Vis Comput (2016) 32:1433–1446
DOI 10.1007/s00371-015-1130-y

Published article:
https://doi.org/10.1007/s00371-015-1130-y

Lossy Volume Compression using Tucker Truncation and Thresholding

Rafael Ballester-Ripoll · **Renato Pajarola**

Received: date / Accepted: date

Abstract Tensor decompositions, in particular the Tucker model, are a powerful family of techniques for dimensionality reduction and are being increasingly used for compactly encoding large multidimensional arrays, images and other visual data sets. In interactive applications, volume data often needs to be decompressed and manipulated dynamically; when designing data reduction and reconstruction methods, several parameters must be taken into account, such as the achievable compression ratio, approximation error and reconstruction speed. Weighing these variables in an effective way is challenging, and here we present two main contributions to solve this issue for Tucker tensor decompositions. First, we provide algorithms to efficiently compute, store and retrieve good choices of tensor rank selection and decompression parameters in order to optimize memory usage, approximation quality and computational costs. Second, we propose a Tucker compression alternative based on coefficient thresholding and zigzag traversal, followed by logarithmic quantization on both the transformed tensor core and its factor matrices. In terms of approximation accuracy, this approach is theoretically and empirically better than the commonly used tensor rank truncation method.

Rafael Ballester-Ripoll
Visualization and MultiMedia Lab, Department of Informatics, University of Zürich
Binzmühlestrasse 14
8050 Zurich, Switzerland
E-mail: rballester@ifi.uzh.ch

Renato Pajarola
Visualization and MultiMedia Lab, Department of Informatics, University of Zürich
Binzmühlestrasse 14
8050 Zurich, Switzerland
E-mail: pajarola@acm.org

Keywords tensor approximation · data compression · higher-order decompositions · tensor rank reduction · multidimensional data encoding

1 Introduction

Multidimensional signals and data sets such as multi-view, time-varying or multi-spectral video, image and volume data in interactive multimedia, scientific visualization and 3D graphics applications are often large in size and continue to grow at a rapid pace. Therefore, effective data reduction, compact data representation and compression techniques are necessary to manage such large multidimensional visual data sets. This is especially critical for higher dimensional data arrays, as their complexity can grow exponentially in what is known as the *curse of dimensionality*.

Many effective data reduction and compression methods are based on *transform coding* approaches, which first perform a data domain transformation followed by (vector) quantization or coefficient thresholding, and often conclude with a variable length (prediction-error entropy) coding of the remaining data coefficients. Well known examples include discrete cosine transform (DCT) based JPEG image or MPEG video compression, as well as wavelet transform (WT) based image, video and 3D volume data compression methods. In the context of compact visual data representation, *tensor approximation* (TA) methods have recently been shown to be a powerful domain transformation alternative [30,31,26,34,32,33,23,5,21,27,22,17].

The all-orthogonal Tucker decomposition [15,12] is an increasingly popular tensor-based technique for dimensionality reduction, which computes a least-squares fitting to a given N-dimensional input array $\mathscr{A}$ of size $I_1 \times \cdots \times I_N$. The fitting results in a multilinear decomposition of $\mathscr{A}$ into a set

of N orthonormal basis factor matrices $\mathbf{U}^{(n)}$ and the coefficients of an N-dimensional reduced core tensor $\mathscr{B}$.

This factorization approximates the input data set by a multilinear combination of the basis factor matrices columns weighted by the core-tensor coefficients (see also Section 3).

The factor matrices are crucial components of the decomposition: their column vectors represent the set of basis functions onto which the data is projected, and thus define the mapping between initial and compressed data and vice versa. These columns are commonly referred to as *tensor ranks*. While in many basis transform compression methods (such as Fourier Transform (FT), DCT and WT) the bases are pre-defined and independent of the input data, tensor decompositions rely on data-dependent bases learned directly from the input data itself. For TA compression methods, the weight coefficients (the core tensor elements) as well as the factor matrix bases are part of the output. Thus, the data-dependent basis functions trade increased approximation accuracy for additional representation cost, which is dominated, however, by the weight coefficients storage for higher dimensional data with $N \geq 3$.

