Eurographics Conference on Visualization (EuroVis) 2013
B. Preim, P. Rheingans, and H. Theisel
(Guest Editors)

Volume 32 (2013), Number 3

TAMRESH: Tensor Approximation Multiresolution Hierarchy for Interactive Volume Visualization

Susanne K. Suter & Maxim Makhynia & Renato Pajarola

Visualization and Multimedia Lab, University of Zurich, Switzerland

Abstract

Interactive visual analysis of large and complex volume datasets is an ongoing and challenging problem. We tackle this challenge in the context of state-of-the-art out-of-core multiresolution volume rendering by introducing a novel hierarchical tensor approximation (TA) volume visualization approach. The TA framework allows us (a) to use a rank-truncated basis for compact volume representation, (b) to visualize features at multiple scales, and (c) to visualize the data at multiple resolutions. In this paper, we exploit the special properties of the TA factor matrix bases and define a novel multiscale and multiresolution volume rendering hierarchy. Different from previous approaches, to represent one volume dataset we use but one set of global bases (TA factor matrices) to reconstruct at all resolution levels and feature scales. In particular, we propose a coupling of multiscalable feature visualization and multiresolution DVR through the properties of global TA bases. We demonstrate our novel TA multiresolution hierarchy *based volume representation and visualization on a number of μCT volume datasets.*

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Picture/Image Generation—Viewing algorithms I.4.7 [Computer Graphics]: Feature Measurement—Feature representation

1. Introduction

Continuing advances in 3D imaging technologies, e.g., phase-contrast Synchrotron Tomography (pcST) and micro-computed X-ray tomography (μCT), and computer simulations lead to ever growing and complex volume datasets. These volumes are not only large but exhibit an increasing complexity of internal structure, showing high variability of spatial frequencies and patterns at different scales.

Interactive *direct volume rendering* (DVR) is an important tool for explorative visual analysis of the spatial distribution and structure of features within the volumetric data. A number of algorithms are available for real-time rendering of moderately sized volumes on desktop platforms [EHK*06]. For larger datasets, adaptive data reduction and *level of detail* (LOD) methods must be integrated with bricking and out-of-core rendering techniques. Hierarchical, out-of-core and multiresolution approaches have been applied successfully to trade-off visual quality for rendering performance (e.g., [LWP*06, GMG08, CNLE09, SIGM*11]).

Aggregation of volume data in a spatial (octree) hierarchy provides multiple levels of spatial resolutions for fast LOD based rendering solutions. The basic hierarchical averaging property, however, only allows for simple blurred, reduced resolution representations. Approaches like Fourier (FT), discrete cosine (DCT) and wavelet transforms (WT) can provide information about the frequency patterns of the data at different scales. Unlike FT, DCT and WT which employ fixed-basis decompositions, however, singular value decomposition (SVD), principal component analysis (PCA) or higher-order tensor approximation (TA) (see [KB09]) methods derive learned basis decompositions, which may capture more compact data-specific structures and patterns.

TA has shown to be a viable tool for compact multiscale volume feature representation [SZP10]. Our goal is to combine a TA based volume representation for data reduction and multiscale feature reconstruction with a hierarchical view-dependent variable resolution rendering, eventually supporting independent control of data reconstruction at different features scales as well as spatial resolutions.

In this context, we demonstrate what a good TA hierarchy is, as it is not a straight forward solution, as well as how it can effectively be constructed and used in an interactive large scale volume visualization application. We identify and describe the important properties of the TA framework and its factor matrices that the solution presented in this paper is based on.

2. Related Work

A number of methods for compact data representation as a linear combination of bases exists that can be differentiated by having *predefined* data-independent or *learned* data-specific bases. In volume rendering, predefined bases like FT, DCT or WT and/or learned bases like vector quantization (VQ), tensor approximation (TA), or sparse coding have been successfully applied (see e.g., [GWGS02, SW03, FM07, SIGM*11, GIM12]). While predefined basis decompositions may be cheaper to produce, learned bases have the potential for improved and more compact data fitting [WWS*05,WXC*08,SZP10]. Actually, the drawback of computing the bases is not real-time critical since the bases are computed in a preprocessing stage. For the above reasons we follow a data-specific basis decomposition approach.

We use the TA framework [KB09], as previously used individually for multiscale volume visualization [SZP10] and for multiresolution volume rendering [SIGM*11]. So far, no combined tensor-based multiscale and multiresolution model has been proposed. Hierarchical TAs have previously been developed [WXC*08, SIGM*11]), where subvolume bricks are represented as tensor decompositions in the nodes of an octree.

In this work, we present an integration of multiscale feature reconstruction and multiresolution volume rendering by use of global TA factor matrices. For this purpose, we identify and exploit special properties of the TA framework and its factor matrices as described in this paper. In particular, we show how to take advantage of the spatial selectivity and subsampling in the spatial dimension I and of the progressive truncation in the tensor rank dimension R of factor matrices, respectively for adaptive multiresolution (along I) and multiscale (along R) reconstruction of individual volume bricks. In fact, we present a novel solution for a hierarchical TA representation, including reduction of brick border artifacts, with fully independent brick reconstruction at variable spatial resolutions and feature approximation scales.

Our proposed TA decomposition is carried out in a preprocess on datasets larger than the available main memory, demanding out-of-core computing approaches. Wang et al. [WWS*05] proposed a localized per brick based out-of-core TA algorithm. Our global factor matrices, however, require a complete full volume decomposition and thus we optimize the process by using memory mapped files and sequentialized data access. Furthermore, the tensor decomposition is optimized using *OpenMP* parallelized tensor times matrix (TTM) multiplications. The runtime reconstruction of bricks, is implemented similar to [SIGM*11] by GPU accelerated TTM multiplications. Hence, we can keep a few global mipmapped factor matrices permanently in fast read-only GPU memory, and we only need to upload and cache the demanded core tensor data for bricks.

3. Tensor Approximation Factor Matrix Properties

In tensor approximation (TA) as summarized in [KB09], a volume dataset can be represented as a third-order tensor (multidimensional array) $\mathscr{A} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$ with elements $a_{i_1 i_2 i_3}$. TA is a higher-order generalization of the matrix SVD/PCA that exhibits the nice properties of (a) rank-R decomposition and (b) orthonormal row-space and column-space vectors. In higher orders, the rank-R decomposition is achieved with the so-called CP model, while the orthonormal vectors are preserved in the so-called Tucker model. In the following, we refer to the Tucker model.

The Tucker model consists of one factor matrix per mode (data dimension) $\mathbf{U}^{(n)} \in \mathbb{R}^{I_n \times R_n}$ and one core tensor $\mathscr{B} \in \mathbb{R}^{R_1 \times R_2 \times R_3}$ (for volumes). The core tensor $\mathscr{B}$ is a projection of the original data $\mathscr{A}$ onto the basis of the factor matrices $\mathbf{U}^{(n)}$. In case of a volume, the Tucker model has three modes as illustrated in Fig. 1, and defines an approximation $\widetilde{\mathscr{A}}$ of the original volume $\mathscr{A}$ (using n-mode products $\times_n$).

