.* 17, 1 (2011), 64–73.

Xin Sun, Kun Zhou, Yanyun Chen, Stephen Lin, Jiaoying Shi, and Baining Guo. 2007. Interactive Relighting with Dynamic BRDFs. *ACM Trans. Graph.* 26, 3 (2007).

Susanne K. Suter, José Antonio Iglesias Guitián, Fabio Marton, Marco Agus, Andreas Elsener, Christoph P. E. Zollikofer, Meenakshisundaram Gopi, Enrico Gobbetti, and Renato B. Pajarola. 2011. Interactive Multiscale Tensor Reconstruction for Multiresolution Volume Visualization. *IEEE Trans. Vis. Comput. Graph.* 17, 12 (2011), 2135–2143.

Susanne K. Suter, Maxim Makhynia, and Renato B. Pajarola. 2013. TAMRESH - Tensor Approximation Multiresolution Hierarchy for Interactive Volume Visualization. *Comput. Graph. Forum* 32, 3 (2013), 151–160.

Yu-Ting Tsai, Kuei-Li Fang, Wen-Chieh Lin, and Zen-Chung Shih. 2011. Modeling Bidirectional Texture Functions with Multivariate Spherical Radial Basis Functions. *IEEE Trans. Pattern Anal. Mach. Intell.* 33, 7 (2011), 1356–1369.

Yu-Ting Tsai and Zen-Chung Shih. 2006. All-Frequency Precomputed Radiance Transfer Using Spherical Radial Basis Functions and Clustered Tensor Approximation. *ACM Trans. Graph.* 25, 3 (2006), 967–976.

Yu-Ting Tsai and Zen-Chung Shih. 2012. K-Clustered Tensor Approximation: A Sparse Multilinear Model for Real-Time Rendering. *ACM Trans. Graph.* 31, 3 (2012).

M. Alex O. Vasilescu and Demetri Terzopoulos. 2004. TensorTextures: Multilinear Image-Based Rendering. *ACM Trans. Graph.* 23, 3 (2004), 336–342.

Chaoli Wang, Hongfeng Yu, and Kwan-Liu Ma. 2010. Application-Driven Compression for Visualizing Large-Scale Time-Varying Data. *IEEE Comput. Graph. Appl.* 30, 1 (2010), 59–69.

Hongcheng Wang, Qing Wu, Lin Shi, Yizhou Yu, and Narendra Ahuja. 2005. Out-of-Core Tensor Approximation of Multidimensional Matrices of Visual Data. *ACM Trans. Graph.* 24, 3 (2005), 527–535.

Gordon Wetzstein, Douglas Lanman, Wolfgang Heidrich, and Ramesh Raskar. 2011. Layered 3D: Tomographic Image Synthesis for Attenuation-Based Light Field and High Dynamic Range Displays. *ACM Trans. Graph.* 30, 4 (2011).

Gordon Wetzstein, Douglas Lanman, Matthew Hirsch, and Ramesh Raskar. 2012. Tensor Displays: Compressive Light Field Synthesis Using Multilayer Displays with Directional Backlighting. *ACM Trans. Graph.* 31, 4 (2012).

John Wright, Yi Ma, Julien Mairal, Guillermo R. Spairo, Thomas S. Huang, and Shuicheng Yan. 2010. Sparse Representation for Computer Vision and Pattern Recognition. *Proc. of the IEEE* 98, 6 (2010), 1031–1044.

Qing Wu, Tian Xia, Chun Chen, Hsueh-Yi Sean Lin, Hongcheng Wang, and Yizhou Yu. 2008. Hierarchical Tensor Approximation of Multidimensional Visual Data. *IEEE Trans. Vis. Comput. Graph.* 14, 1 (2008), 186–199.

Received November 2014; revised March 2015; accepted March 2015