Coefficient reduction is a key element of many compression algorithms. Under this concept, and after computing a transform of the input data, the least significant coefficients are eliminated in order to decrease the total memory needs. Therefore, it is important to develop and study effective data reduction strategies in the case of Tucker-based decomposition models: namely, which coefficients from the transform core tensor are considered the least significant and thus are preferable to discard for the sake of a compression in the most efficient way. We address these issues in this paper.

2 Related Work

In the 2D case, the singular value decomposition (SVD) was used for image coding by Andrews and Patterson [2]. They select the first and most signicant pairs of singular vectors, which are known capture the most energy of the image. Singular vectors form by definition an orthonormal basis; they use this redundancy to produce an even more compact representation and reduce the required coefficients by a 50% factor. For higher-order data, the Tucker decomposition technique has been described as a generalization of the SVD, frequently denoted as HOSVD. Lathauwer et al. [15] show the strong links that exist between SVD and HOSVD. The higher-order orthogonal iteration (HOOI) [16] is a popular algorithm for obtaining this decomposition. It produces orthogonal factor matrices, similar to the left- and right-singular vector matrices that SVD yields. While the Tucker model and the algorithms for its computation were born in the context of multiway data analysis [12], they are increasingly applied in multidimensional visual data compression, interactive visualization and computer graphics. Indeed, they have been compared favorably to 3D compression algorithms such as Fourier-based [19] or wavelet-based [34, 33, 23, 4]. Other graphics applications include compressing scattering response fields [14], BRDFs [20] and texture functions [29].

The Tucker model allows a variable number of ranks R_i for each tensor mode (alternatively called dimension, or way), as opposed to other models such as the CANDECOMP/PARAFAC (CP) [6]. The choice of the number of ranks determines the amount of dimensionality reduction to be performed along each mode, and the target size of the core tensor $\mathscr{B}$. In most Tucker-related applications, this is a critical matter and it strongly influences the resulting approximation accuracy. Typical 3D Tucker compression approaches involve knowing in advance the desired core tensor size, so that either a) a rank-$R_1R_2R_3$ approximation can be directly computed [16]; or b) a subset of an existing approximation is selected in favor of a lower memory usage, and at the cost of reduced reconstruction quality. Setting the target ranks R_i beforehand simplifies the problem and can be used for a faster decomposition (Vannieuwenhoven et al. [28]). However, selecting a meaningful number of ranks is a non-trivial issue [12]. Chen et al. [8] detail an algorithm for computing it, provided that a target compression ratio is given. They compute a full-rank approximation as an initial step, followed by a rank selection based on the core entries. Alternatively, rank reduction can be performed by further reducing an already non-full core, e.g. dynamic rank selection as used in Suter et al. [22].

Another major focus in the present paper is the study of hard thresholding, namely discarding the smallest, least significant coefficients of the approximation. Coefficient thresholding is a common practice in many forms of data compression (for example, DCT [35] or WT [7]). Tucker core thresholding has been used by Rajwade et al. [18], who exploit its high-frequency removal properties in the context of data denoising. A closely related topic is compressive sensing, which aims to recover data from a limited amount of observations (e.g. a sparse core). Several tensor recovery approaches have been proposed [9, 24, 13]; however, they focus on (almost) perfect recovering low-rank instances, usually because part of the input has been corrupted. To the best of our knowledge, no study has been conducted yet to explore the effectiveness of hard thresholding on HOSVD-based models for volume compression and, in particular, the Tucker decomposition for 3 or more dimensions.

In practical applications, additional quantization is often applied on the computed tensor decomposition (e.g. linear in [33], non-linear in [21]), but simply to reduce precision storage costs and not as the primary method to eliminate coefficients. Given the proven usefulness of quantization to reduce the overall size, and in order to evaluate our method in a realistic setting, we choose to combine thresholding with a

non-linear quantization step applied a posteriori. Under this perspective, hard thresholding can be regarded as an actual quantization approach that maps a whole region of elements to zero (the so-called *deadzone threshold*).