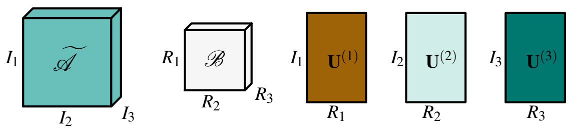


Figure 1: Approximation: $\widetilde{\mathscr{A}} = \mathscr{B} \times_1 \mathbf{U}^{(1)} \times_2 \mathbf{U}^{(2)} \times_3 \mathbf{U}^{(3)}$

The two axes of the factor matrices represent two different spaces: The rows correspond to the spatial dimension in the corresponding mode, and the columns to the feature scale (see Fig. 2). Next, we show how these properties can be exploited for multiresolution modeling (spatial selection and subsampling of rows) and multiscale feature visualization (rank truncation of the columns).

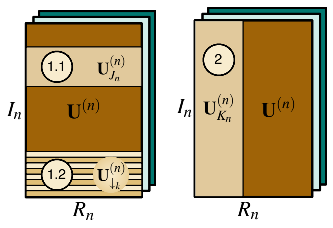


Figure 2: Factor matrix properties: (1.1) spatial selectivity and (1.2) subsampling in the spatial dimension, (2) feature scale reduction along the rank dimension.

3.1. Spatial Selection and Reduction

3.1.1. Spatial Selectivity

For view-frustum culling and adaptive brick selection in interactive multiresolution volume visualization, efficient access to spatially restricted subvolumes is required. Since a

TA factor matrix's rows directly correspond to its spatial dimension, we can exploit this fact for the reconstruction of a subvolume directly from the global factor matrices. We first describe the spatial selection for a fixed resolution.

The Tucker model defines an approximation of a volume $\mathscr{A}$ by the decomposition $\widetilde{\mathscr{A}} = \mathscr{B} \times_1 \mathbf{U}^{(1)} \times_2 \mathbf{U}^{(2)} \times_3 \mathbf{U}^{(3)}$, and each element of $\widetilde{\mathscr{A}}$ is defined as

$$\widetilde{a}_{i_1 i_2 i_3} = \sum_{r_1}\sum_{r_2}\sum_{r_3} b_{r_1 r_2 r_3} \cdot u^{(1)}_{i_1 r_1} \cdot u^{(2)}_{i_2 r_2} \cdot u^{(3)}_{i_3 r_3}, \qquad (1)$$

with factor matrix and core tensor entries $u^{(n)}_{i_n r_n}$ and $b_{r_1 r_2 r_3}$.

Due to the correspondence of the rows of $\mathbf{U}^{(n)}$ to the spatial dimension n (see Fig. 2(a)), we can define row-index subranges $J_n \subseteq [0 \dots I_n]$ to reconstruct a spatial subvolume $J_1 \times J_2 \times J_3$ using only the indices $i_n \in J_n$ in Eq. 1. As shown in Fig. 3, we can thus select and reconstruct a subvolume (e.g., an octree brick) by choosing a subset of the row vectors of all factor matrices. Using these row-block submatrices $\mathbf{U}^{(n)}_{J_n}$ we can formulate the subvolume reconstruction as

$$\widetilde{\mathscr{A}}_{J_1 \times J_2 \times J_3} = \mathscr{B} \times_1 \mathbf{U}^{(1)}_{J_1} \times_2 \mathbf{U}^{(2)}_{J_2} \times_3 \mathbf{U}^{(3)}_{J_3}. \qquad (2)$$

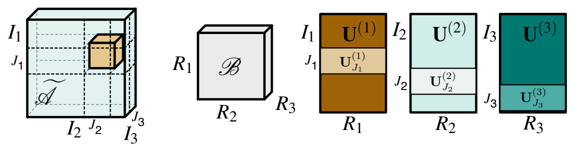


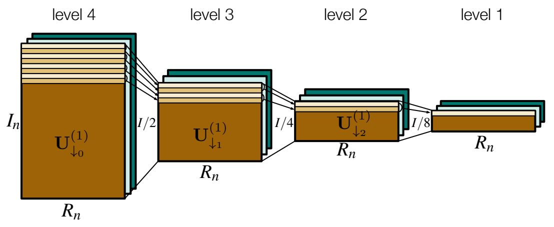
Figure 3: Illustration of spatial selectivity: A range of selected submatrix rows reconstructs a well defined subvolume (in brown) of the original whole dataset.

In Fig. 6 an example of two selected octree bricks (1 and 2) is illustrated. For the two different subvolumes, we selected the factor matrix row vectors corresponding to the position of the subvolume in the input dataset.

3.1.2. Subsampling

In multiresolution volume rendering, lower resolutions of subvolumes are used for view-dependent adaptive LOD rendering. Due to the correspondence of factor-matrix rows to the spatial dimensions as outlined above, we can use row-subsampling on factor matrices before brick reconstruction from the TA to achieve lower resolution representations.

Since the I_n rows of a factor matrix $\mathbf{U}^{(n)}$ correspond to the resolution of the volume $\widetilde{\mathscr{A}}$ in that mode, we can construct a lower-resolution reconstruction in the n-th dimension by averaging (pairs of) rows to get a downsampled matrix $\mathbf{U}^{(n)}_{\downarrow 1}$ (with $I_n/2$ rows). This is possible because the columns of a factor matrix $\mathbf{U}^{(n)}$ capture the data variation along the n-th dimension. Therefore, downsampling and averaging pairs of rows correspond to halving the reconstructed volume resolution. This downsampling of factor matrices is illustrated in Fig. 4 and corresponds to the principle of *mipmapping*.



level 4
level 3
level 2
level 1
I_n
$\mathbf{U}^{(1)}_{\downarrow 0}$
R_n
$I/2$
$\mathbf{U}^{(1)}_{\downarrow 1}$
R_n
$I/4$
$\mathbf{U}^{(1)}_{\downarrow 2}$
R_n
$I/8$
R_n

Figure 4: Factor-matrix subsampling by pair-wise row averaging generates a mipmapped factor matrix hierarchy.

Fig. 7 shows the factor matrix averaging as used for a hierarchical tensor representation and its effects on the visual reconstruction. The top row uses standard scalar value averaging directly on the input volume, while in the middle we show the direct TA of these subsampled datasets. In the third row we demonstrate the tensor reconstruction based on the subsampled and averaged factor matrices as proposed. As can be seen the reconstructions are extremely close.

