Eurographics 2013 Tutorial: Tensor Approximation in Visualization and Graphics: Background Theory

Susanne K. Suter

Visualization and Multimedia Lab, University of Zurich, Switzerland

Abstract

This compendium on tensor approximation (TA) gives an overview on typical tensor approximation notation and definitions. TA is a tool for data approximation in higher orders. Precisely speaking, TA is an higher-order extension of the matrix singular value decomposition and is a generalization of a data factorization of multidimensional datasets into a set of bases and coefficients. TA consists of two main parts: the tensor decomposition and the tensor reconstruction. In TA, there are several decomposition models available, which are summarized in this document including the main different decomposition algorithms. Furthermore, since low-rank tensor approximations is an interesting tool for data reduction and data factorization, the tensor rank reduction is another important topic. For interactive visualization and graphics applications, the tensor reconstruction is another critical issue since often a fast real-time reconstruction process is required. In this compendium, several reconstruction processes for the different TA models are presented. Finally, some particular TA bases properties that are useful for computer graphics or scientific visualization applications are outlined.

Keywords: *Tensor decompositions, tensor approximations, Tucker model, CANDECOMP/PARAFAC model, compact visual data representation, higher-order SVD methods, data reduction.*

This TA background theory document is a compendium of the Ph.D. thesis of Susanne Suter (see [Sut13]).

1. Introduction

Data approximation is widely used in the fields of computer graphics and scientific visualizations. One way to achieve data approximation is to decompose the data into a more compact and compressed representation. The general idea of a compact data representation is to express a dataset by a set of bases, which are used to reconstruct the dataset to its approximation when needed (see Fig. 1). Precisely speaking, a set of bases usually consists of the actual bases and coefficients describing the relationship between the original data and the actual bases. Typically, such bases sets constitute less data than the original dataset, capture the most significant features, and, moreover, describe the data in a format that is convenient/appropriate for adaptive data loading.

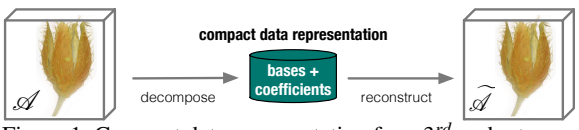


Figure 1: Compact data representation for a 3^{rd}-order tensor $\mathscr{A}$ (a volume) by bases and coefficients that can be used to reconstruct the data to its approximation $\widetilde{\mathscr{A}}$ at run-time.

Bases for compact data representation can be classified into two different types: *pre-defined* and *learned* bases. Pre-defined bases comprise a given function or filter, which is applied to the dataset without any a priori knowledge of the correlations in the dataset. In contrast, learned bases are generated from the dataset itself. Established examples of pre-defined bases are the Fourier transform (FT) and the Wavelet transform (WT). Well-known examples of learned bases are the PCA or the SVD. Using pre-defined bases is often computationally cheaper, while using learned bases requires more computing time (to generate the bases), but potentially removes more redundancy from a dataset.

Generally, PCA-like methods are able to extract the main data direction of the dataset and represent the data in a different coordinate system such that it makes it easier for the user to find the major contributions within the dataset. To exploit this, PCAs higher-order extension – tensor approximation (TA) – can be used for multidimensional datasets.

1.1. Higher-order Data Decompositions

As stated previously, the most common tools for data approximation with learned bases are the matrix SVD and the PCA. Their higher-order extensions are summarized under

© The Eurographics Association 2013.

the term tensor approximation (TA). The first occurrence of TA was in [Hit27]. The idea of multi-way analysis, however, is generally attributed to Catell in 1944 [Cat44]. It took a few decades until tensor approximations received attention, which was by several authors in the field of psychometrics [Tuc66, CC70, Har70].

The matrix SVD/PCA work on 2D matrix data and exploit the fact that the dataset can be represented with a few highly significant coefficients and corresponding reconstruction vectors based on the matrix rank reduction concept. The SVD and the PCA, being multilinear algebra tools compute (a) a rank-R decomposition, and (b) orthonormal row and column vector matrices. The extension to higher-orders is not unique and the two properties from the SVD are captured by two different models that are both given the term tensor approximation: the Tucker model [Tuc66, TBDLK87, dLdMV00a, dLdMV00b, KB09] preserves the orthonormal factor matrices while the CP model (from CANDECOMP [CC70] and PARAFAC [Har70]) preserves the rank-R decomposition.

Generally speaking, a tensor is a term for a higher-order generalization of a vector or a multidimensional array. In TA approaches, a multi-dimensional input dataset in array form, i.e., a tensor, is factorized into a sum of rank-one tensors or into a product of a core tensor (coefficients that describe the relationship to input data) and matrices (bases), i.e., one for each dimension. This factorization process is generally known as *tensor decomposition*, while the reverse process of the decomposition is the *tensor reconstruction*.

Tensor decompositions have been widely studied in other fields and were reviewed [Mor04, KB09, dL09] and summarized [SBG04, Kro08]. Since TA was emerging from different disciplines, it was developed under various names. In particular, the Tucker model is known in the literature under multiple terms. The CP model was independently developed under the terms CANDECOMP and PARAFAC, therefore it is sometimes referenced with a single name. The Tucker model takes its name from Tucker, who initially worked on the *three-mode factor analysis* (3MFA), which is sometimes referred to as the Tucker3 model. [KDL80, TBDLK87, Kro08] called it the *three mode PCA* (3MPCA). Similarly the model was referenced as *N-mode PCA* by [KNW86]. [dLdMV00a] captured all these previous works and wrote down the generalization of the SVD as *multilinear singular SVD*, which is usually termed as higher-order SVD or HOSVD. Thereafter, [VT02, VT04] called it *N-mode SVD*.

Tensor approximation has been used in many areas among which there are applications in the domain of visualization and computer graphics. An overview of theses is given in the next section.

1.2. TA Applications in Graphics and Visualization

TA approaches have been applied to a wide range of application domains. Starting from psychometrics, in recent years, tensor approximation has been applied to visual data. A highly studied area is TA used for image ensembles [SL01, VT02, WA04, HCLH05, SH05, WA05, WA08, YWT*09, MUH11] and/or TA used for pattern recognition, e.g., [SL01, WA05, SE07, SS08, EcGG11, LLWY12]. In (real-time) rendering, tensor decompositions have recently been used as method for global illumination models, e.g., for bidirectional reflectance distribution functions (BRDFs) [SZC*07, BÖK11] or precomputed radiance transfer (PRT) [TS06, SZC*07, TS12]. TA, furthermore, is successfully used for bidirectional texture functions (BTFs) [FKIS02, VT04, WWS*05, WXC*08, RK09, RSK12, TS12], texture synthesis [WXC*08], time-varying visual data [WWS*05, WXC*08], 3D face scanning [VBPP05], compression in animation [Vas02, MK07, PSK*07, WSZP07, KTMW08, MLC10, LXPER11], and multiresolution and multiscale direct volume rendering [SZP10, SIGM*11].

In this compendium, we first give a brief introduction to the singular value decomposition, before we show how the linear algebra notation and definitions are extended to higher-order tensor approximation. Then, the main tensor decomposition models and their low-rank representations, and tensor decomposition algorithms are presented. Subsequently, the alternatives for the inverse process – the tensor reconstruction – are illustrated. Finally, we give hints on particular TA bases properties, which can be used for scientific visualization or computer graphics.

2. Singular Value Decomposition (SVD)

The singular value decomposition (SVD) is a widely used matrix factorization procedure to solve linear least-square problems. The SVD can be applied to any square or rectangular matrix $\mathbf{A} \in \mathbb{R}^{M \times N}$. Hence, the decomposition is always possible. The aim of the SVD is to produce a diagonalization of the input matrix $\mathbf{A}$. Since the input matrix $\mathbf{A}$ is not symmetric, two bases (matrices) are needed to diagonalize $\mathbf{A}$. Therefore, the SVD produces a matrix factorization into two orthogonal bases $\mathbf{U} \in \mathbb{R}^{M \times M}$ and $\mathbf{V} \in \mathbb{R}^{N \times N}$ and a diagonal matrix $\Sigma \in \mathbb{R}^{M \times N}$, as expressed in Eq. (1) (matrix form) or Eq. (2) (summation form).

$$\mathbf{A} = \mathbf{U}\Sigma\mathbf{V}^{-1} = \mathbf{U}\Sigma\mathbf{V}^{\mathbf{T}} \tag{1}$$

$$a_{mn} = \sum_{r=1}^{P} u_{mp}\sigma_p v_{np} \tag{2}$$

The bases $\mathbf{U}$ and $\mathbf{V}$ contain orthogonal unit length vectors u_j and v_j, respectively, and represent a r-dimensional column space ($\mathbb{R}^M$) and a r-dimensional row space ($\mathbb{R}^N$). Hence, the bases $\mathbf{U}$ and $\mathbf{V}$ are even orthonormal, as indicated in Eq. (1), where the inverse of the matrix $\mathbf{V}^{-1}$ equals its transpose $\mathbf{V}^{\mathbf{T}}$. The diagonal matrix Σ contains the *singular values* σ_i, where $\sigma_1 \geq \sigma_2 \geq \ldots \sigma_P \geq 0$, where $P = \min(M,N)$. A *singular value* and a pair of *singular vectors* of a square or rectangular matrix $\mathbf{A}$ are a non-negative scalar

© The Eurographics Association 2013.

σ and two non-zero vectors u_j and v_j so that $\mathbf{A} \cdot v_j = \sigma_j \cdot u_j$ or $\mathbf{A}^T \cdot u_j = \sigma_j \cdot v_j$. The vectors u_j are the *left singular vectors*, and the vectors v_j are the *right singular vectors* (see Fig. 2). The number of non-zero singular values determines the rank R of the matrix $\mathbf{A}$.

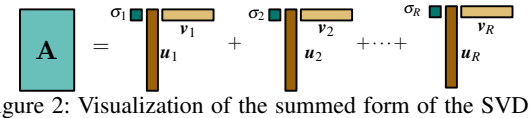


Figure 2: Visualization of the summed form of the SVD as shown in Eq. (2) – illustrating the singular values with the corresponding left and right singular vector pairs.

The SVD can be seen as linear transformation of the orthogonal vectors u_j into the orthogonal vectors v_j, where σ_j is the scaling factor. In other words: singular values are used when the matrix is transformed from one vector space to a different vector space.

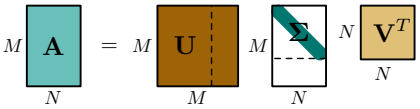
In some applications truncated versions of the SVD are desired. That is, only the first K singular values $\sigma_1 \dots \sigma_K$ and the corresponding K singular vectors $u_1 \dots u_K$ and $v_1 \dots v_K$ are used for the reconstruction. This approach is referred to as low-rank approximation of a truncated SVD.

The singular value decomposition is usually represented in its compact or reduced form (Fig. 3(b)). If we look at the full SVD in Fig. 3(a), we notice that there are only P singular values, where $P = \min(M,N)$, in the diagonal matrix Σ. Therefore, the last columns of $\mathbf{U}$ will be multiplied by zeros. Hence, it is more economic to use the reduced form for computations using the SVD. For so-called low-rank approximations, even smaller decompositions are required known as partial or truncated SVD (Fig. 3(c)) and limiting the number of singular values to $K < P$. In other words, the full SVD has P singular values, the compact/reduced SVD has N singular value and the truncated SVD has K singular values.