3 Multilinear Tensor Decomposition

3.1 Tucker Model

In this paper, we study the Tucker decomposition in the context of dimensionality reduction and compact multidimensional data representation. This model performs a least-squares fitting of an N-dimensional input data array $\mathscr{A} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ and defines a multilinear decomposition of $\mathscr{A}$ into the following:

- a set of N so-called *basis factor matrices*, denoted $\mathbf{U}^{(n)}$, each of dimension $I_n \times R_n$ for $n = 1, \ldots, N$, and
- an N-dimensional *core tensor* $\mathscr{B} \in \mathbb{R}^{R_1 \times \cdots \times R_N}$, which contains most of the data coefficients and, having a significantly smaller size than the input $\mathscr{A}$ for $R_i \ll I_i \ \forall i$, successfully allows for lossy data compression.

Given a fixed set of (orthogonal) bases $\mathbf{U}^{(n)}$ and an input data set $\mathscr{A}$, the corresponding coefficient core tensor is determined uniquely by

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

The n-mode product $\times_n$ denotes the *tensor-times-matrix* operation, which essentially projects data onto given basis factors. For more comprehensive details on tensor decomposition and notation we refer the reader to the survey of Kolda and Bader [12]. The transformation can be conveniently inverted in order to obtain an approximation of the original data:

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

This formulation is depicted in Figure 1 for the 3D case. The magnitude that the least squares fitting attempts to minimize is $\|\mathscr{A} - \widetilde{\mathscr{A}}\|$, where $\|\cdot\|$ denotes the Frobenius norm: $\|\mathscr{A}\| = \sqrt{\sum_{i,j,k} \mathscr{A}_{ijk}^2}$.

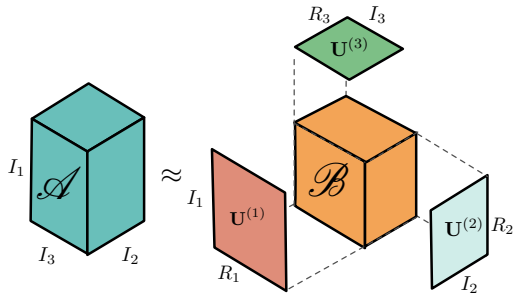


Fig. 1 The Tucker decomposition model for 3-way data, with a 3D $R_1 \times R_2 \times R_3$ core and 3 factor matrices of size $I_i \times R_i$.

3.2 Target Variables

For the ease of presentation, from now on the ideas in this work will be detailed for the 3D case. Extensions using the general higher-order Tucker decompositions just defined are easy to formulate from this point. Important parameters in the Tucker-based compression include:

- The *compression factor*, measuring the size after compression versus the original,

$$F = \frac{\text{size}(\mathscr{B}) + \text{size}(\mathbf{U}^{(1)}) + \text{size}(\mathbf{U}^{(2)}) + \text{size}(\mathbf{U}^{(3)})}{\text{size}(\mathscr{A})}$$

- The *decomposition* and *reconstruction* times, T_D and T_R, related to the amount of operations needed to compress/decompress a tensor to/from its decomposed format. They grow proportionally with the input size times the number of ranks chosen.
- The *relative error*, in the L_2 sense, of the approximation compared to the original: $\varepsilon = \|\mathscr{A} - \widetilde{\mathscr{A}}\| / \|\mathscr{A}\|$.

In general, the aim is to minimize all of these four quantities. Usually the first three are directly correlated with each other (a lower compressed size often demands less computing time) and inversely correlated with the fourth one, since improving them comes normally at the expense of reduced approximation quality.

Because Tucker's computing times T_D and T_R grow significantly with respect to the input data size [4], it is common practice for compression applications to split large data sets into smaller bricks; e.g. in an octree manner. Under this principle, several hierarchical tensor approaches have been proposed [34, 33, 23, 22].