3.1.3. Implications to Rank Truncation

The selective usage of the factor matrices, i.e., the projection of bricks onto submatrices, affects the orthogonality properties. Notably, we loose the so-called all-orthogonality property (see [dLdMV00a]) of the core tensor since the TA submatrices do not have orthogonal columns any longer. However, it is critical to maintain the all-orthogonality property since it allows for truncating a tensor decomposition (similar to rank truncation within a matrix SVD).

3.2. Feature Scale and Rank Truncation

The Tucker model defines a rank-(R_1,R_2,R_3) approximation, where a small R_n corresponds to a low-rank approximation (low feature scale with details removed) and a large R_n corresponds to a more accurate approximation of the original (high feature scale). The tensor rank R_n defines the number of coefficients and bases used for the reconstruction. As illustrated in Figs. 2(b) and 5, the rank indicates how many factor matrix columns and corresponding core tensor entries are used for a reconstruction.

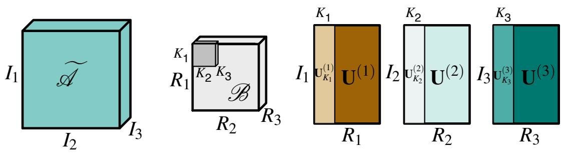


Figure 5: Illustration of a rank truncated reconstruction: Truncated factor matrices with corresponding fewer core tensor entries reconstruct at the full spatial resolution, but at a lower approximation, i.e., at a lower feature scale.

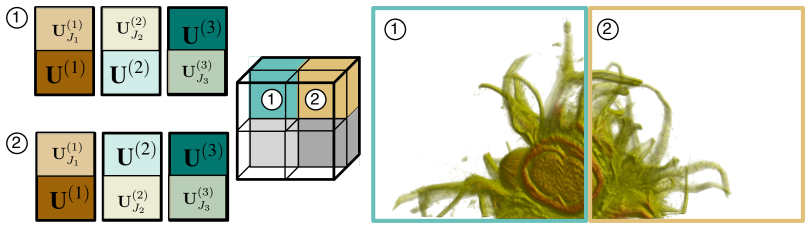


Figure 6: Spatial selectivity: Selected bricks reconstructed by the corresponding selection of row-index factor matrix subranges.

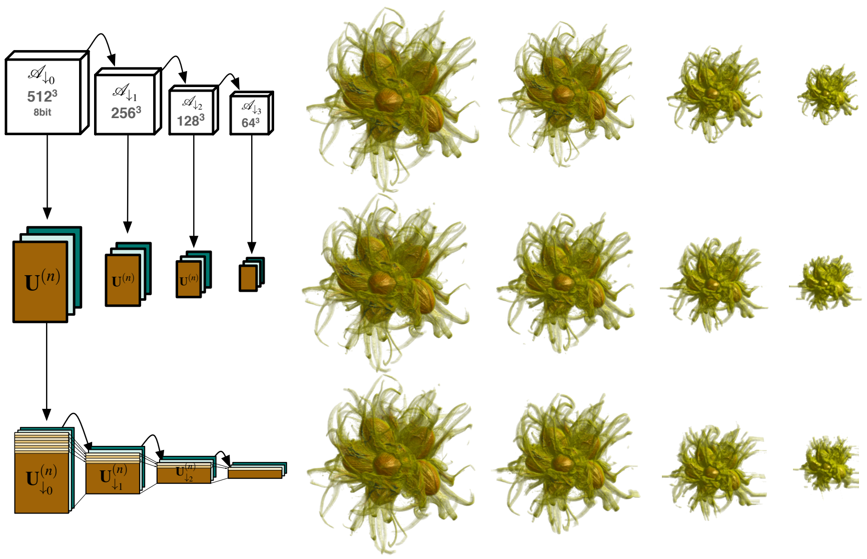

$\mathscr{A}_{\downarrow 0}$ 512^3 8bit
$\mathscr{A}_{\downarrow 1}$ 256^3
$\mathscr{A}_{\downarrow 2}$ 128^3
$\mathscr{A}_{\downarrow 3}$ 64^3
$U^{(n)}$
$U^{(n)}_{\downarrow 0}$
$U^{(n)}_{\downarrow 1}$
$U^{(n)}_{\downarrow 2}$

Figure 7: Factor matrix subsampling (bottom) compared to direct TA (center) derived from original subsampled datasets (top).

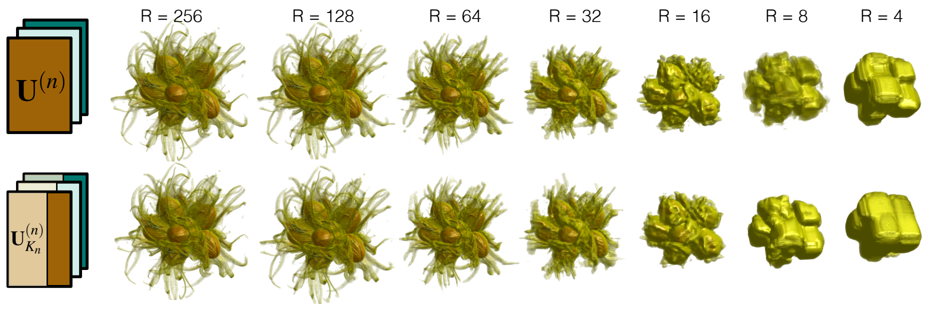


Figure 8: Progressive rank truncations (bottom row) compared to fixed rank-(R,R,R) TAs (top row).

The highest rank R_{init} for the initial Tucker decomposition is chosen explicitly. Still, rank truncations for ranks $K_n < R_{init}$ can be applied after the initial decomposition (similar to the rank truncation in the matrix SVD). Even though the core tensor coefficients are not guaranteed to be in strictly decreasing order, as in the matrix SVD, in practice, progressive tensor rank truncation in the Tucker model works for adaptive data visualization at different feature scales.

Fig. 8 compares the progressive rank truncation from an initial rank-$(256,256,256)$ decomposition (bottom) to a specific fixed rank-(R,R,R) decomposition (top) of a 512^3 volume. Both representations are visually similar down to the lowest ranks, which, however, are hardly used.

3.3. Core Tensor Hierarchy

As previously described, we can observe the different outcomes of applying rank truncation and spatial subsampling as shown in Figs. 7 and 8, corresponding to spatial multiresolution modeling in the former and multiscale structure recovery in the latter case. From the spatial selection, subsampling and rank truncation properties of factor matrices, we derive a novel and efficient TA hierarchy for multiscale and multiresolution volume visualization.