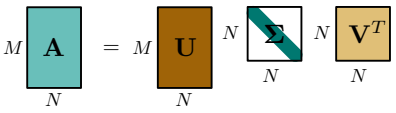
2.1. Computing the SVD

Most frequently, the SVD is computed by using a Householder reduction to a bidiagonal matrix followed by a diagonalization using the QR factorization (for details we refer to [PTVF92, GV96]). However, the SVD can also be computed by using symmetric eigenvalue decomposition. That means, instead of computing the SVD of $\mathbf{A}$, we compute the symmetric eigenvalue decomposition of $\mathbf{A}\mathbf{A}^T$ or $\mathbf{A}^T\mathbf{A}$, which are both symmetric matrices and referred to as covariances matrices of $\mathbf{A}$. In order to find the $u_1 \dots u_m$, we use the symmetric matrix $\mathbf{A}\mathbf{A}^T$ (Eq. (3)); in order to find the $v_1 \dots v_n$, we produce the symmetric matrix $\mathbf{A}^T\mathbf{A}$ and decompose it as in Eq. (4). P is the number of singular values, where $P = \min(M,N)$.

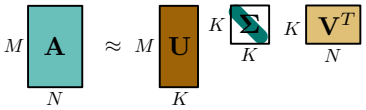
$$\mathbf{A}\mathbf{A}^T = (\mathbf{U}\Sigma\mathbf{V}^\mathbf{T})(\mathbf{U}\Sigma\mathbf{V}^\mathbf{T})^T = \mathbf{U} \begin{bmatrix} \sigma_1^2 & & \\ & \ddots & \\ & & \sigma_P^2 \end{bmatrix} \mathbf{U}^\mathbf{T} \quad (3)$$



(a) full SVD



(b) reduced SVD



(c) truncated SVD

Figure 3: SVD variants: (a) full SVD (P singular values, where $P = \min(M,N)$), (b) reduced/compact SVD (N singular values), and (c) truncated SVD (K singular values).

$$\mathbf{A}^T\mathbf{A} = (\mathbf{U}\Sigma\mathbf{V}^\mathbf{T})^T(\mathbf{U}\Sigma\mathbf{V}^\mathbf{T}) = \mathbf{V} \begin{bmatrix} \sigma_1^2 & & \\ & \ddots & \\ & & \sigma_P^2 \end{bmatrix} \mathbf{V}^\mathbf{T} \quad (4)$$

Note that $\mathbf{U}^T\mathbf{U} = \mathbf{I}$ and $\mathbf{U}^T = \mathbf{U}^{-1}$, $\mathbf{V}^T\mathbf{V} = \mathbf{I}$ and $\mathbf{V}^T = \mathbf{V}^{-1}$. Thus in the example of the matrix $\mathbf{V}$ computation, $\mathbf{V} \begin{bmatrix} \sigma_1^2 & & \\ & \ddots & \\ & & \sigma_P^2 \end{bmatrix} \mathbf{V}^\mathbf{T}$ has the same form as an eigenvalue decomposition of a symmetric matrix (Eq. (5)), where the symmetric matrix is $\mathbf{A}^T\mathbf{A}$. The columns of $\mathbf{V}$ are the eigenvectors of this matrix. The diagonal matrix produces the squares σ^2 of the singular values σ. Note, no matter with which initial symmetric covariance matrix ($\mathbf{A}\mathbf{A}^T$ and $\mathbf{A}^T\mathbf{A}$) we start, the non-zero eigenvalues stay the same.

$$\mathbf{A} = \mathbf{Q}\Lambda\mathbf{Q}^{-1} = \mathbf{Q}\Lambda\mathbf{Q}^T \quad (5)$$

In the following, it is shown how the notation and definitions of the linear algebra concepts are extended from the matrix SVD to higher orders.

3. Tensor Approximation Notation and Definitions

The notation taken here is inspired by that ones of De Lathauwer et al. [dLdMV00a], Smilde et al. [SBG04], and Kolda and Bader [KB09], who follow the notation proposed by Kiers [Kie00]. Other standards have been proposed as well (see [Har01] and [HH02]). To illustrate higher-order extensions we mostly make examples of order three.

3.1. General

A tensor is a multi-dimensional array (or an N-way data array): a 0^{th}-order tensor (tensor0) is a scalar, a 1^{st}-order ten-

© The Eurographics Association 2013.

sor (tensor1) is a vector, a 2^{nd}-order tensor (tensor2) is a matrix, and a 3^{rd}-order tensor (tensor3) is a volume. We consistently use the letter A to represent the data. This follows the notation of, e.g., [dLdMV00a, dLdMV00b, WWS*05, WXC*08, TS12][†]. We use lower case letters for a scalar a, lower case boldfaced letters for a vector $\mathbf{a}$ in $\mathbb{R}^{I_1}$, capital boldfaced letters for a matrix $\mathbf{A}$ in $\mathbb{R}^{I_1 \times I_2}$, and calligraphic letters for a 3^{rd}-order tensor $\mathscr{A}$ in $\mathbb{R}^{I_1 \times I_2 \times I_3}$ (see Fig. 4).

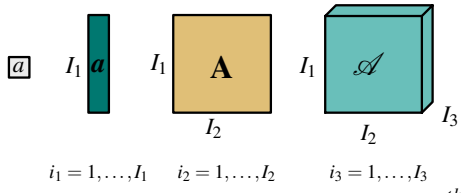
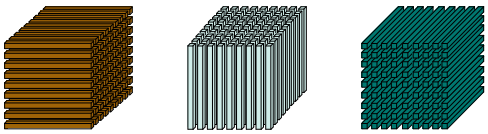


Figure 4: A tensor is a multi-dimensional array: a 0^{th}-order tensor (tensor0) is a scalar a, a 1^{st}-order tensor (tensor1) is a vector a, a 2^{nd}-order tensor (tensor2) is a matrix $\mathbf{A}$, and a 3^{rd}-order tensor (tensor3) is a volume $\mathscr{A}$.

The *order* of a tensor is the number data directions, also referred as *ways* or *modes*. Along a mode j, the index i_j runs from 1 to I_J. By using lower script indices for the modes, we can extend the index scheme to any order, i.e., $I_1, I_2, I_3, I_4, \ldots$. The i^{th} entry of a vector a is denoted by a_i, an element (i_1, i_2) of a matrix $\mathbf{A}$ is denoted by $a_{i_1 i_2}$, and an element (i_1, i_2, i_3) of a 3^{rd}-order tensor $\mathscr{A}$ is denoted by $a_{i_1 i_2 i_3}$.

The general term *fibers* is used as a generalization for vectors taken along different modes in a tensor (see Fig. 5). A fiber is defined by fixing every index but one. A matrix column is a mode-1 fiber and a matrix row is a mode-2 fiber. 3^{rd}-order tensors have column, row, and tube fibers, denoted by a_{i_1}, a_{i_2}, and a_{i_3}, respectively. Sometimes, fibers are called mode-n vectors.



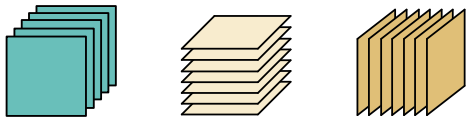
(a) rows a_{i_1} (b) columns a_{i_2} (c) tubes a_{i_3}

Figure 5: Fibers of a tensor3 $\mathscr{A}$.

Slices are two-dimensional sections of a tensor (e.g., one fixed index in a tensor3). For a 3^{rd}-order tensor $\mathscr{A}$, there are, for example, frontal, horizontal, and lateral slices, denoted by $\mathbf{A}_{i_1}, \mathbf{A}_{i_2}$, and $\mathbf{A}_{i_3}$, respectively, (see Fig. 6).

For computations, a tensor is often reorganized into a matrix what we denote as *tensor unfolding* (sometimes called *matricization*). There are two main unfolding strategies,

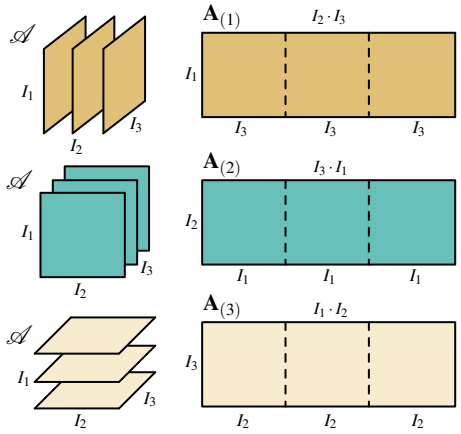
[†] In other areas, however, as for example in statistics, it is common to use the letter X for the data [Kie00, KB09].



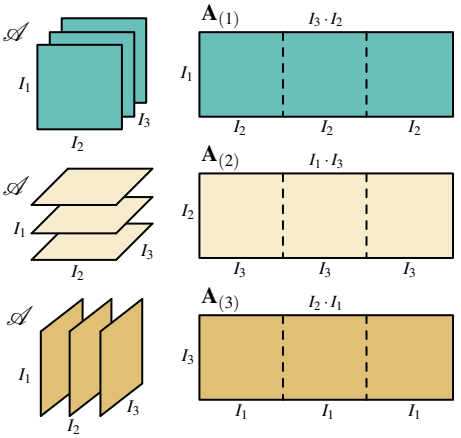
(a) frontal slices $\mathbf{A}_{i_3}$ (b) horizontal slices $\mathbf{A}_{i_1}$ (c) lateral slices $\mathbf{A}_{i_2}$

Figure 6: Slices of a tensor3 $\mathscr{A}$.

backward cyclic unfolding [dLdMV00a] and *forward cyclic unfolding* [Kie00] (see Fig. 7). An unfolded tensor in matrix shape is denoted with a subscript in parentheses, e.g., $\mathbf{A}_{(n)}$.



(a) backward cyclic unfolding [dLdMV00a]



(b) forward cyclic unfolding [Kie00]

Figure 7: Backward vs. frontal unfolding of a tensor3.

3.2. Computing with Tensors

Here, the most common products used while computing with tensors are outlined. The notation taken here is mostly taken from [KB09] and follows the notations proposed by Kiers [Kie00]. Some notations are, however, taken from [dL09] and [SBG04].

© The Eurographics Association 2013.

- An N^{th}-order tensor is defined as $\mathscr{A} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$.
- The *tensor product* is denoted here by $\otimes$: however, other symbols are used in the literature, too. For rank-one tensors, the tensor product corresponds to the *vector outer product* ($\circ$) of N vectors $b^{(n)} \in \mathbb{R}^{I_n}$ and results in an N^{th}-order tensor $\mathscr{A}$. The tensor product or vector outer product for a 3^{rd}-order rank-one tensor is illustrated in Fig. 8: $\mathscr{A} = b^{(1)} \circ b^{(2)} \circ b^{(3)}$, where an element (i_1, i_2, i_3) of $\mathscr{A}$ is $a_{i_1 i_2 i_3} = b^{(1)}_{i_1} b^{(2)}_{i_1} b^{(3)}_{i_3}$.

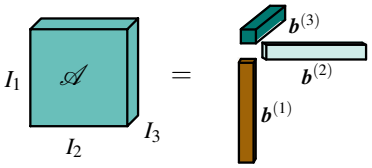


Figure 8: Three-way outer product for a rank-one tensor3 $\mathscr{A} = b^{(1)} \circ b^{(2)} \circ b^{(3)}$.