3.3 Properties of Tucker and HOOI

We use the HOOI algorithm in the present work for solving the Tucker minimization problem, as there are a number of useful properties associated to both this technique and the Tucker tensor decomposition which we exploit. These are as follows:

1. Core norm invariance: the factors $\mathbf{U}^{(1)}$, $\mathbf{U}^{(2)}$ and $\mathbf{U}^{(3)}$ are orthonormal matrices, which means projecting a tensor onto them does not change its norm. Then from $\tilde{\mathscr{A}} = \mathscr{B} \times_1 \mathbf{U}^{(1)} \times_2 \mathbf{U}^{(2)} \times \mathbf{U}^{(3)}$, it follows that $\|\tilde{\mathscr{A}}\| = \|\mathscr{B}\|$.
2. Scalar product: $\langle \mathscr{A}, \tilde{\mathscr{A}} \rangle = \|\mathscr{B}\|^2$. A derivation can be found in [8].
3. Approximation error identity: $\|\mathscr{A} - \tilde{\mathscr{A}}\|^2 = \|\mathscr{A}\|^2 - 2\langle \mathscr{A}, \tilde{\mathscr{A}} \rangle + \|\tilde{\mathscr{A}}\|^2 = \|\mathscr{A}\|^2 - \|\mathscr{B}\|^2$ (using the previous identities). This provides a fast and handy way for relative error computation, that does not require explicit reconstruction: $\varepsilon = \sqrt{\|\mathscr{A}\|^2 - \|\mathscr{B}\|^2} / \|\mathscr{A}\|$

4. HOOI produces factor matrices which are column-wise orthonormal. It follows that $\left\langle \mathbf{U}^{(1)}_{i_1} \circ \mathbf{U}^{(2)}_{j_1} \circ \mathbf{U}^{(3)}_{k_1}, \mathbf{U}^{(1)}_{i_2} \circ \mathbf{U}^{(2)}_{j_2} \circ \mathbf{U}^{(3)}_{k_2} \right\rangle = 1$ if and only if $i_1 = i_2, j_1 = j_2, k_1 = k_2$, and 0 otherwise (the symbol $\circ$ denotes the outer product between vectors).
5. HOOI yields a core with slices in non-strictly decreasing norm along the 3 dimensions [15]. These norms can be regarded as a higher-order generalization of the singular values of a matrix. The largest core coefficients often tend to concentrate around the first corner, often known as the core's *hot corner*.
6. Tensor-times-matrix commutativity [15]: the factors can always be permuted when they operate along different modes ($\mathscr{B} \times_i \mathbf{U}^{(i)} \times_j \mathbf{U}^{(j)} = \mathscr{B} \times_j \mathbf{U}^{(j)} \times_i \mathbf{U}^{(i)}$ for $i \neq j$).

3.4 Logarithmic Quantization

Quantizing the resulting tensor decomposition coefficients has already proven useful in several graphics applications and is thus an established way to further compress the data, at the expense of some newly introduced error. In order to account for this effect, we incorporate in our experiments a logarithmic quantization scheme of the Tucker core, similar to [21]. Every coefficient $x \in \mathscr{B}$ gets scaled to 9 bits. 8 of them are used to quantize its absolute value: $|x|$ becomes $255 \cdot \log_2(1+|x|)/\log_2(1+\max(|\mathscr{B}|)) \in [0,255]$. The remaining bit encodes the sign of the original value. Due to its large magnitude, it is beneficial to separately encode the hot corner $\mathscr{B}(1,1,1)$. For this reason we store this element separately and do not consider it for the quantization formula.

The factor matrices' $\mathbf{U}^{(n)}$ impact on the overall TA storage grows in relation to the core tensor $\mathscr{B}$'s size with smaller number of ranks. Further importance is added to the factor matrices' sizes when applying the core thresholding technique because it removes additional coefficients from the core. This motivates also applying a quantization scheme to the factor matrices as well. In our experiments (Section 6) we compare both approaches: global quantization (core and factors) versus quantizing solely the core.

4 Core Truncation

As mentioned before, eliminating coefficients of the decomposition is a common tensor-based compression approach. The simplest way to select a subset of elements is to truncate them by discarding part of the basis functions and the corresponding slices of the core. Because of Property 5 in Section 3.3, the optimal subset of bases to keep must be chosen from the left, i.e. the columns $\mathbf{U}^{(n)}_{1\ldots R_n}$ where $R_n \leq I_n$ for $n = \{1,2,3\}$ and the core elements $\mathscr{B}(1:R_1, 1:R_2, 1:R_3)$. This *rank truncation* is illustrated in Figure 2, and it is computationally inexpensive. It is therefore a useful alternative to directly computing a rank-reduced decomposition by means of an HOOI algorithm.