Our new TA hierarchy is illustrated in Fig. 9 and in principle follows an octree subdivision of the input volume dataset. A key part of our approach is, unlike any other TA proposed in visual computing before, that we maintain a global set of mipmapped factor matrices $\mathbf{U}^{(n)}_{\downarrow k}$ that are small enough even for large input datasets to be kept in CPU or GPU memory (see also Sec. 4.3). Thus all nodes on one level l of the octree hierarchy are reconstructed using the same factor matrices $\mathbf{U}^{(n)}_{\downarrow k}$ corresponding to that octree level (with $k = l_{\max} - l$). Unlike the factor matrix rows, the core tensors do not exhibit any mapping to the spatial dimensions. Thus, we define a small core tensor $\mathscr{B}_{\text{brick}\downarrow_k}$ per octree node. Eventually, the core tensors are stored in quantized form ($float32$ to $uint8$, as described in [SIGM*11]).

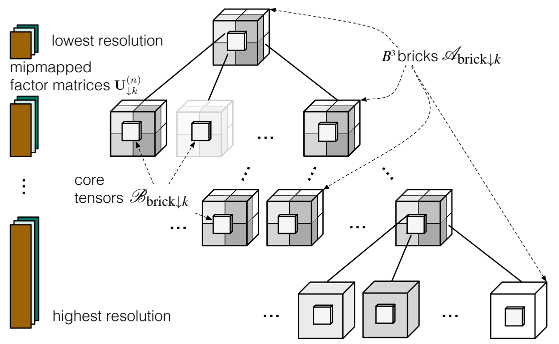

lowest resolution
mipmapped factor matrices $\mathbf{U}^{(n)}_{\downarrow k}$
B^3 bricks $\mathscr{A}_{\text{brick}\downarrow k}$
core tensors $\mathscr{B}_{\text{brick}\downarrow k}$
highest resolution

Figure 9: Octree based on the mipmapped factor matrices.

To solve the issue of maintaining rank-reducible core tensors (see Sec. 3.1.3), we re-span the subspace of each row-block global factor submatrix $\mathbf{U}^{(n)}_{J_n}$ by applying an SVD (similar to [TS12]). The row-block matrices' columns are then replaced with the orthogonal singular vectors, as indicated in Fig. 10. That way, we are able to define per-brick core tensors that can be truncated. Intuitively, this recomposition of the global matrices can be seen as a different representation of the same local subspaces as defined by the initial non-orthogonal submatrices. J_n corresponds to the brick size (including borders) of the octree hierarchy. Due to equally sized bricks along all spatial octree directions, the sub-block replacements can be used for spatially-corresponding bricks.

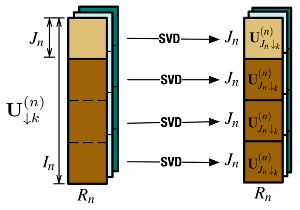

J_n
$\mathbf{U}^{(n)}_{\downarrow k}$
I_n
R_n
SVD
J_n
$\mathbf{U}^{(n)}_{J_n\downarrow k}$
R_n

Figure 10: Processing the mipmapped initial global factor matrices in order to obtain orthogonal localized row-block submatrices and thus all-orthogonal per-brick core tensors.

Our TA hierarchy is hence defined by a set of global mipmapped factor matrices $\mathbf{U}^{(n)}_{\downarrow k}$ and an octree hierarchy that stores quantized core tensors $\mathscr{B}_{\text{brick}\downarrow_k}$. Reconstruction is now possible in a flexible way, according to a desired spatial resolution by choosing the octree level, and adapting the feature scale by adjusting the rank truncation level $8 \le R_n \le R_{init}$ (ranks less than 8 seem not useful).

4. Hierarchical Global TA Factor Matrices

After describing the factor matrix properties that form the basis of our multiscale and multiresolution TA octree hierarchy, we now elaborate on how to compute the initial tensor decomposition needed to (1) derive the mipmapped factor matrices and subsequently (2) generate the core tensor octree hierarchy.

4.1. Initial Tensor Decomposition

The initial decomposition is performed as a Tucker tensor decomposition of the full size input volume. The actual method is based on an *alternating least-squares* (ALS) algorithm implemented as a *higher-order orthogonal iteration* (HOOI) [dLdMV00b]. One iteration of the HOOI ALS for a third-order tensor (a volume) consists of three optimization steps, one along each mode (see Alg. 1).

Algorithm 1 HOOI optimization of one mode (e.g. $n = 1$).

```
1: for mode $n$ optimization do
2:     TTM of tensor $\mathscr{A}_{I_1\times I_2\times I_3}$ times ${\mathbf{U}^{(2)}_{I_2\times R_2}}^T \to \mathscr{T}_{I_1\times R_2\times I_3}$
3:     TTM of tensor $\mathscr{T}_{I_1\times R_2\times I_3}$ times ${\mathbf{U}^{(3)}_{I_3\times R_3}}^T \to \mathscr{P}_{I_1\times R_2\times R_3}$
4:     HOSVD on $\mathbf{P}_n$ (unfolded $\mathscr{P}$ along mode $n$)
5: end for
```

The HOOI decomposition of large datasets exhibits two main bottlenecks: (1) the tensor times matrix multiplications (TTMs) of large tensors as in line 2 of Alg. 1, and (2) the HOSVD as in line 4 of Alg. 1. We addressed those bottlenecks as follows:

1. Larger than main memory data tensors $\mathscr{A}$ are unfolded in the required mode direction ($\to \mathbf{A}_n$) once and accessed from memory mapped files.
2. The HOOI iterations are designed such that the TTM operations access the unfoldings of $\mathbf{A}_n$ sequentially and memory aligned.
3. The TTM operation has been implemented using parallel multi-threaded matrix-matrix multiplications.
4. The HOSVD is computed either based on the SVD or on the symmetric eigenvalue decomposition (EIGS), depending on the size of the input tensor.

The algorithms used for the large tensor decomposition have been integrated into an open source vector and matrix math library *vmmlib* [vmm]. Specifically, we use a wrapper to BLAS DGEMM and OpenMP for all matrix-matrix multiplications (TTMs and covariance), and LAPACK wrappers for the SVD and the EIGS. The symmetric eigenvalue decomposition is extended to return the first R largest magnitude eigenvalues with their eigenvectors corresponding in the $\mathbf{C}_n = \mathbf{P}_n \times {\mathbf{P}_n}^T$ covariance matrix scenario to the R first left singular vectors that are used as TA factor matrices $\mathbf{U}^{(n)}$ (see [dLdMV00a]). To further save computing time and memory, we initialize the factor matrices for the ALS with random values as in [WWS*05].

The choice of the initial rank R_{init} is an important factor of the computational cost of the initial tensor decomposition. The smaller R_{init}, the fewer computations in bottleneck (1) and bottleneck (2) are needed. The typical setting for a reduced rank TA would be half of the dimension of the input volume $\mathscr{A}$, thus $R_{init} = \frac{I}{2}$, see [WXC*08,SZP10,SIGM*11]. However, in the octree, we have much smaller brick dimensions, e.g. of 64 only, thus we are aiming at final ranks in the range of $R_n \leq 32$. We could start the initial decomposition with large R_{init} to get the initial global factor matrices $\mathbf{U}^{(n)} \in \mathbb{R}^{I_n \times \frac{I_n}{2}}$, only to then reduce the number of columns from $\frac{I_n}{2}$ to 32. This would be costly, and moreover, our reconstruction experiments have shown that an initial rank close to the final brick core rank was favorable. Thus we perform the initial tensor decomposition with R_{init} corresponding to half of the octree node brick size.