- The *inner product* of two same-sized tensors $\mathscr{A}, \mathscr{B} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ is the sum of the products of their entries, i.e., Eq. (6).

$$\langle \mathscr{A}, \mathscr{B} \rangle = \sum_{i_1=1}^{I_1} \sum_{i_2=1}^{I_2} \cdots \sum_{i_N=1}^{I_N} a_{i_1,i_2,\ldots i_N} b_{i_1,i_2,\ldots i_N} \tag{6}$$

- The *n-mode product* [dLdMV00a] multiplies a tensor by a matrix (or vector) in mode n. The n-mode product of a tensor $\mathscr{B} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ with a matrix $\mathbf{C} \in \mathbb{R}^{J_n \times I_n}$ is denoted by $\mathscr{B} \times_n \mathbf{C}$ and is of size $I_1 \times \cdots \times I_{n-1} \times J_n \times I_{n+1} \times \cdots \times I_N$. That is, element-wise we have Eq. (7).

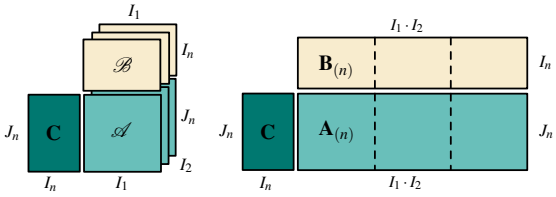
$$(\mathscr{B} \times_n \mathbf{C})_{i_1 \ldots ß_{n-1} j_n i_{n+1} \ldots i_N} = \sum_{i_n=1}^{I_n} b_{i_1 i_2 \ldots i_N} \cdot c_{j_n i_n} \tag{7}$$

  Each mode-n fiber is multiplied by the matrix $\mathbf{C}$. The idea can also be expressed in terms of unfolded tensors (reorganization of tensor into a matrix; see Sec. 3.1).

$$\mathscr{A} = \mathscr{B} \times_n \mathbf{C} \Leftrightarrow \mathbf{A}_{(n)} = \mathbf{C}\mathbf{B}_{(n)} \tag{8}$$

  The n-mode product of a tensor with a matrix is related to a change of basis in the case when a tensor defines a multilinear operator [KB09]. The n-mode product is the generalized operand to compute tensor times matrix (TTM) multiplications, as illustrated in Fig. 17.
- The *Hadamard product* ($*$) is the element-wise product between two matrices $\mathbf{A} \in \mathbb{R}^{I \times J}$ and $\mathbf{B} \in \mathbb{R}^{I \times J}$ of the same size (see Eq. (9)).

$$\mathbf{A} * \mathbf{B} = \begin{bmatrix} a_{11}b_{11} & \ldots & a_{1J}b_{1J} \\ \vdots & \ddots & \vdots \\ a_{I1}b_{I1} & \ldots & a_{IJ}b_{IJ} \end{bmatrix} \tag{9}$$



(a) TTM of a tensor3 (b) TTM of an unfolded tensor3

Figure 9: Tensor times matrix (TTM) multiplication.

- The *Kronecker product* ($\otimes$) multiplies two matrices $\mathbf{A} \in \mathbb{R}^{I \times J}$ and $\mathbf{B} \in \mathbb{R}^{K \times M}$ block-wise as in Eq. (10), while the resulting matrix $\mathbf{A} \otimes \mathbf{B}$ is of size $(IK \times JM)$. The Kronecker product ($\otimes$) is denoted by the same operator as the outer product and is a generalization of the vector outer product to matrices. The Kronecker product is in fact a special case of the tensor product, but not every tensor product is a Kronecker product [Bur95].

$$\mathbf{A} \otimes \mathbf{B} = \begin{bmatrix} a_{11}\mathbf{B} & \ldots & a_{1J}\mathbf{B} \\ \vdots & \ddots & \vdots \\ a_{I1}\mathbf{B} & \ldots & a_{IJ}\mathbf{B} \end{bmatrix} \tag{10}$$

- The *Khatri-Rao product* ($\odot$) [SBG04] is denoted as a column-wise Kronecker product. The resulting matrix $\mathbf{A} \odot \mathbf{B}$ is of size $(IJ) \times K$ for the two matrices $\mathbf{A} \in \mathbb{R}^{I \times K}$ and $\mathbf{B} \in \mathbb{R}^{J \times K}$ (see Eq. (11)).

$$\mathbf{A} \odot \mathbf{B} = \begin{bmatrix} a_1 \otimes b_1 & a_2 \otimes b_2 & \ldots & a_K \otimes b_K \end{bmatrix} \tag{11}$$

  Note: If a and b are vectors, then the Khatri-Rao and Kronecker products are identical, i.e., $a \otimes b = a \odot b$.
- The *Moore-Penrose inverse* [Moo20, Pen55] is a generalized matrix pseudo inverse $\mathbf{A}^+ \in \mathbb{R}^{I \times J}$, which works for rectangular matrices $\mathbf{A} \in \mathbb{R}^{I \times J}$. There are other matrix pseudo inverses; however, here the robust and SVD-based Moor-Penrose inverse is used: $\mathbf{A}^+ = \mathbf{U}\Sigma^+\mathbf{V}^T$, where Σ^+ represents the pseudo inverse of Σ as in Eq. (1) of the SVD.
- The *norm of a tensor* $\mathscr{A} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ is defined analogously to the matrix Frobenius norm $\|\mathbf{A}\|_F$ and is the square root of the sum squares of all its elements, i.e., Eq. (12).

$$\|\mathscr{A}\|_F = \sqrt{\sum_{i_1=1}^{I_1} \sum_{i_2=1}^{I_2} \cdots \sum_{i_N=1}^{I_N} a^2_{i_1,i_2,\ldots i_N}} \tag{12}$$

3.3. Rank of a Tensor

In order to describe the definitions of the tensor rank, the definition for the matrix rank is recaptured. The *matrix rank* of a matrix $\mathbf{A}$ is defined over its column and row ranks, i.e., the column and row *matrix rank* of a matrix $\mathbf{A}$ is the maximal number of linearly independent columns and rows of

© The Eurographics Association 2013.

A, respectively. For matrices, the column rank and the row rank are always equal and, a matrix rank is therefore simply denoted as $rank(\mathbf{A})$. A *tensor rank* is defined similarly to the matrix rank. However, there are differences. In fact, the extension of the rank concept is not uniquely defined in higher-orders. The definitions for the tensor ranks are taken from [dLdMV00a].

- The *n-rank* of a tensor $\mathscr{A}$, denoted by $R_n = rank_n(\mathscr{A})$, is the dimension of the vector space spanned by mode-n vectors, where the mode-n vectors of $\mathscr{A}$ are the column vectors of the unfolding $\mathbf{A}_{(n)}$, and $rank_n(\mathscr{A}) = rank(\mathbf{A}_{(n)})$. Unlike matrices, the n-ranks of a tensor are not necessarily the same.
- A higher-order tensor has a *multilinear rank* $(R_1, R_2, \ldots, R_N)$ [Hit27] if its mode-1 rank (row vectors), mode-2 rank (column vectors) until its mode-N rank are equal to $R_1, R_2, \ldots, R_N$, e.g., a multilinear rank-(R_1, R_2, R_3) for a 3^{rd}-order tensor.
- A *rank-one tensor* is an N-way tensor $\mathscr{A} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ under the condition that it can be expressed as the outer product of N vectors, as in Eq. (13) (see also [Kru89, CM96]). A rank-one tensor is also known under the term *Kruskal tensor*.

$$\mathscr{A} = b^{(1)} \circ b^{(2)} \circ \cdots \circ b^{(N)} \tag{13}$$

- The *tensor rank* $R = rank(\mathscr{A})$ is the minimal number of rank-one tensors that yield $\mathscr{A}$ in a linear combination (see [Kru89, CM96, dLdMV00a, KB09]). Except for the special case of matrices, the tensor rank is not necessarily equal to any of its n-ranks. It always holds that $R_n \leq R$.

Now that the basic notation and definitions with tensor approximation algebra is given, the possible factorizations into tensor decompositions models are summarized next.

4. Tensor Decompositions

In general, in tensor decompositions an input tensor $\mathscr{A} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ is decomposed into a set of factor matrices $\mathbf{U}^{(n)} \in \mathbb{R}^{I_n \times R_n}$ and coefficients that describe the relationship/interactivity between $\mathscr{A}$ and the set of $\mathbf{U}^{(n)}$.

Historically, as seen earlier, tensor decompositions are a higher-order extension of the matrix SVD/PCA. The nice properties of the matrix SVD, i.e., rank-R decomposition and orthonormal row-space vectors and column-space vectors, do not extend uniquely to higher orders. The rank-R decomposition can be achieved with the so-called CP model, while the orthonormal row and column vectors are preserved in the so-called Tucker model. An extensive review of the two models and further hybrid models can be found in [KB09]. Here, we outline the two most common models, the Tucker model and the CP model. Hybrid models are mentioned only briefly.

4.1. Tucker Model

The Tucker model is a widely used approach for tensor decompositions. As given in Eq (14), any higher-order tensor is approximated by a product of a core tensor $\mathscr{B} \in \mathbb{R}^{R_1 \times R_2 \times \cdots \times R_N}$ and its factor matrices $\mathbf{U}^{(\mathbf{n})} \in \mathbb{R}^{I_n \times R_n}$, where the products $\times_n$ denote the n-mode product (see Sec. 3.2) between the tensor and the matrices. This decomposition can later be reconstructed to its approximation $\widetilde{\mathscr{A}}$. The missing information of the input tensor $\mathscr{A}$ that cannot be captured by $\widetilde{\mathscr{A}}$ is denoted with the error ε. The Tucker decomposition is visualized for a 3^{rd}-order tensor in Fig. 10. Alternatively, a Tucker decomposition can be expressed as a sum of rank-one tensors (Eq. (15) and Fig. 11).

$$\mathscr{A} = \mathscr{B} \times_1 \mathbf{U}^{(1)} \times_2 \mathbf{U}^{(2)} \times_3 \cdots \times_N \mathbf{U}^{(N)} + e \tag{14}$$

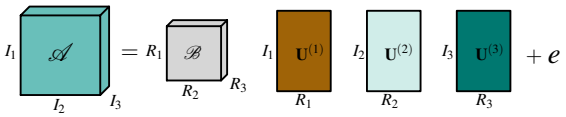


Figure 10: Tucker tensor3: $\mathscr{A} = \mathscr{B} \times_1 \mathbf{U}^{(1)} \times_2 \mathbf{U}^{(2)} \times_3 \mathbf{U}^{(3)}$ + e.

$$\mathscr{A} = \sum_{r_1=1}^{R_1} \sum_{r_2=1}^{R_2} \cdots \sum_{r_N=1}^{R_N} b_{r_1 r_2 \ldots r_N} \cdot u_{r_1}^{(1)} \circ u_{r_2}^{(2)} \circ \cdots \circ u_{r_N}^{(N)} + e \tag{15}$$

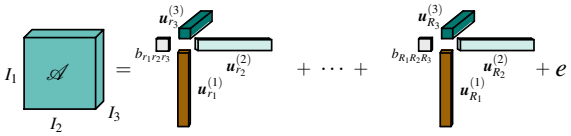


Figure 11: Tucker tensor3 as a sum of rank-one tensors: $\mathscr{A} = \sum_{r_1=1}^{R_1} \sum_{r_2=1}^{R_2} \sum_{r_3=1}^{R_3} b_{r_1 r_2 r_3} \cdot u_{r_1}^{(1)} \circ u_{r_2}^{(2)} \circ u_{r_3}^{(3)} + e$.