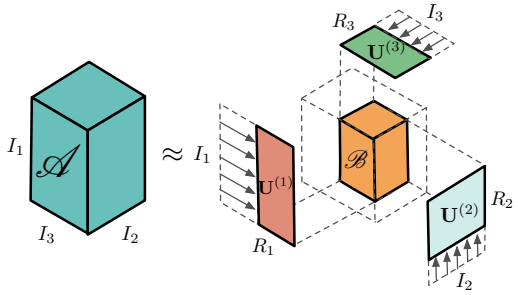


Fig. 2 The Tucker rank truncation data reduction strategy.

While not being optimal, Lathauwer et al. [16] support that such a truncation scheme is usually a very good approximation of what is obtained with a direct decomposition. Hackbusch [10] gives a bound on the truncation error, which is at most $\sqrt{N}$ times larger than the smallest possible error among all solutions with the same ranks.

4.1 Non-symmetric Cores

In this paper we choose to explore arbitrary core shapes (where R_1, R_2 and R_3 may be distinct) since, in general, they allow for better results than symmetric truncation (where no asymmetry of the core values is assumed, and the condition $R_1 = R_2 = R_3$ is imposed). An exact rank-(R_1,R_2,R_3) Tucker decomposition can be relatively easily found [12] if each R_i is chosen as $\text{rank}_i(\mathscr{A})$. This is defined as the dimension of the vector space spanned by the mode-i fibers of $\mathscr{A}$. A mode-i fiber of a tensor is the subspace obtained by freely moving the i-th index, while fixing all the others. In our setting we focus on lossy compression and will tolerate a certain small error rather than imposing equality. This means that we aim for a good choice of $R_{\{1,2,3\}}$ that maximizes the compression ratio while minimizing the relative error, or optimizing a function that assigns a weight to each of these. The issue can be tackled by computing a large enough core and set of bases and selecting a good enough subset from them. Chen et al. [8] make use of the observation that, after computing a full-rank decomposition, the shape of the truncation (i.e. the number of columns kept per dimension) can be selected for minimizing the error by maximizing the norm of the selected subcore (Property 3). The authors assume a target compression factor and design an algorithm that takes the entire core as input and finds good enough rank values. Instead, we propose to obtain the whole set of good solutions because of two reasons: a) we wish to exhaustively evaluate all the truncation possibilities over a Tucker core; and b) for practical situations where the relative

weight of the target variables can change frequently (such as applications in interactive visualization), it is convenient to quickly retrieve an optimal solution without having to revisit the whole core each time. As we show in the next section, we can efficiently precompute and store these solutions in a comparatively small list.

4.2 The Optimal Solution Curve

Rank truncation is a discrete operation: only an integer number of ranks may be selected, and often the exact desired compression factor F cannot be achieved. Thus, in general, we must admit a margin of variability to evaluate the truncation strategy. In order to provide a broad comparison (error versus many different compression factors), for every possible subcore $\mathscr{B}_i$ we compute a) the relative error $\varepsilon_i = \sqrt{\|\mathscr{A}\|^2 - \|\mathscr{B}_i\|^2}/\|\mathscr{A}\|$ (we quickly obtain the norm of the subcores by progressively building a *summed area table* in $O(I_1 I_2 I_3)$ operations); and b) the compression factor $F_i = (R_1R_2R_3 + I_1R_1 + I_2R_2 + I_3R_3)/(I_1I_2I_3)$ based on the number of coefficients. This way we produce a two-dimensional point plot that contains all $I_1I_2I_3$ possible solutions. Eventually, we want to discard every solution for which there exists a better one in terms of both F and ε. We can do this by sorting them in ascending order with respect to F and storing a variable with the best ε achieved so far (Alg. 1). We traverse the sorted list and keep a point only when it improves the best ε; i.e. a point i is selected if and only if for every $j \neq i$, either $F_i < F_j$ or $\varepsilon_i < \varepsilon_j$.