4.2. Octree Building

The basic multiresolution octree structure has been described in Sec. 3.3 and is illustrated in Fig. 9. The data of an octree node corresponding to an input subvolume $\mathscr{A}_{J_1 \times J_2 \times J_3}$ is given by its core tensor $\mathscr{B}_{R_1 \times R_2 \times R_3}$, whose coefficients model the relationship between the global factor matrices and the original data. Given the mipmapped factor matrices $\mathbf{U}^{(n)}_{\downarrow k}$, with R_{init} columns each, we compute the core tensor $\mathscr{B}_{R_1 \times R_2 \times R_3}$ per octree node by projecting the input subvolume brick $\mathscr{A}_{J_1 \times J_2 \times J_3}$, at the required spatial resolution, onto the row-block SVD factor submatrices $\mathbf{U}^{(n)}_{J_n \downarrow k}$ of the appropriate subsampling level k as described in Sec. 3.3. The octree building follows state-of-the-art implementations and is described for TAMRESH-specific modifications in the supplementary material.

Furthermore, to reduce border artifacts between bricks during rendering, we introduce a 6-voxel-border overlap for the per-brick core tensor generation. 4-voxels are used to provide the local brick TA information about the neighboring bricks; 2-voxels are used for the gradient interpolation between bricks during rendering.

Without restricting the generality of the described concepts, in the following we assume a subvolume brick size of 64^3 (from $J_{1,2,3} = 64$, plus borders), thus per-brick core tensor ranks $R = 32$ and core tensors of size R^3. For empty bricks, the node's core tensor is null and skipped.

Feature Scale Parameter As part of the preprocessing, we compute a parameter to measure the feature scale of every octree brick. As seen in Sec. 3.2, the tensor rank truncation can be used for multiscale feature visualization. Specifically, the feature scale parameter is computed on the differences of the approximations and the original at different feature scales (i.e., different rank truncations). Therefore, we compute per brick (excluding borders) the differences in terms of the root-mean-square error (RMSE). With respect to different resolution approximations, we use trilinear interpolation to compute the RMSE between any LOD brick and the original. We compute for every brick a number of different rank truncated reconstructions and store this information in a separate file. Eventually, the visualization system adjusts the feature scale of the multiresolution reconstruction, as shown in Sec. 5.

4.3. Storage

The storage cost of multiresolution volume hierarchies is dominated by the cubic growth of the volume. A simple I^3 volume octree will introduce a hierarchy overhead of $\frac{I^3-1}{7}$ and thus in total require $\frac{8 \cdot I^3-1}{7}$ elements, or $\approx \frac{8}{7} \cdot I^3$. In contrast, our representation consists of a set of global mipmapped factor matrices $\mathbf{U}^{(n)}_{\downarrow k}$ and an octree hierarchy of core tensors $\mathscr{B}_{\text{brick}\downarrow k}$. Below, we present our analysis for 64^3 bricks ($B = 64$) and an initial rank $R_{init} = 32$.

The mipmap hierarchy doubles the storage $I \cdot R_{init}$ of a single initial factor matrix $\mathbf{U}^{(n)}$. For all three matrices, that is $2 \cdot 3 \cdot I \cdot 32 = 192 \cdot I$, thus linear in I. Since each core tensor uses eight times less space than the original data, the total storage cost for the core tensor hierarchy is $\approx \frac{I^3}{7}$. Our meta information is 56 bytes per node and sums up to $56 \cdot \frac{I^3}{B^3} \cdot \frac{8}{7} = \frac{1}{64^2} \cdot I^3 \approx 0.00024 \cdot I^3$ over the entire hierarchy. Therefore, we get a total storage cost of $192 \cdot I + \frac{1}{7} \cdot I^3 + \frac{1}{64^2} \cdot I^3 \approx 192 \cdot I + 0.14 \cdot I^3$. For large I, this is about $\frac{1}{8}$ of the uncompressed volume octree size.

Compared to state-of-the-art [SIGM*11], the storage costs differ mainly in terms of the factor matrix costs. The model in Suter et al. [SIGM*11] has to store three individual factor matrices for each octree node, i.e., $3 \cdot B \cdot R_{init}$. The cumulated matrices of all leave nodes are of size $3 \cdot B \cdot R_{init} \cdot \frac{I^3}{B^3} = \frac{3}{2 \cdot B} \cdot I^3$ for $R_{init} = \frac{1}{2} \cdot B$. Incorporating the general octree overhead, this results in a total factor matrices size of $\frac{8}{7} \cdot \frac{3}{2 \cdot B} \cdot I^3 = \frac{12}{7 \cdot B} \cdot I^3 = \frac{12}{7 \cdot 64} \cdot I^3 = \frac{3}{112} \cdot I^3$. Thus a larger total storage cost of $\frac{3}{112} \cdot I^3 + \frac{1}{7} \cdot I^3 = \frac{19}{112} \cdot I^3 \approx 0.17 \cdot I^3$ results.

Incorporating quantization, the storage costs are affected differently. Both store 8-bit core tensor values (logarithmic encoding). [SIGM*11] use a 16-bit linear factor matrix encoding, while we use non-quantized 32-bit float values for the global factor matrices. Nevertheless, this results in total storage costs for [SIGM*11] of $2 \cdot \frac{3}{112} \cdot I^3 + \frac{1}{7} \cdot I^3 = \frac{11}{56} \cdot I^3 \approx 0.20 \cdot I^3$ and a lower storage costs of $\approx 768 \cdot I + 0.14 \cdot I^3$ for our new model with the global mipmapped factor matrices.

5. Multiscale and Multiresolution Visualization

The goal of the interactive visualization system was to develop a multiresolution volume renderer that selects bricks not only at a certain spatial resolution, but also at a chosen feature scale. We achieved this by modeling the multiscalability within the multiresolution TA octree data structure, as shown in Sec. 4.2. The multiresolution renderer builds upon state-of-the-art view-dependent LOD selection, out-of-core data loading, brick caching on the GPU, asynchronous loading and rendering budgets. The bricks are decoded and reconstructed on demand using consecutive tensor times matrix multiplications, as introduced in [SIGM*11], but using the new global mipmapped factor matrices hierarchy.