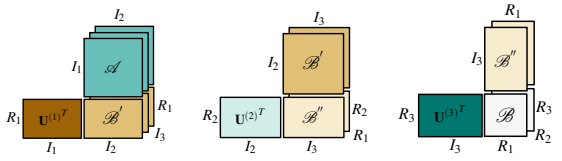
The column vectors of the factor matrices $\mathbf{U}^{(n)} \in \mathbb{R}^{I_n \times R_n}$ are usually orthonormal and can be thought of as principal components R_n in each mode n [KB09]. The core tensor $\mathscr{B} \in \mathbb{R}^{R_1 \times R_2 \times \cdots \times R_N}$ represents a projection of the original data $\mathscr{A} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ onto its factor matrices and is always of the same order as the input data. The core tensor is computed in general, as shown in Eq. (16), and for orthogonal factor matrices as in Eq. (17) (see Fig. 12). The element-wise core tensor computation is denoted in Eq. (18). In other words, the core tensor coefficients $b_{r_1 r_2 \ldots r_N}$ show the relationship or interactivity between the Tucker model and the original data.

$$\mathscr{B} = \mathscr{A} \times_1 \mathbf{U}^{(1)^{(-1)}} \times_2 \mathbf{U}^{(2)^{(-1)}} \times_3 \cdots \times_N \mathbf{U}^{(N)^{(-1)}} \tag{16}$$

$$\mathscr{B} = \mathscr{A} \times_1 \mathbf{U}^{(1)^T} \times_2 \mathbf{U}^{(2)^T} \times_3 \cdots \times_N \mathbf{U}^{(N)^T} \tag{17}$$

© The Eurographics Association 2013.

$$\mathscr{B} = \sum_{i_1=1}^{I_1} \sum_{i_2=1}^{I_2} \cdots \sum_{i_N=1}^{I_N} a_{i_1 i_2 \ldots i_N} \cdot u_{i_1}^{(1)^T} \circ u_{i_2}^{(2)^T} \circ \cdots \circ u_{i_N}^{(N)^T} \tag{18}$$



(a) mode 1 TTM (b) mode 2 TTM (c) mode 3 TTM

Figure 12: Forward cyclic tensor times matrix (TTM) computation after [Kie00] in order to produce the core tensor $\mathscr{B}$: n-mode products along the three modes.

The Tucker decomposition is not unique, which means that we can modify the core tensor $\mathscr{B}$ without affecting the model fit as long as we apply the same changes to the factor matrices (so-called core tensor rotations). This provides the option to rearrange the core tensor such that, for example, more values are zero. For details see [KB09].

4.2. CP Model

The parallel factor analysis (PARAFAC) or the canonical decomposition (CANDECOMP), called CP in short, factorizes a tensor into a sum of R rank-one tensors. Hence, a tensor $\mathscr{A} \in \mathbb{R}^{I_1 \times I_2 \cdots \times I_N}$ can be rank decomposed as a sum of R rank-one tensors as in Eq. (19). An example of a 3^{rd}-order CP decomposition is illustrated in Fig. 13. Note: The column vectors of the matrices in Eq. (19) are normalized, which yields a weighting factor λ_r for each term. The information not captured by the CP model is represented with the error ε.

$$\mathscr{A} = \sum_{r=1}^{R} \lambda_r \cdot u_r^{(1)} \circ u_r^{(2)} \circ \cdots \circ u_r^{(N)} + e \tag{19}$$

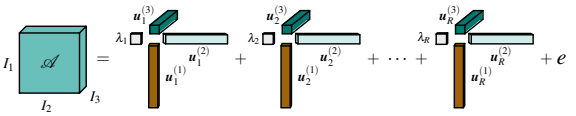


Figure 13: CP tensor3, sum of rank-one tensors: $\mathscr{A} = \sum_{r=1}^{R} \lambda_r \cdot u_r^{(1)} \circ u_r^{(2)} \circ u_r^{(3)} + e$.

The CP model is in fact a special case of the Tucker model. The vector containing the λ-values can be arranged as the super-diagonal of a Tucker core tensor with R diagonal values, while the rest of the core tensor is zero (see Fig. 14). In contrast to the Tucker model, the CP model is unique under certain constraints (see [KB09]). In this context, uniqueness means that the current CP model is the only possible combination of rank-one tensors that sums to $\widetilde{\mathscr{A}}$. However, permutation freedom and scaling is still possible.

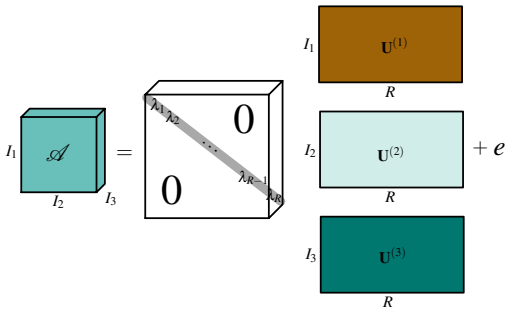


Figure 14: CP tensor3 visualized as a Tucker tensor3.

4.3. Other Models

There are a number of other models, mostly some hybrid forms of the CP model and the Tucker model. One such model is the so-called *block-diagonal tensor decomposition* by [DL08a, DL08b, dLN08], which produces a super-diagonal of P core tensor with zeros except for the blocks forming the diagonal, as illustrated in Fig. 15. Other hybrid models can be found in the extensive review by [KB09].

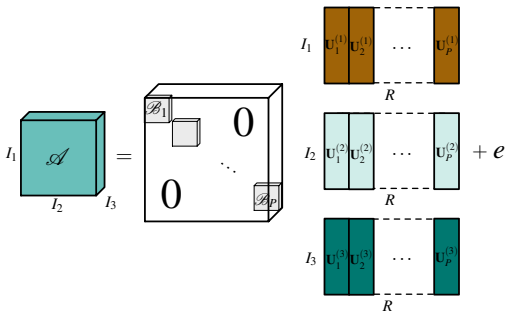


Figure 15: Block-diagonal tensor3.

Often, we are interested in compact models, which enable a compression of the input dataset. For example, after computing a Tucker decomposition the core tensor $\mathscr{B}$ has the same size as the original input dataset $\mathscr{A}$ and all the factor matrices are quadratic. However, we are more interested in light-weight, approximative Tucker decompositions, where $\mathscr{B}$ is an element of $\mathbb{R}^{R_1 \times R_2 \times R_3}$ with $R_1 < I_1$, $R_2 < I_2$ and $R_3 < I_3$. Using so-called *rank-reduced tensor decompositions* or *truncated tensor decompositions* one can directly obtain more compact decompositions. Furthermore, the rank-reduced decompositions are usually better in terms of the difference between approximated and original data [KB09]. In the next section, the tensor rank approximations corresponding to the Tucker model and the CP model are defined.

5. Tensor Rank Reduction

As seen in Sec. 3.3, the extension of the matrix rank concept to higher orders is not unique. There are two main directions followed, which are based on either a rank-one, i.e., a

© The Eurographics Association 2013.

rank-R tensor decomposition or a rank-$(R_1, R_2, \ldots, R_N)$ tensor decomposition. Their rank-reduced approximations are defined accordingly:

i) A rank-one approximation is defined as $\widetilde{\mathscr{A}} = \lambda \cdot u^{(1)} \circ u^{(2)} \cdots \circ u^{(N)}$ from the rank-one tensor (vector) product ($\circ$) of its basis vectors $u^{(n)} \in \mathbb{R}^{I_n}$ and the scalar λ. Hence a tensor $\mathscr{A}$ can be approximated by a linear combination of rank-one approximations as in Eq. (20). This approximation, previously defined as a CP model, and is called a *rank-R approximation*.

$$\widetilde{\mathscr{A}} \approx \sum_{r=1}^{R} \lambda_r \cdot u_r^{(1)} \circ u_r^{(2)} \circ \cdots \circ u_r^{(N)} \tag{20}$$

ii) Alternatively, a *rank-*$(R_1, R_2, \ldots R_N)$ *approximation* of $\mathscr{A}$ is formulated as a decomposition into a lower-rank tensor $\widetilde{\mathscr{A}} \in \mathbb{R}^{I_1 \times I_2 \cdots \times I_N}$ with $rank_n(\widetilde{\mathscr{A}}) = R_n \leq rank_n(\mathscr{A})$. The approximated tensor is the n-mode product $\times_n$ of factor matrices $\mathbf{U}^{(n)} \in \mathbb{R}^{I_n \times R_n}$ and a core tensor $\mathscr{B} \in \mathbb{R}^{R_1 \times R_2 \cdots \times R_N}$ in a given reduced rank space (Eq. (21)). This rank-$(R_1, R_2, \ldots R_N)$ approximation was previously introduced as the *Tucker* model.

$$\widetilde{\mathscr{A}} \approx \mathscr{B} \times_1 \mathbf{U}^{(1)} \times_2 \mathbf{U}^{(2)} \times_3 \cdots \times_N \mathbf{U}^{(N)} \tag{21}$$

In general a rank-reduced approximation is sought such that the least-squares cost function in Eq. (22) is minimized.

$$\widetilde{\mathscr{A}} = \arg\min(\widetilde{\mathscr{A}}) \left\| \mathscr{A} - \widetilde{\mathscr{A}} \right\|^2 \tag{22}$$

The notation of the different rank-approximations becomes interesting for compression approaches. Given that $(R_1, R_2, \ldots R_N)$ or R are sufficiently smaller than the initial lengths $(I_1, I_2, \ldots, I_N)$, the coefficients $\Lambda \in \mathbb{R}^R$ or $\mathscr{B} \in \mathbb{R}^{R_1 \times R_2 \times \cdots \times R_N}$ and the factor matrices $\mathbf{U}^{(n)} \in \mathbb{R}^{I_n \times R_n}$ lead to a compact approximation of $\widetilde{\mathscr{A}}$ of the original tensor $\mathscr{A}$. In particular, the multilinear rank-$(R_1, R_2, \ldots R_N)$ is typically explicitly chosen to be smaller than the initial ranks in order to achieve a compression on the input data. In contrast, the CP model often needs larger factor matrices, where often $R_n \gg I_n$ is necessary to represent the dataset (see Fig. 14).

5.1. Choosing Principal Components

In tensor approximation, we would like to make use of selecting major components from the decomposition, as similarly known from the matrix PCA. That is, by eliminating the higher-ranked principal components and their basis vectors, we preserve the most important directions/structures in the dataset. In other words, we reconstruct the major components of the original datasets, but details are missing. These details can be added by progressively reconstructing more and more principal components to the approximated form of the original dataset. In practice, many of the insignificant principal components or their basis vectors are very low or close to zero, i.e., they are negligible. Typically, the first couple of principal components already define most of the total variability within a dataset. For data approximation techniques, we therefore often use only a certain number of principal components and their basis vectors to represent a dataset, i.e., we work with a reduced set of singular values σs and truncated factor matrices (see Sec. 2). Correspondingly, a rank-reduced or truncated tensor decomposition is desired.

5.2. Truncated Tensor Decompositions

Similar as to matrix PCA, tensor rank reduction can be used to generate lower-rank reconstructions $\widetilde{\mathscr{A}}$ of $\mathscr{A}$. The tensor rank parameter R_n is responsible for the number of TA coefficients and bases that are used for the reconstruction and hence is responsible for the approximation level. In higher orders, the CP decomposition produced from an alternating least squares (ALS) algorithm (see Sec. 6), is not rank-reducible per se. The ex post truncation of the Tucker decomposition, however, is possible due to the *all-orthogonality* property of the core tensor.