Algorithm 1 Computing a set C of optimal rank choices for a Tucker decomposition of the input tensor $\mathscr{A}$ with core $\mathscr{B}$.

```
L ← {}
B ← square(𝓑) {Each element is squared individually}
B ← summedAreaTable(B) {Computed inductively}
for R1 = 1,...,I1 do
    for R2 = 1,...,I2 do
        for R3 = 1,...,I3 do
            F ← (R1R2R3 + I1R1 + I2R2 + I3R3)/(I1I2I3)
            ε ← sqrt(||𝓐||^2 − B_{R1R2R3}^2)/||𝓐||
            L ← L ∪ (F, ε, R1, R2, R3)
        end for
    end for
end for
L ← sort(L) {Increasing order}
C ← {}
ε0 ← ∞
for i = 1,...,size(L) do
    (F, ε, R1, R2, R3) ← L.getElement(i)
    if ε < ε0 then
        C ← C ∪ (F, ε, R1, R2, R3)
        ε0 ← ε
    end if
end for
return C
```

The result is a connected subset C of the *orthogonal convex hull* of the point plot, which means that by joining the points it defines a piecewise curve that we can use for comparison with other compression methods. Only a border of the original point plot is selected with this approach, yielding a much smaller structure compared to the initial global pool of choices. Figure 3 shows a small example of such an optimal curve retrieval.

If needed, C can be stored in a tree-like search structure in order to quickly retrieve optimal solutions, e.g. determine the lowest possible ε for a given F, or vice versa. Even though we mostly target these two variables in the present work, it is worth noting that the concept of optimal solution curve just outlined above can be extended in practical applications to include other variables. For instance, T_R can be estimated for each triplet (R_1, R_2, R_3) as detailed in section 4.5, and used as a target variable.

4.3 Computation Time

The cost of computing the optimal curve of solutions is dominated by sorting the list of $I_1I_2I_3$ solutions, amounting to $O(I_1I_2I_3 \log(I_1I_2I_3))$ operations. This is a relatively small amount compared to the cost of producing the Tucker decomposition in the first place, $O(I_1I_2I_3R_1)$, with R_1 often not far from $I_1/2$.

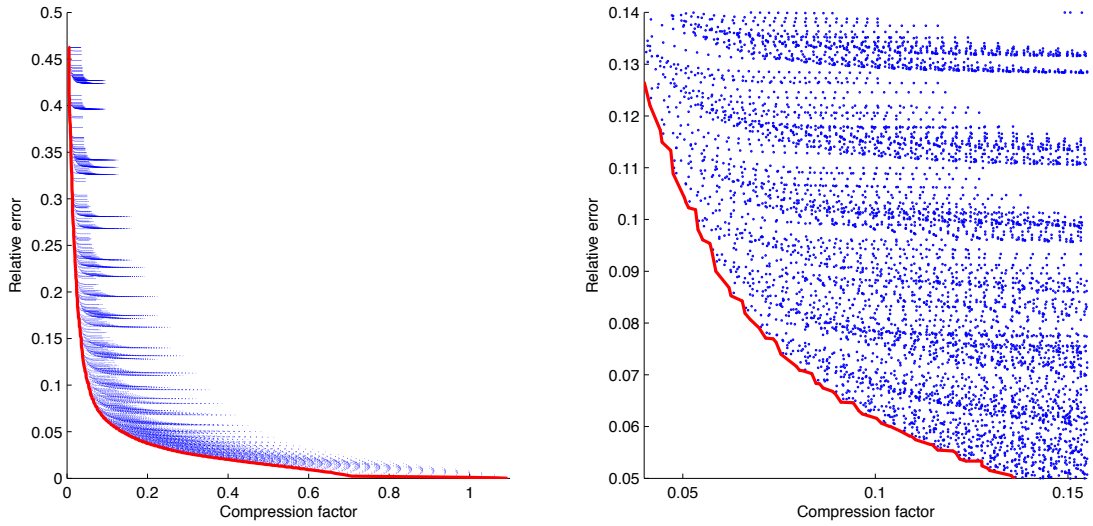
4.4 Space Requirements

Truncating a full-rank core tensor decreases the overall decomposition size from $I_1I_2I_3 + I_1^2 + I_2^2 + I_3^2$ to $R_1R_2R_3 + I_1R_1 + I_2R_2 + I_3R_3$ elements.