In addition to the view-dependent screen-projection based LOD selection, the feature scale of the reconstruction is chosen based on a user input (see Alg. 2). The user input is a feature scale parameter, which maps to a tensor rank. A higher feature scale is achieved by reconstructing more ranks, a lower feature scale is achieved by reconstructing fewer ranks. During the actual visualization, a lower feature scale means that we perform a coarser approximation. This principle is exploited to steer the feature scale via the per-brick RMSE error ranges, as described in Sec. 4.2.

Fig. 11 illustrates the multiscale adjustments to the multiresolution LOD selection given a feature scale, represented by ε_{target} (Alg. 2, line 12). The minimum error front corresponds to $\varepsilon_{target} > \varepsilon_{rank=8}(\mathscr{A}_{\text{brick}\downarrow k})$, which prevents further resolution refinement; in contrast, the maximum error front corresponds to $\varepsilon_{target} > \varepsilon_{rank=32}(\mathscr{A}_{\text{brick}\downarrow k})$, which enforces refinement. During the brick refinement, the rank is updated based on the ε_{target} (Alg. 2, line 16). As mentioned in Alg. 2, line 10, we follow this adjusted multiresolution front as long as we stay within the given rendering and memory budgets.

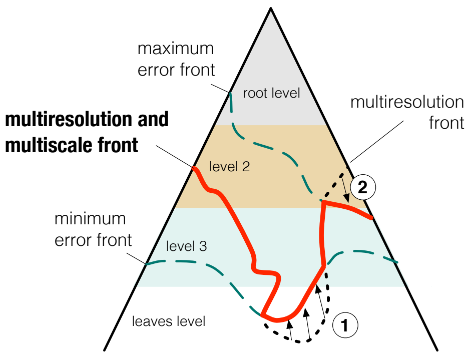


Figure 11: Multiresolution and multiscale octree front (bold): The multiresolution front (dotted) is adjusted depending on (1) the minimum error octree front (prevent refinement), and (2) the maximum error octree front (enforce refinement).

6. Experimental Results

To verify the multiscale and multiresolution TA hierarchy introduced in this paper, we implemented a volume rendering application in C++ based on a GPU-ray-caster using GLSL. Interactive visualization is demonstrated on a Quad-Core Intel i7 3.2GHz with 8GB RAM and a Geforce GTX 580 with 1.5GB memory. Preprocessing has been carried out on an Quad-Core Intel Xeon 2.4GHz MacPro5,1 with 22GB RAM, a 500GB SSD hard disk, and a Geforce GTX 285 graphics card with 1GB memory. The test datasets include three μCT volumes: a *hazelnut* (512^3, $128MB$, $8bit$), a *flower* (1024^3, $1GB$, $8bit$) and a *wood branch* (2048^3, $16GB$, $16bit$) dataset. To avoid excessive type conversions, the input data is preprocessed in floating point precision and the large data tensors (e.g. $32GB$ for wood branch) are accessed from memory mapped files.

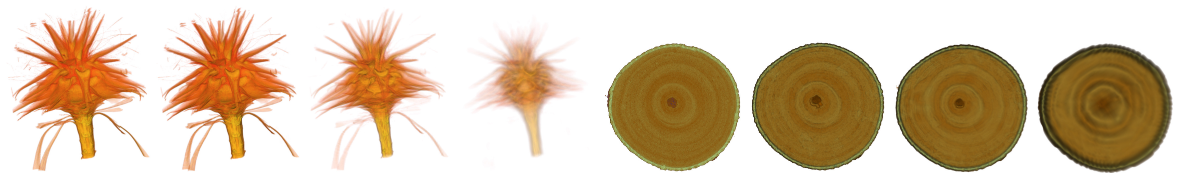
Algorithm 2 Per frame TA-error-based LOD traversal

```
 1: rendering list $L$, loading queue $Q$, heap $H$ (screen-size sorting)
 2: assign a rank corresponding to $\varepsilon_{target}$ to the root brick $\mathscr{A}_{brick\_1}$
 3: if $\mathscr{A}_{brick\_1}$ is on $GPU$ and $\mathscr{A}_{brick\_1}$ is visible then
 4:     push $\mathscr{A}_{brick\_1}$ to $H$ and then push $\mathscr{A}_{brick\_1}$ to $L$
 5: end if
 6: push $\mathscr{A}_{brick\_1}$ to $Q$
 7: while $H$ not empty do
 8:     set current brick $\mathscr{A}_{\text{brick}\downarrow k}$ to the front of $H$
 9:     remove front from $H$
10:     if $(size(Q) \geq budget_{GPU}) \parallel (size(L) \geq budget_{render})$ then
11:         break
12:     else if $(\mathscr{A}_{\text{brick}\downarrow k}$ has no children$) \parallel (\varepsilon_{target} > \varepsilon_{rank=8}(\mathscr{A}_{\text{brick}\downarrow k})) \parallel [(\varepsilon_{target} > \varepsilon_{rank=32}(\mathscr{A}_{\text{brick}\downarrow k}))$ & $(screen\_size(\mathscr{A}_{\text{brick}\downarrow k})) < screen\_size\_threshold)]$ then
13:         continue
14:     end if
15:     set list $C$ to all visible children of $\mathscr{A}_{\text{brick}\downarrow k}$
16:     assign all of $C$ with ranks corresponding to $\varepsilon_{target}$
17:     if all of $C$ are on $GPU$ then
18:         sort $C$ according to the rendering order
19:         find $\mathscr{A}_{\text{brick}\downarrow k}$ in $L$ and replace it with all of $C$
20:         push all of $C$ to $H$
21:     end if
22:     push all of $C$ to $Q$
23: end while
24: update $GPU$ usage statistics based on $Q$ and $L$
25: request missing bricks on $GPU$ from $Q$ to (re)load async
26: render bricks from $L$
```

6.1. Multiresolution Model Using Global TA Bases

Our experiments show that it is feasible, first, to decompose large initial factor matrices of a μCT volume dataset, and second, to reconstruct the volumes at multiple resolutions by mipmapping/subsampling of the large initial factor matrices, see Fig. 12 and Sec. 3. Moreover, we could produce all-orthogonal rank-reducible per-brick core tensors for the multiresolution TA hierarchy from the global TA bases by applying row-block SVDs, as outlined in Sec. 3.3.

To give an idea of how such global TA bases look like, we visualize in Fig. 13 the mipmapped factor matrices of $\mathbf{U}^{(1)}$ of the hazelnut dataset. The intensity distributions look similar to frequency patterns but in fact show the input data specific distribution of the TA's data-specific factor matrix bases. Furthermore, similar to a matrix PCA the first rank is represented by one major base frequency, while the frequencies increase with subsequent ranks, i.e., higher frequency details are encoded with additional ranks.



(a) original (b) leaves (c) level 3 (d) root (e) original (f) level 5 (g) level 3 (h) root

Figure 12: Different spatial resolution levels reconstructed from the mipmapped TA bases: (a-d) flower, (e-h) wood branch.