For a 3^{rd}-order tensor, all-orthogonality means that the different horizontal matrices of the core $\mathscr{B}$ (the first index i_1 is kept fixed, while the two other indices, i_2 and i_3, are free) are mutually orthogonal with respect to the scalar product of matrices (i.e., the sum of the products of the corresponding entries vanishes). The same holds for fixed indices i_2 and i_3 (see [dLdMV00a]). Therefore, given an initial rank-(R_1, R_2, R_3) Tucker model, we can progressively choose lower ranks $K_n < R_n$ for reduced quality reconstruction. As indicated in Fig. 16 on the example of the Tucker model, the rank indicates how many factor matrix columns and corresponding core tensor entries are used for the reconstruction. From that, we conclude that there are basically two ways to go: (1) either start with the desired rank reduction as initially described or (2) subsequently or progressively truncate the given decomposition.

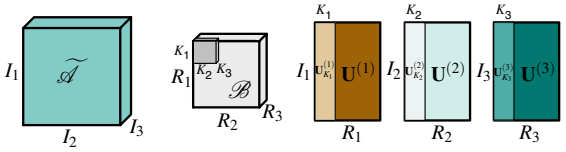


Figure 16: Illustration of a rank reduced Tucker tensor reconstruction: A reduced range of factor matrix columns with corresponding fewer core tensor entries reconstructs a lower quality approximation but at full resolution.

As in the matrix PCA case, a small R_n corresponds to a low-rank Tucker tensor approximation (many details removed) and a large R_n corresponds to an approximation matching the original more closely. The ordering of the coefficients in the core tensor is not strictly decreasing as in the matrix SVD case the singular values are; however, in practice, it can be shown that progressive tensor rank reduction

© The Eurographics Association 2013.

in the Tucker model works well for adaptive visualization of the data at different feature scales.

The algorithms to compute such rank-reducible tensor decompositions are summarized in the next section.

6. Tensor Decomposition Algorithms

There are a couple of different strategies for how to perform tensor decompositions or rank approximations. The most popular and widely uses group of algorithms belongs to the *alternating least squares* (ALS) algorithms. The other group of algorithms used various Newton methods. The respective algorithms differ also for the computation of the CP model and the Tucker model.

For the Tucker model, the first decomposition algorithms used were a simple higher-order SVD (HOSVD) (see [dLdMV00a]), the so-called *Tucker1* [Tuc66], a the three-mode SVD. However, the truncated decompositions of higher orders are not optimal in terms of best fit, which is measured by the Frobenius norm of the difference. Starting from an Tucker1 (or HOSVD) algorithm, tensor approximation ALS algorithms [KDL80, Kro83] were developed, where one of the first Tucker ALS was the so-called *TUCKALS* [TBDLK87]. Later various optimizations accelerated [AB98] or optimized the basic TUCK-ALS. The *higher-order orthogonal iteration* (HOOI) algorithm [dLdMV00b] is an iterative algorithm to perform a better fit for a truncated HOSVD version.

Newton methods are used for the Tucker decomposition or rank-$(R_1, R_2, \ldots, R_N)$ approximation as well. They typically start with an HOOI initialization and then converge faster to the final point. [ES09] developed a *Newton-Grassman optimization* approach, which takes much fewer iterations than the HOOI - even though one single iteration is more expensive due to the computation of the Hessian. While the HOOI is not guaranteed to converge, the Newton-Grassmann Tucker decomposition is guaranteed to converge to a stationary point. Another Newton method was proposed by [IDLAVH09], who developed a *differential-geometric Newton* algorithm with a fast quadratic convergence of the algorithm in a neighborhood of the solution. Since this method is not guaranteed to converge to a global maximum, they support the method by starting with an initial guess of several HOOI iterations, which increases the chances of converging to a solution.

For the CP model, one question addressed is how to find the number of rank-one tensors: CORCONDIA [BK03] is an algorithm that performs a consistency diagnostic to compare different numbers of components. For a fixed number of components, there is a CP ALS algorithm, which was presented in the two original CP articles [CC70, Har70]. [ZG01] proposed to use *incremental rank-one fitting procedures*, which first fit the original tensor by a rank-one tensor, then subtract the rank-one approximation from the original and fit the residue with another rank-one tensor until a certain given number of F incremental rank-one approximations have been performed. They propose a *Jacobi Gauss-Newton* (JGN) iteration to execute the incremental rank-one approximations.

In the following, the basic HOSVD algorithm and the widely used ALS algorithms to produce the Tucker model and the CP model are explained.

6.1. HOSVD Algorithm

The *HOSVD* or *multilinear SVD* [dLdMV00a], which is a higher-order generalization of the SVD, is a basic algorithm that is used to compute the different tensor decomposition models. The idea of the HOSVD is to compute a matrix SVD along every mode of the input tensor $\mathscr{A} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$. To achieve this, the tensor $\mathscr{A}$ is unfolded along every mode n to its matrix representation $\mathbf{A}_{(n)}$, as shown in Fig. 7. Then a matrix SVD is computed on the unfolded matrix $\mathbf{A}_{(n)}$. The R_n leading left singular vectors are chosen as the basis $\mathbf{U}^{(n)} \in \mathbb{R}^{I_n \times R_n}$ for the mode n. As shown in Alg. 1, this procedure is repeated for every mode n.

Algorithm 1 HOSVD along every mode n.

1: **for** every mode n of N **do**
2: unfold $\mathscr{A} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ into its matrix representation $\mathbf{A}_{(n)} \in \mathbb{R}^{I_n \times (I_1 \cdots I_{n-1} \cdot I_{n+1} \cdots I_N)}$
3: compute the matrix SVD $\mathbf{A}_{(n)} = \mathbf{U}^{(n)} \mathbf{\Sigma} \mathbf{V}^{(n)T}$
4: set the R_n leading left singular vectors to the mode-n basis $\mathbf{U}^{(n)} \in \mathbb{R}^{I_n \times R_n}$
5: **end for**

6.2. ALS Algorithms

Alternating least-squares algorithms are used to find parameters of a model, which corresponds to an optimization problem. In particular, if no closed-form solutions to problems are available, iterative algorithms that gradually improve the estimates and converge to the optimum solution are used. The tensor ALS produces a tensor decomposition consisting of N basis matrices $\mathbf{U}^{(1 \ldots N)}$ and coefficients representing the relationship between the input tensor and the basis matrices (see Sec. 4). The general idea with the multiway/tensor ALS algorithms is to fix all basis matrices but one and optimize only for $\mathbf{U}^{(n)}$. By fixing all bases but one, the optimization problem is reduced to a linear least squares problem. This procedure is repeated for every mode-n basis. One iteration step comprises the optimization of all bases individually. The improvement of the solution is measured after each iteration by a predefined *set of convergence/stopping criteria*, which decides if the current fit is considered to be the best fit.

Often it is difficult to define the stopping criteria [Kro08]. In order to have a termination of the algorithm, a maximum number of iterations should be set since ALS algorithms typically suffer from converging neither to a global maximum nor a stationary point. It is, however, possible that we only arrive at a local maximum instead of a global one, e.g., by performing many small steps. Likewise, the definition of

© The Eurographics Association 2013.

some restrictions on the step size should be considered (e.g., larger steps at the beginning, smaller ones towards the end). Generally, it can be said that the more structure there is in the dataset, the greater the chance that a global optimum can be reached. This means that one of the ALS convergence criteria is the maximum number iterations allowed. The number of iterations needed also depends on the goodness of the initial starting point, called the *initial guess*. Common solutions are to start either with a *random initial guess* or with the *HOSVD initial guess*. Nevertheless, we could end up with *multiple solutions* by choosing different initial guesses. Another typical convergence criterion is the so-called *fit*, which basically computes the differences of the least-squares cost function Eq. (22). That is, in tensor approximation, the norm of the tensor decomposition is compared to the norm of the original data. If this difference changes, i.e., the improvement of the fit from the last step is below a certain threshold, the ALS algorithm exits after the current iteration.

In the following, we describe the two ALS algorithms selected by [dLdMV00b]: HOOI for the Tucker ALS and HOPM for the CP ALS. However, we would like to mention that many other authors have come up with variants of the ALS algorithms, and these can be looked up in [KB09].

6.3. Tucker ALS

Given an N^{th}-order tensor $\mathscr{A} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ the optimization problem to be solved for $\widetilde{\mathscr{A}} \stackrel{\text{def}}{=} \mathscr{B} \times_1 \mathbf{U}^{(1)} \times_2 \cdots \times_N \mathbf{U}^{(N)}$ is the minimization of the least-squares cost function Eq. (22). This problem can be turned into a maximization problem (Eq. (23)) in order to get a maximized basis matrix $\mathbf{U}^{(n)}$ along mode n (details see [AB98, dLdMV00b, KB09]). The maximization problem is implemented in the HOOI ALS as described in Alg. 2. Both, the formula and the algorithm are given for orthogonal basis matrices, where $\mathbf{U}^{(n)} \in \mathbb{R}^{I_n \times R_n}$. In the case of non-orthogonal basis matrices, each matrix transpose has to be replaced by a matrix inverse.

$$\max_{\mathbf{U}^{(n)}} \left\| \mathscr{A} \times_1 {\mathbf{U}^{(1)}}^T \times_2 {\mathbf{U}^{(2)}}^T \cdots \times_N {\mathbf{U}^{(N)}}^T \right\| \qquad (23)$$

In fact, [dLdMV00b] show that we can substitute the minimization problem for orthonormal bases such that Eq. (24) holds, which is used in line 13 in Alg. 2.

$$\arg\min(\widetilde{\mathscr{A}}) = \|\mathscr{A}\|^2 - \|\mathscr{B}\|^2 \qquad (24)$$

6.4. CP ALS

For a CP-ALS, the least-squares problem to be solved can be described as follows [dLdMV00b]: Given a real N^{th}-order tensor $\mathscr{A} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$, find a scalar λ and unit-norm vectors $\mathbf{U}^{(1)}, \mathbf{U}^{(2)}, \ldots, \mathbf{U}^{(N)}$ such that the rank-one tensor $\widetilde{\mathscr{A}} \stackrel{\text{def}}{=} \lambda \cdot \mathbf{U}^{(1)} \circ \mathbf{U}^{(2)} \circ \cdots \circ \mathbf{U}^{(N)}$ minimizes the least-squares cost function Eq. (22) over the manifold of rank-one tensors. In other words, a rank-one approximation is defined as minimization of the distance between the given tensor and its approximation on the rank-one manifold. In [dLdMV00b] they show that this is equivalent to the maximization of the norm of the projection of the original tensor onto the rank-one manifold. The actual computation of the CP tensor approximation is performed by the HOPM ALS, as described in Alg. 3. Note: The norm of the $\left\|\widetilde{\mathscr{A}}\right\|_F$ (Alg. 3, line 12) can be approximated by computing $\|\mathscr{A}\|_F^2$, adding the squared norm of the Kruskal tensor, subtracting twice the inner product of the Kruskal tensor, and taking the square root of it (see [KB09, BK*12]). This approach saves significant computing time.

Algorithm 2 The higher-order orthogonal iteration: HOOI($\mathscr{A}, R_1, R_2, \ldots, R_n$).

1: init basis matrices $\mathbf{U}^{(\mathbf{n})}$ (random, HOSVD)
2: compute max norm $\|\mathscr{A}\|_F$
3: set fit change tolerance: $1.0e-4$
4: set max number of iterations: typically, we use 10
5: **while** fit change greater than tolerance AND max number of iterations not reached **do**
6: fitold = fit
7: **for** mode $n = 1, 2, 3, \ldots, N$ **do**
8: optimize mode n: $\mathscr{P} \leftarrow \mathscr{A} \times_1 {\mathbf{U}^{(1)}}^T \cdots \times_{n-1} {\mathbf{U}^{(n-1)}}^T \times_{n+1} {\mathbf{U}^{(n+1)}}^T \cdots \times_N {\mathbf{U}^{(N)}}^T$
9: compute new basis matrix: $\mathbf{U}^{(\mathbf{n})} \leftarrow HOSVD(\mathbf{P}_{(n)})$
10: **end for**
11: compute core: $\mathscr{B} = \mathscr{P} \times_N \mathbf{U}^{(N)}$
12: compute Frobenius norm on current core tensor: $\|\mathscr{B}\|_F$
13: compute norm residual: $\|\mathscr{A}_\delta\|_F = \sqrt{\|\mathscr{A}\|_F^2 - \|\mathscr{B}\|_F^2}$
14: compute fit: $1 - \frac{\|\mathscr{A}_\delta\|_F}{\|\mathscr{A}\|_F}$
15: compute fit change: $|fitold - fit|$
16: **end while**

Algorithm 3 The higher-order power method: HOPM($\mathscr{A}, R$).

1: init basis matrices $\mathbf{U}^{(\mathbf{n})}$ (random, HOSVD)
2: compute max norm $\|\mathscr{A}\|_F$
3: set fit change tolerance: $1.0e-4$
4: set max number of iterations: typically, we use $50-100$
5: **while** fit change greater than tolerance AND max number of iterations not reached **do**
6: fitold = fit
7: **for** mode $n = 1, 2, 3, \ldots, N$ **do**
8: optimize mode n: $\mathbf{V} \leftarrow {\mathbf{U}^{(1)}}^T \mathbf{U}^{(1)} * \cdots * {\mathbf{U}^{(n-1)}}^T \mathbf{U}^{(n-1)} * {\mathbf{U}^{(n+1)}}^T \mathbf{U}^{(n+1)} * \cdots * {\mathbf{U}^{(N)}}^T \mathbf{U}^{(N)}$
9: compute new basis matrix: $\mathbf{U}^{(n)} \leftarrow \mathbf{A}_{(n)} (\mathbf{U}^{(N)} \odot \cdots \odot \mathbf{U}^{(n+1)} \odot \mathbf{U}^{(n-1)} \odot \cdots \odot \mathbf{U}^{(1)}) \mathbf{V}^+$
10: normalize new $\mathbf{U}^{(n)}$ (norm becomes λ)
11: **end for**
12: compute norm residual: $\|\mathscr{A}_\delta\|_F = \left\|\widetilde{\mathscr{A}}\right\|_F$
13: compute fit: $1 - \frac{\|\mathscr{A}_\delta\|_F}{\|\mathscr{A}\|_F}$
14: compute fit change: $|fitold - fit|$
15: **end while**
16: sort decomposition

© The Eurographics Association 2013.

The described HOOI ALS and HOPM ALS produce tensor approximations for either a given rank R or a given multilinear rank $(R_1, R_2, \ldots R_N)$, respectively. In particular, for the Tucker model rank-reduced approximations are often desired in order to compress the amount of data, while in the CP model the number of chosen ranks is an important factor as well, as described in the previous section. In fact, we can distinguish, for the CP decomposition, between algorithms that directly compute a rank-R decomposition and algorithms that incrementally compute rank-one decompositions [ZG01]. In the latter approach some computationally expensive steps can be skipped; however, the reconstruction step used for the incremental approach is expensive as well.

After having introduced the concepts for (rank-reduced) tensor decompositions, we look at the possible reconstruction approaches. In fact, it is critical to choose the appropriate reconstruction approach in order to achieve a real-time reconstruction for interactive visualization.

7. Tensor Reconstruction

The *tensor reconstruction* of a reduced-rank tensor decomposition can be achieved in different ways. One alternative is a progressive reconstruction: Each entry in the (superdiagonal) core tensor $\mathscr{B}$ is considered as weight for the outer product between the corresponding column vectors in the factor matrices. This looks like Eq. (25) for the Tucker reconstruction and Eq. (20) for the CP reconstruction.

$$\widetilde{\mathscr{A}} \approx \sum_{r_1=1}^{R_1} \sum_{r_2=1}^{R_2} \cdots \sum_{r_N=1}^{R_N} b_{r_1 r_2 \ldots r_N} \cdot u_{r_1}^{(1)} \circ u_{r_2}^{(2)} \circ \cdots \circ u_{r_N}^{(N)} \quad (25)$$

This reconstruction strategy corresponds to reconstructing rank-one tensors and cumulatively summing them up. The weighted "subtensors" then form the approximation $\widetilde{\mathscr{A}}$ of the original data $\mathscr{A}$. In particular for the Tucker model, this is an expensive reconstruction strategy since it involves multiple for-loops to run over all the summations, which typically slows down the computing time.

7.1. Element-wise Reconstruction

Another approach, is to reconstruct each element of the approximated dataset individually, which we call *element-wise reconstruction* approach. Each element $\widetilde{a}_{i_1 i_2 i_3}$ is reconstructed, as shown in Eq. (26) for the Tucker reconstruction, and as shown in Eq. (27) for the CP reconstruction. For the Tucker model that is: All core coefficients multiplied with the corresponding coefficients in the factor matrices are summed up (weighted sum). Similarly, this applies for the CP model expect that we have only diagonal core coefficients or λ's as they are usually called.

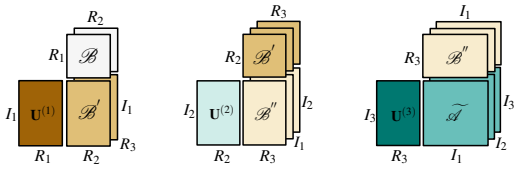
$$\widetilde{a}_{i_1 i_2 \ldots i_N} \approx \sum_{r_1 r_2 \ldots r_N} b_{r_1 r_2 \ldots r_N} \cdot u_{i_1 r_1}^{(1)} \cdot u_{i_2 r_2}^{(2)} \cdot \cdots \cdot u_{i_N r_N}^{(N)} \quad (26)$$

$$\widetilde{a}_{i_1 i_2 \ldots i_N} \approx \sum_{r=1}^{R} \lambda_r \cdot u_{i_1 r}^{(1)} \cdot u_{i_2 r}^{(2)} \cdot \cdots \cdot u_{i_N r}^{(N)} \quad (27)$$

The element-wise reconstruction can be beneficial for applications where only a sparse amount of reconstructed elements are needed.

7.2. Optimized Tucker Reconstruction

A third reconstruction approach – applying only to the Tucker reconstruction – is to build the n-mode products along every mode, which leads to a tensor times matrix (TTM) multiplication for each mode, e.g., TTM1 along mode 1, (see Eq. (8)). This is analogous to the Tucker model given by Eq. (21). In Fig. 17 we visualize the TTM reconstruction applied to a 3^{rd}-order tensor using n-mode products.



(a) TTM 1 (b) TTM 2 (c) TTM 3

Figure 17: Forward cyclic TTM multiplications after [Kie00] along the three modes (n-mode products).

Given the fixed cost of generating a $I_1 \times I_2 \times I_3$ grid, the computational overhead factor varies from cubic rank complexity $R_1 \cdot R_2 \cdot R_3$ in the case of the progressive reconstruction (Eq. (25)) to a linear rank complexity R_1 for the TTM or the n-mode product reconstruction (Eq. (26)). For example, for $R = 16$, the improvement to $R^3 = 4'096$ is 256-fold.

7.3. CP Reconstruction by the Khatri-Rao Product

For completeness, an alternative CP reconstruction strategy is mentioned as well. That is, the CP reconstruction can be computed with the Khatri-Rao product $\odot$ (see Sec. 3.2), as in the example of a 3^{rd}-order tensor in Eq. (28).

$$\widetilde{\mathbf{A}}_n \approx \mathbf{U}^{(1)} \left(\mathbf{U}^{(2)} \odot \mathbf{U}^{(3)}\right)^T \quad (28)$$

However, it is to be noted that this reconstruction strategy produces large matrices (see Sec. 3) due to the Khatri-Rao product, which extends the matrix rows and the matrix columns to a multiple between the rows and the columns of two matrices, respectively. Obviously, that results in more expensive matrix-matrix multiplications.

After the general notation and concepts of TA has been introduced, we outline next some ways to exploit specific TA bases properties for scientific visualization applications.

8. Useful TA Properties for Scientific Visualization

As stated in the introduction, TA is the higher-order generalization of the matrix SVD, which exhibits the nice properties of (a) rank-R decomposition and (b) orthonormal row-space and column-space vectors. In higher orders, the orthonormal row and column vectors are preserved in the Tucker model.

© The Eurographics Association 2013.

The higher-order TA bases properties can be exploited in order to steer different aspects of visualization.

The Tucker model (Sec. 4.1) consists of one factor matrix per mode (data direction) $\mathbf{U}^{(\mathbf{n})} \in \mathbb{R}^{I_n \times R_n}$ and one core tensor $\mathscr{B} \in \mathbb{R}^{R_1 \times R_2 \ldots R_N}$. The core tensor $\mathscr{B}$ is in effect a projection of the original data $\mathscr{A}$ onto the basis of the factor matrices $\mathbf{U}^{(\mathbf{n})}$. In case of a volume, the Tucker model has three modes, as illustrated in Fig. 10, and defines an approximation $\widetilde{\mathscr{A}} = \mathscr{B} \times_1 \mathbf{U}^{(1)} \times_2 \mathbf{U}^{(2)} \times_3 \mathbf{U}^{(3)}$ of the original volume $\mathscr{A}$ (using n-mode products $\times_n$).

The row and column axes of the factor matrices represent two different spaces: (1) the rows correspond to the spatial dimension in the corresponding mode, and (2) the columns to the approximation quality. Next, we show how these properties can be exploited for multiresolution modeling (spatial selection and subsampling of rows) and multiscale approximation (rank reduction on the columns) (see Fig. 18).

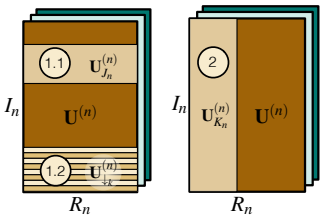


Figure 18: Factor matrix properties along the vertical axis: (1.1) spatial selectivity, (1.2) spatial subsampling, and (2) low-rank approximation.

The next two subsections not only elaborate the spatial properties of the TA matrices, but also illustrate how these properties can be exploited in multiresolution DVR, as a sample application in scientific visualization.

8.1. 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 given fixed resolution and explain the multiresolution access in the following section.