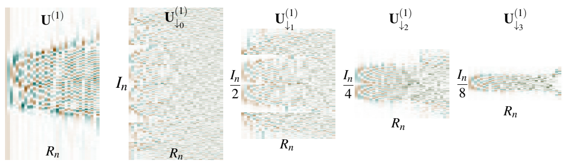
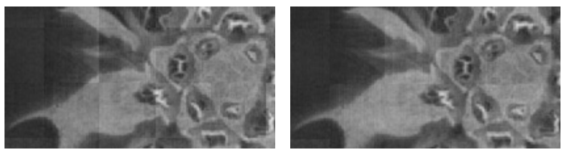


Figure 13: Visualization of an initial factor matrix $\mathbf{U}^{(1)}$ of the hazelnut and its full resolution row-block SVD replacement $\mathbf{U}^{(1)}_{\downarrow 0}$. Subsampled matrices $\mathbf{U}^{(1)}_{\downarrow k}$ are stretched to fit and value coded: brown (negative), white (zero), green (positive).

In order to have smoother brick transitions, we use a 6-voxel-border to generate the core tensors; however, we only reconstruct 2 voxels for the gradient interpolation during rendering (see Fig. 14).



(a) no brick borders (b) with bricks borders

Figure 14: A slice through the reconstructed flower dataset once without (a) and once with additional brick borders (b).

The best approximation of our work, at highest resolution and measured by the normalized RMSE (ε_{approx}), is comparable to state-of-the-art. The presented work and [SIGM*11] have a maximal approximation quality of $\varepsilon_{approx} = 0.00738$ vs. $\varepsilon_{approx} = 0.00676$ (hazelnut), $\varepsilon_{approx} = 0.00968$ vs. $\varepsilon_{approx} = 0.01012$ (flower), and $\varepsilon_{approx} = 0.00702$ vs. $\varepsilon_{approx} = 0.00487$ (wood branch), respectively. The general sampling noise ratio was measured at $\varepsilon_{noise} = 0.00281$ (hazelnut), $\varepsilon_{noise} = 0.00715$ (flower), and $\varepsilon_{noise} = 0.00249$ (wood branch). Thus the maximal approximation quality is close to the sampling noise level.

In addition to [SIGM*11], the multiresolution property of the mipmapped TA factor matrices is coupled with multiscale feature visualization achieved through rank-reduced TA bricks, as shown next.

6.2. Coupling of Multiresolution and Multiscalability

The main idea of this coupling is to have one parameter that a user can adjust in order to balance the scale of features in the dataset (multiscalability). We implemented this idea by computing per brick RMSE errors, whereas we used trilinear interpolation to match the original LOD for comparison. As a result, we use an error parameter that automatically adapts to a feature scale and a LOD resolution from the given error. The average errors per LOD are visualized in Fig. 15, which shows that the error is gradually decreasing when refining the resolution, and it overlaps between octree levels and rank ranges. As the rank-reduced approximations of the Tucker model do not guarantee a strictly decreasing error, we map the minimum-maximum error range to the range of ranks $R_i = \{8, 9, 10, \ldots, 31, 32\}$, which works well in practice.

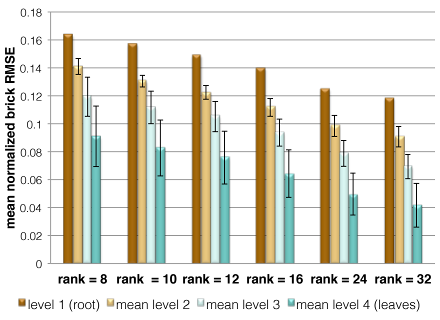


Figure 15: Mean normalized brick RMSEs for all octree levels of the hazelnut. The standard deviation of the errors is additionally indicated.

As can be seen in Fig. 16, when the feature scale parameter is lowered by increasing the target error ε_{target}, the spatial brick resolutions are lowered as well by adjusting the LOD front according to our elaborations in Sec. 5.

6.3. Interactive Performance

As already shown in [SIGM*11], the rendering from TA compressed data is dominated by the ray-caster, see Fig. 17. Our multiresolution and multiscale DVR system shows interactive performance achieving typical volume ray-casting levels. In particular, the timings reveal that our adaptive online tensor reconstruction, including all bricks-to-RAM, bricks-to-GPU and reloading tasks, constitutes only a negligible overhead with respect to the overall rendering cost. The decompression and ray-casting are performed in parallel in separate threads on the GPU. The rendering is performed adaptively according to the zooming factor and the defined error, which was chosen to be a medium error. The bricks are loaded according to the previously defined error-reconstruction-rank coupling.

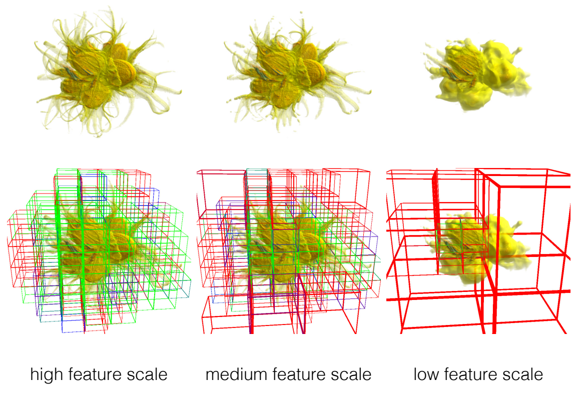


Figure 16: Coupling of multiresolution and multiscalability by a feature scale metric (rank-based). The rank is color encoded (red–blue–green bricks correspond to few-more-many ranks). The size of each brick indicates its spatial resolution.

6.4. Storage and Preprocessing

For all floating point factor matrices and all quantized octree core tensors, the hazelnut dataset ($128MB$) requires $9MB$, the flower dataset ($1GB$, $8bit$) needs $58MB$, and the wood branch dataset ($16GB$, $16bit$) uses $1GB$. The actual compression ratio depends (a) on the voxel-bit-depth of the original, and (b) on the amount of empty space. The flower dataset and the wood branch dataset both have empty surrounding space and are thus below the theoretical storage costs as computed in Sec. 4.3.

The preprocessing time for the initial tensor decomposition in floating point precision to produce the large matrices was around $1min$ for the flower and around $3min$ for the wood branch. The most expensive part in the initial HOSVD decomposition, which was $5sec$ per ALS iteration for the 1025^3 dataset and $42sec$ for the 2048^3 volume. The octree build was around $3min$ for the flower dataset and around $30min$ for the wood branch dataset. The mipmapping and the row-block SVD recomposition of the matrices was below $1sec$ for all datasets. The error computation for the LOD coupling of multiresolution and multiscalability was the most time-consuming part of the preprocessing since a trilinear interpolation was performed for all nodes over several LODs relative to the original data. Note that the preprocessing was performed on the CPU. For the memory critical initial decomposition this is the favored solution. For the octree build and the LOD error computation, however, a parallel and/or GPGPU version could be implemented in future work.