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

$$\widetilde{a}_{i_1 i_2 i_3} = \sum_{r_1} \sum_{r_2} \sum_{r_3} b_{r_1 r_2 r_3} \cdot u^{(1)}_{i_1 r_1} \cdot u^{(2)}_{i_2 r_2} \cdot u^{(3)}_{i_3 r_3}, \qquad (29)$$

with factor matrix and core tensor entries $u^{(n)}_{i_n r}$ 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. 18), we can define row-index subranges $J_n \subseteq [0 \ldots I_n]$ that reconstruct a well defined spatial subvolume $J_1 \times J_2 \times J_3$ for the reduced index ranges $i_n \in J_n$ in Eq. 29. As illustrated in Fig. 19, we can thus select and reconstruct a subvolume of the dataset 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

$$\widetilde{\mathscr{A}}_{J_1 \times J_2 \times J_3} = \mathscr{B} \times_1 \mathbf{U}^{(1)}_{J_1} \times_2 \mathbf{U}^{(2)}_{J_2} \times_3 \mathbf{U}^{(3)}_{J_3}. \qquad (30)$$

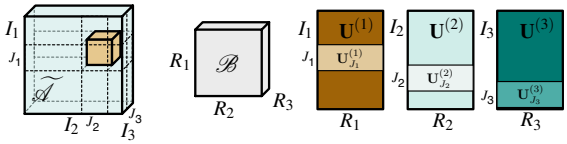


Figure 19: Illustration of spatial selectivity within TA bases: A range of selected submatrix rows reconstructs a well defined subvolume (in brown) of the original whole dataset.

In Fig. 23 an example of two selected subvolumes (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.

8.2. Spatial Subsampling

For multiresolution volume visualization, we need lower resolution subsampled and averaged representations of subvolume bricks for view-dependent adaptive level-or-detail rendering. Due to the direct spatial correspondence of factor-matrix rows to the spatial dimensions as outlined above, we can apply the lower-resolution subsampling on factor matrices before brick reconstruction from the TA representation.

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 first merging and 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 capture the data variation along that dimension. Therefore, downsampling and averaging pairs of rows corresponds to halving the reconstructed volume resolution. This downsampling of factor matrices is indicated in Fig. 20 and corresponds to what is known as *mipmapping*.

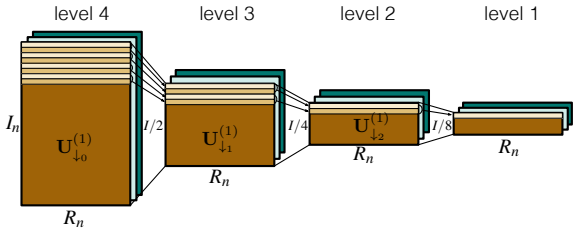


Figure 20: Mipmapping of the global factor matrices: Subsampling by averaging (example input data is of size 512^3 and the bricks are of size 64^3).

© The Eurographics Association 2013.

Fig. 24 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 the 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.

8.3. Orthogonality Properties Affected by Spatial Selectivity or Spatial Subsampling

The described selective usage of the factor matrices, i.e., the projection of bricks onto submatrices, changes the energy distribution within the core tensor. In the classical Tucker decomposition, we end up with a core tensor having one extremely high value (hottest core value) and many small values (that roughly follow a logarithmic distribution). In particular, the distribution of the core coefficient changes with the usage of selective factor matrices. Furthermore, we lose the all-orthogonality property [dLdMV00a] within the core tensor since the subselections or subsamplings of the TA factor matrices lose their orthogonality (in fact orthonormality). In fact, the all-orthogonality property in the core tensor makes it possible to truncate ranks of the tensor decomposition (similar to matrix rank reduction within an SVD). Non-orthogonal TA factor matrices hence produce non-rank-reducible core tensors. Therefore, it is necessary to develop a strategy to produce all-orthogonal core tensors for multiresolution modeling directly on the TA bases.

To solve the issue of maintaining rank-reducible core tensors, the subspace of each row-block factor submatrix $\mathbf{U}^{(n)}_{J_n}$ can be re-spanned by applying an SVD and replacing its columns with the singular vectors (similar to [TS12, Sut13]). The TA matrices are then replaced and recomposed from the re-spanned submatrices along each mode (see Fig. 21). In that way, we are able to define rank-reducible per-brick core tensors. Intuitively, this recomposition of the matrices can be seen as a different representation of the same local subspaces as those defined by the initial non-orthogonal submatrices. J_n corresponds to the size of the subvolumes.

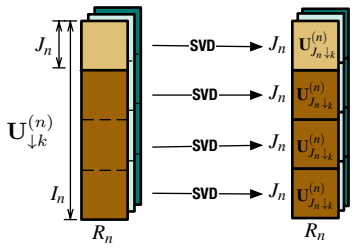


Figure 21: Postprocessing the mipmapped initial factor matrices in order to obtain orthogonal localized row-block submatrices and thus all-orthogonal core tensors.

To give an idea what such updated TA bases look like, we visualize in Fig. 22 the factor matrices of $\mathbf{U}^{(1)}$ of the scientific visualization dataset (hazelnut). 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.

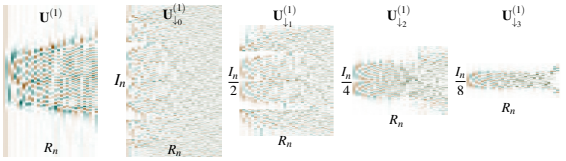


Figure 22: Visualization of an initial factor matrix $\mathbf{U}^{(1)}$ of the hazelnut and its full resolution row-block SVD replacement $\mathbf{U}^{(1)}_{\downarrow 0}$. Subsampled matrices $\mathbf{U}^{(1)}_{\downarrow k}$ are stretched to fit and value coded: brown (negative), white (zero), green (positive).

Until now, the modifications that can be applied to the spatial axis of the factor matrices were explained; however, modifications along the vertical axis can be applied as well and result in changes regarding the approximation level.

8.4. Approximation and Rank Reduction

The Tucker model defines a rank-(R_1, R_2, R_3) approximation, where a small R_n corresponds to a low-rank approximation (with details removed) and a large R_n corresponds to a more accurate approximation of the original. The highest rank R_n for the initial Tucker decomposition has to be given explicitly. However, rank reductions can be applied after the initial decomposition (similar to the rank reduction in matrix SVD). Even tough the core tensor coefficients are not guaranteed to be in decreasing order, as in matrix SVD, in practice it can be shown that progressive tensor rank reduction in the Tucker model works well for adaptive visualization of the data at different feature scales.

The tensor rank R_n defines the number of coefficients and bases used for the reconstruction, and hence is responsible for the approximation level. As illustrated in Fig. 16, the rank indicates how many factor matrix columns and corresponding core tensor entries are used for a reconstruction. Thus, given a rank-(R_1, R_2, R_3) Tucker model, we can progressively choose lower ranks $K_n < R_n$ for reduced quality reconstruction at the same spatial resolution dictated by I_n.

Fig. 25 compares the progressive rank reduction 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 reduced-rank representations are visually similar down to the lowest ranks, which, however, are hardly used. Fig. 25 indicate, moreover, that low-rank tensor approximation can be used for multiscale feature visualization or progressive image refinement.

Acknowledgments

This work was supported in parts by the Forschungskredit of the University of Zürich and a Swiss National Science Foun-

© The Eurographics Association 2013.

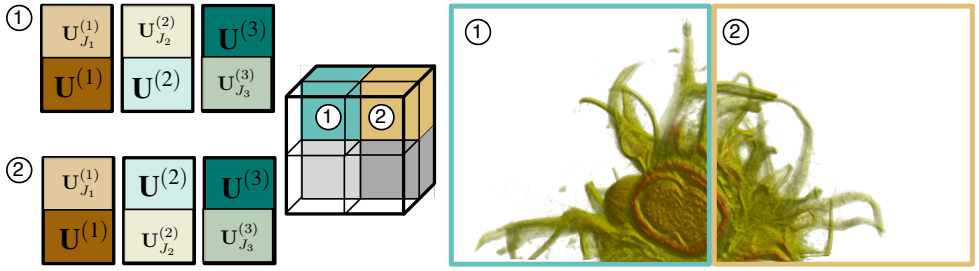


Figure 23: Spatial selectivity of factor matrices. Two selected bricks are reconstructed by the corresponding selection of row index subranges.

Figure 24: Factor matrix subsampling (bottom row) compared to direct TA (middle row) derived from original subsampled input datasets (top row).

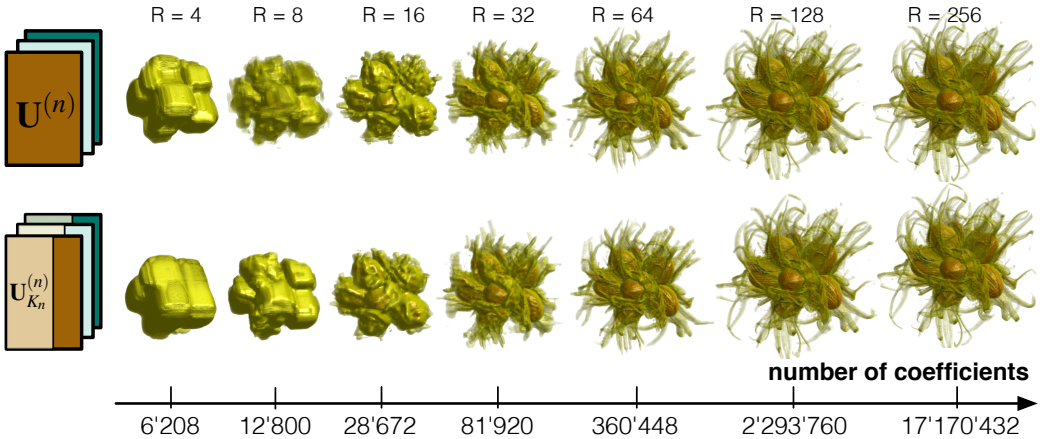


Figure 25: Multiscale volume visualization by tensor rank reduction (bottom row) compared to rank-(R,R,R) tensor approximations to specific ranks R (top row).

© The Eurographics Association 2013.

dation (SNSF) grant (project n°200021_132521). Furthermore, we would like to acknowledge the Computer-Assisted Paleoanthropology group and the Visualization and Multi-Media Lab at University of Zürich for the acquisition of the test datasets.

References

[AB98] ANDERSSON C. A., BRO R.: Improving the speed of multi-way algorithms: Part i. Tucker3. *Chemometrics and Intelligent Laboratory Systems 42* (1998), 93–103. 9, 10

[BK03] BRO R., KIERS H. A.: A new efficient method for determining the number of components in PARAFAC models. *Journal of Chemometrics 16* (2003), 387–400. 9

[BK*12] BADER B. W., KOLDA T. G., ET AL.: MATLAB tensor toolbox version 2.5. Available online, January 2012. 10

[BÖK11] BILGILI A., ÖZTÜRK A., KURT M.: A general BRDF representation based on tensor decomposition. *Computer Graphics Forum 30*, 8 (December 2011), 2427–2439. 2

[Bur95] BURDICK D. S.: An introduction to tensor products with applications to multiway data analysis. *Chemometrics and Intelligent Laboratory Systems 28* (1995), 229–237. 5

[Cat44] CATTELL R. B.: Parallel proportional profiles and other principles for determining the choice of factors by rotation. *Psychometrika 9*, 4 (December 1944), 267–283. 2

[CC70] CAROLL J. D., CHANG J.-J.: Analysis of individual differences in multidimensional scaling via an n-way generalization of "Eckart–Young" decompositions. *Psychometrika 35* (1970), 283–319. 2, 9

[CM96] COMON P., MOURRAIN B.: Decomposition of quantics in sums of powers of linear forms. *Signal Processing, Special Issue on Higher Order Statistics 53* (1996), 93–108. 6

[DL08a] DE LATHAUWER L.: Decompositions of a higher-order tensor in block terms — part i: Lemmas for partitioned matrices. *SIAM Journal on Matrix Analysis and Applications 30*, 3 (2008), 1022–1032. 7

[DL08b] DE LATHAUWER L.: Decompositions of a higher-order tensor in block terms – Part II: Definitions and uniqueness. *SIAM Journal on Matrix Analysis and Applications 30*, 3 (2008), 1033–1066. 7

[dL09] DE LATHAUWER L.: A survey of tensor methods. In *Proceedings IEEE International Symposium on Circuits and Systems* (2009), pp. 2773–2776. 2, 4

[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. 2, 3, 4, 5, 6, 8, 9, 13

[dLdMV00b] DE LATHAUWER L., DE MOOR B., VANDEWALLE J.: On the best rank-1 and rank-$(R_1, R_2, \ldots, R_N)$ approximation of higher-order tensors. *SIAM Journal of Matrix Analysis and Applications 21*, 4 (2000), 1324–1342. 2, 4, 9, 10

[dLN08] DE LATHAUWER L., NION D.: Decompositions of a higher-order tensor in block terms – Part III: Alternating least squares algorithms. *SIAM Journal of Matrix Analysis and Applications 30*, 3 (2008), 1067–1083. 7

[EcGG11] ERGIN S., ÇAKIR S., GEREK O. N., GÜLMEZOĞLU M. B.: A new implementation of common matrix approach using third-order tensors for face recognition. *Expert Systems with Applications* (2011), 3246–3251. 2

[ES09] ELDEN L., SAVAS B.: A Newton–Grassmann method for computing the best multilinear rank-(r_1, r_2, r_3) approximation of a tensor. *SIAM Journal on Matrix Analysis and Applications 31*, 2 (2009), 248–271. 9

[FKIS02] FURUKAWA R., KAWASAKI H., IKEUCHI K., SAKAUCHI M.: Appearance based object modeling using texture database: acquisition, compression and rendering. In *Proceedings Eurographics Workshop on Rendering* (2002), pp. 257–266. 2

[GV96] GOLUB G. H., VAN LOAN C. F.: *Matrix Computations*. The John Hopkins University Press, 1996. 3

[Har70] HARSHMAN R. A.: Foundations of the PARAFAC procedure: Models and conditions for an "explanatory" multi–modal factor analysis. *UCLA Working Papers in Phonetics 16* (1970), 1–84. 2, 9

[Har01] HARSHMAN R. A.: An index formulism that generalized the capabilites of matrix notation and algebra to n-way arrays. *Journal of Chemometrics 15* (2001), 689–714. 3

[HCLH05] HE X., CAI D., LIU H., HAN J.: Image clustering with tensor representation. In *Proceedings ACM Multimedia Conference* (2005), pp. 132–140. 2

[HH02] HARSHMAN R. A., HONG S.: 'strech' vs. 'slice' methods for representing three-way structure via matrix notation. *Journal of Chemometrics 16* (2002), 198–205. 3

[Hit27] HITCHCOCK F. L.: The expression of a tensor or a polyadic as a sum of products. *Journal of Mathematics and Physics 6* (1927), 164–169. 2, 6

[IDLAVH09] ISHTEVA M., DE LATHAUWER L., ABSIL P.-A., VAN HUFFEL S.: Differential-geometric newton method for the best rank-(R_1, R_2, R_3) approximation of tensors. *Numerical Algorithms 51*, 2 (June 2009), 179–194. 9

[KB09] KOLDA T. G., BADER B. W.: Tensor decompositions and applications. *SIAM Review 51*, 3 (September 2009), 455–500. 2, 3, 4, 5, 6, 7, 10

[KDL80] KROONENBERG P. M., DE LEEUW J.: Principal component analysis of three-mode data by means of alternating least squares algorithms. *Psychometrika 45* (1980), 69–97. 2, 9

[Kie00] KIERS H. A.: Towards a standardized notation and terminology in multiway analysis. *Journal of Chemometrics 14*, 3 (May/June 2000), 105–122. 3, 4, 7, 11

[KNW86] KAPTEYN A., NEUDECKER H., WANSBEEK T.: An approach to n-mode components analysis. *Psychometrika 51*, 2 (1986), 269–275. 2

[Kro83] KROONENBERG P. M.: *Three-mode Principal Component Analysis: Theory and Applications*. Leiden: DSWO Press, 1983. 9

[Kro08] KROONENBERG P. M.: *Applied Multiway Data Analysis*. Wiley, 2008. 2, 9

[Kru89] KRUSKAL J. B.: *Rank, Decomposition, and Uniqueness for 3-way and N-way Arrays*. 1989, pp. 7–18. 6

[KTMW08] KRÜGER B., TAUTGES J., MÜLLER M., WEBER A.: Multi-mode tensor representation of motion data. *Journal of Virtual Reality and Broadcasting 5*, 5 (July 2008). 2

[LLWY12] LIU J., LIU J., WONKA P., YE J.: Sparse non-negative tensor factorization using columnwise coordinate descent. *Pattern Recognition 45*, 1 (2012), 649–656. 2

[LXPER11] LIU G., XU M., PAN Z., EL RHALIBI A.: Human motion generation with multifactor models. *Journal of Visualization and Computer Animation 22*, 351–359 (2011), 4. 2

[MK07] MUKAI T., KURIYAMA S.: Multilinear motion synthesis with level-of-detail controls. In *Proceedings Pacific Conference on Computer Graphics and Applications* (November 2007), pp. 9–17. 2

© The Eurographics Association 2013.

[MLC10] MIN J., LIU H., CHAI J.: Synthesis and editing of personalized stylistic human motion. In *Proceedings ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games* (2010), pp. 39–46. 2

[Moo20] MOORE E. H.: On the reciprocal of the general algebraic matrix. *Bulletin of the American Mathematical Society 26* (1920), 394–395. 5

[Mor04] MORAVITZ MARTIN C. D.: Tensor decompositions workshop discussion notes. Amercan Institute of Mathematics, Palo Alto, CA, 2004. 2

[MUH11] MOROZOV O. V., UNSER M., HUNZIKER P.: Reconstruction of large, irregularly sampled multidimensional images. a tensor-based approach. *IEEE Transactions on Medical Imaging* (February 2011), 366–74. 2

[Pen55] PENROSE R.: A generalized inverse for matrices. In *Proceedings Cambridge Philosophical Society* (1955), vol. 51, pp. 406–413. 5

[PSK*07] PERERA M., SHIRATORI T., KUDOH S., NAKAZAWA A., IKEUCHI K.: Multilinear analysis for task recognition and person identification. In *Proceedings IEEE Conference on Intelligent Robots and Systems* (November 2007), pp. 1409–1415. 2

[PTVF92] PRESS W. H., TEUKOLSKY S. A., VETTERLING W. T., FLANNERY B. P.: *Numerical Recipes in C*. Cambridge University Press, 1992. 3

[RK09] RUITERS R., KLEIN R.: BTF compression via sparse tensor decomposition. *Computer Graphics Forum 28*, 4 (July 2009), 1181–1188. 2

[RSK12] RUITERS R., SCHWARTZ C., KLEIN R.: Data driven surface reflectance from sparse and irregular samples. *Computer Graphics Forum 31*, 2 (May 2012), 315–324. 2

[SBG04] SMILDE A., BRO R., GELADI P.: *Multi-Way Analysis: Applications in the Chemical Sciences*. Wiley, West Sussex, England, 2004. 2, 3, 4, 5

[SE07] SAVAS B., ELDÉN L.: Handwritten digit classification using higher order singular value decomposition. *Pattern Recognition 40*, 3 (2007), 993–1003. 2

[SH05] SHASHUA A., HAZAN T.: Non-negative tensor factorization with applications to statistics and computer vision. In *Proceedings ACM Machine Learning* (2005), vol. 119, pp. 792–799. 2

[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. 2

[SL01] SHASHUA A., LEVIN A.: Linear image coding for regression and classification using the tensor-rank principle. In *Proceedings IEEE Computer Vision and Pattern Recognition Conference* (December 2001), pp. 42–49. 2

[SS08] SCHULTZ T., SEIDEL H.-P.: Estimating crossing fibers: A tensor decomposition approach. *IEEE Transactions on Visualization and Computer Graphics 14*, 6 (2008), 1635–1642. 2

[Sut13] SUTER S. K.: *Interactive Multiresolution and Multiscale Visualization of Large Volume Data*. PhD thesis, University of Zurich, Switzerland, April 2013. 1, 13

[SZC*07] SUN X., ZHOU K., CHEN Y., LIN S., SHI J., GUO B.: Interactive relighting with dynamic BRDFs. *ACM Transactions on Graphics 26*, 3 (2007), 27. 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. 2

[TBDLK87] TEN BERGE J. M. F., DE LEEUW J., KROONENBERG P. M.: Some additional results on principal components analysis of three-mode data by means of alternating least squares algorithms. *Psychometrika 52* (1987), 183–191. 2, 9

[TS06] TSAI Y.-T., SHIH Z.-C.: All-frequency precomputed radiance transfer using spherical radial basis functions and clustered tensor approximation. *ACM Transactions on Graphics 25*, 3 (2006), 967–976. 2

[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), 19:1–19:17. 2, 4, 13

[Tuc66] TUCKER L. R.: Some mathematical notes on three-mode factor analysis. *Psychometrika 31*, 3 (September 1966), 279–311. 2, 9

[Vas02] VASILESCU M. A. O.: Human motion signatures: Analysis, synthesis, recognition. In *Proceedings IEEE Conference on Pattern Recognition* (2002), vol. 3, pp. 456–460. 2

[VBPP05] VLASIC D., BRAND M., PFISTER H., POPOVIĆ J.: Face transfer with multilinear models. *ACM Transactions on Graphics 24*, 3 (2005), 426–433. 2

[VT02] VASILESCU M. A. O., TERZOPOULOS D.: Multilinear analysis of image ensembles: TensorFaces. In *Procedings in European Conference on Computer Vision* (May 2002), vol. 2350, pp. 447–460. 2

[VT04] VASILESCU M. A. O., TERZOPOULOS D.: TensorTextures: Multilinear image-based rendering. *ACM Transactions on Graphics 23*, 3 (2004), 336–342. 2

[WA04] WANG H., AHUJA N.: Compact representation of multidimensional data using tensor rank-one decomposition. In *Proceedings Pattern Recognition Conference* (2004), pp. 44–47. 2

[WA05] WANG H., AHUJA N.: Rank-R approximation of tensors: Using image-as-matrix representation. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition* (2005), pp. 346–353. 2

[WA08] WANG H., AHUJA N.: A tensor approximation approach to dimensionality reduction. *International Journal of Computer Vision 76*, 3 (2008), 217–229. 2

[WSZP07] WAMPLER K., SASAKI D., ZHANG L., POPOVIĆ Z.: Dynamic, expressive speech animation from a single mesh. In *Proceedings SIGGRAPH/Eurographics Symposium on Computer Animation* (2007), pp. 53–62. 2

[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, 4

[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, 4

[YWT*09] YAN S., WANG H., TU J., TANG X., HUANG T. S.: Mode-kn factor analysis for image ensembles. *IEEE Transactions on Image Processing 18*, 3 (Mar 2009), 670–676. 2

[ZG01] ZHANG T., GOLUB G. H.: Rank-one approximation to high order tensors. *SIAM Journal of Matrix Analysis and Applications 23*, 2 (2001), 534–550. 9, 11

© The Eurographics Association 2013.