7. Summary

We presented a new concept to couple multiresoltuion DVR and multiscale feature visualization. The idea exploits a novel TA hierarchy for both, the multiresolution modeling and the multiscale feature representation, and is implemented using a state-of-the-art GPU-based ray-caster. The multiresolution and multiscale TA properties are coupled through a feature scale parameter that can be operated at runtime by the user. The feature scale parameter is precomputed per brick over a range of rank-reduced approximations errors (RMSEs). By adjusting the tolerated feature scale parameter, the DVR implementation automatically choses whether to increase/reduce spatial resolutions and feature scales.

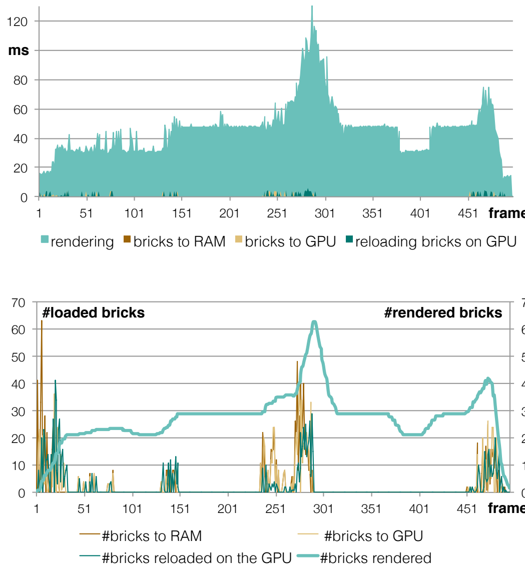


Figure 17: Performance measurements of the flower rendering. Time in ms per frame (top) as well as number of loaded and rendered blocks per frame (bottom).

Our experiments demonstrate the feasibility of the generation of global TA bases and the spatial selectivity within the factor matrices applied to multiresolution data modeling. Furthermore, the coupling of multiresolution DVR and multiscale feature visualization was demonstrated successfully on different datasets and at interactive frame rates. Compared to state-of-the-art [SIGM*11], we maintained a comparable maximum approximation quality, while further reducing the total storage costs for the LOD data structure, and while exploiting one set of global factor matrices that combine multiresolution and multiscale modeling in one.

Acknowledgments

This work was supported in part by a Swiss National Science Foundation (SNSF) grant (project number 200021_132521), as well as by the EU FP7 People Programme (Marie Curie Actions) under REA Grant Agreement n°290227. We would like to acknowledge the Computer-Assisted Paleoanthropology group and the Visualization and MultiMedia Lab at University of Zürich for the acquisition of the μCT datasets, which are available at http://vmml.ifi.uzh.ch. Finally, we thank colorbrewer.org for color schemes used in the figures.

References

[CNLE09] Crassin C., Neyret F., Lefebvre S., Eisemann E.: GigaVoxels: Ray-guided streaming for efficient and detailed voxel rendering. In *Proceedings ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games* (2009), pp. 15–22. 1

[dLdMV00a] de Lathauwer L., de Moor B., Vandewalle J.: A multilinear singular value decomposition. *SIAM Journal of Matrix Analysis and Applications 21*, 4 (2000), 1253–1278. 3, 5

[dLdMV00b] de Lathauwer L., de Moor B., Vandewalle J.: On the best rank-1 and rank-$(R_1, R_2, ..., R_N)$ approximation of higher-order tensors. *SIAM Journal of Matrix Analysis and Applications 21*, 4 (2000), 1324–1342. 5

[EHK*06] Engel K., Hadwiger M., Kniss J. M., Rezk-Salama C., Weiskopf D.: *Real-Time Volume Graphics*. AK Peters, 2006. 1

[FM07] Fout N., Ma K.-L.: Transform coding for hardware-accelerated volume rendering. *IEEE Transaction on Visualization and Computer Graphics 13*, 6 (2007), 1600–1607. 2

[GIM12] Gobbetti E., Iglesias Guitián J., Marton F.: COVRA: A compression-domain output-sensitive volume rendering architecture based on a sparse representation of voxel blocks. *Computer Graphics Forum 31* (2012), 1315–1324. 2

[GMG08] Gobbetti E., Marton F., Guitiàn J. A. I.: A single-pass GPU ray casting framework for interactive out-of-core rendering of massive volumetric datasets. *The Visual Computer 24*, 7-9 (July 2008), 797–806. 1

[GWGS02] Guthe S., Wand M., Gonser J., Strasser W.: Interactive rendering of large volume data sets. In *Proceedings IEEE Visualization* (2002), pp. 53–60. 2

[KB09] Kolda T. G., Bader B. W.: Tensor decompositions and applications. *SIAM Review 51*, 3 (September 2009), 455–500. 1, 2

[LWP*06] Ljung P., Winskog C., Persson A., Lundstrom C., Ynnerman A.: Full body virtual autopsies using a state-of-the-art volume rendering pipeline. *IEEE Transactions on Visualization and Computer Graphics 12*, 5 (Sep/Oct 2006), 869–876. Proceedings IEEE Visualization. 1

[SIGM*11] Suter S. K., Iglesias Guitiàn J. A., Marton F., Agus M., Elsener A., Zollikofer C. P., Gopi M., Gobbetti E., Pajarola R.: Interactive multiscale tensor reconstruction for multiresolution volume visualization. *IEEE Transactions on Visualization and Computer Graphics 17*, 12 (Dec 2011), 2135–2143. 1, 2, 5, 6, 7, 8, 9

[SW03] Schneider J., Westermann R.: Compression domain volume rendering. In *Proceedings IEEE Visualization* (2003), pp. 293–300. 2

[SZP10] Suter S. K., Zollikofer C. P., Pajarola R.: Application of tensor approximation to multiscale volume feature representations. In *Proceedings Vision, Modeling and Visualization* (2010), pp. 203–210. 1, 2, 6

[TS12] Tsai Y.-T., Shih Z.-C.: K-clustered tensor approximation: A sparse multilinear model for real-time rendering. *ACM Transactions on Graphics 31*, 3 (May 2012). 5

[vmm] vmmlib: A vector and matrix math library. https://github.com/VMML/vmmlib/. 5

[WWS*05] Wang H., Wu Q., Shi L., Yu Y., Ahuja N.: Out-of-core tensor approximation of multi-dimensional matrices of visual data. *ACM Transactions on Graphics 24*, 3 (Jul 2005), 527–535. 2, 5

[WXC*08] Wu Q., Xia T., Chen C., Lin H.-Y. S., Wang H., Yu Y.: Hierarchical tensor approximation of multidimensional visual data. *IEEE Transactions on Visualization and Computer Graphics 14*, 1 (Jan/Feb 2008), 186–199. 2, 6