4.5 Reconstruction Time

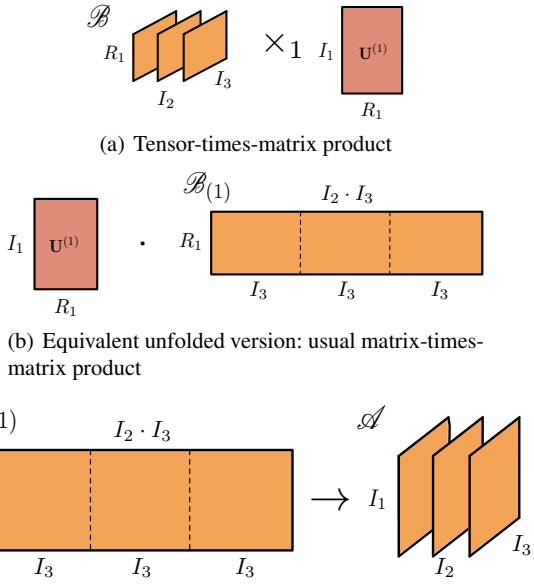
The Tucker core truncation leaves the remaining elements arranged in a conveniently compact fashion. As a bonus, the procedure reduces the number of basis functions (factor matrix elements) that need to be stored. While a naive Tucker strategy traversing each core element individually would take time $O(I_1I_2I_3R_1R_2R_3)$, it has been shown that the reconstruction can be efficiently performed in time $O(I_1I_2I_3R_1 + I_2I_3R_1R_2 + I_3R_1R_2R_3) = O(I_1I_2I_3R_1)$ by exploiting the compact structure and applying successive *unfoldings* [21,22]. Tensor unfolding, also known as *matricization*, expresses an N-dimensional tensor as a matrix by slicing it and stitching together the resulting slabs. Figure 4 shows an example in the 3-way case.

This transformation turns tensor-times-matrix operations into the more convenient matrix-matrix products: $\mathscr{B}' = \mathscr{B} \times_n \mathbf{U}^{(n)} \Rightarrow \mathscr{B}'_{(n)} = \mathbf{U}^{(n)} \mathscr{B}_{(n)}$. Efficient implementations



(a) Global set (b) Detail (zoom-in)

Fig. 3 In blue, set of all 32768 possible truncation choices for a full Tucker decomposition of a volume sized 32^3 that lies in the center of a CT scan of a Bonsai (see Section 6.2). In red, the curve connecting the 285 optimal solutions (0.87% of all possibilities).



(a) Tensor-times-matrix product

(b) Equivalent unfolded version: usual matrix-times-matrix product

(c) Result, folded back into a 3-way tensor

Fig. 4 Unfolding (matricization) to perform a tensor-times-matrix product in $O(I_1I_2I_3R_1)$ operations. This multiplication is the last and most expensive out of the 3 ones that must be computed for a complete reconstruction, yielding a final $I_1 \times I_2 \times I_3$ tensor.

exist for the matrix product; the result is folded afterwards in order to obtain a tensor again. In addition and because of Property 6, the ordering of the factors can be varied as desired for optimizing the speed, so that the final cost is $O(I_1I_2I_3R_i)$ with $R_i = \min\{R_1,R_2,R_3\}$.

5 Core Thresholding

When eliminating core coefficients in the Tucker model, rank truncation is not the only possibility. There are in fact $2^{I_1I_2I_3}$ possible subsets of elements to choose. In this section, we show that removing the elements with smallest norm is the optimal way to minimize the L_2 approximation error. To prove this claim, it suffices to write the approximation as a sum of orthogonal basis elements. Let $\mathscr{A} = \mathscr{B} \times_1 \mathbf{U}^{(1)} \times_2 \mathbf{U}^{(2)} \times \mathbf{U}^{(3)}$ be an exact (full-rank) decomposition of the original, and $\tilde{\mathscr{A}} = \tilde{\mathscr{B}} \times_1 \mathbf{U}^{(1)} \times_2 \mathbf{U}^{(2)} \times \mathbf{U}^{(3)}$ a thresholded approximation. Let us number the elements of the core $\mathscr{B}$ as a sequence $B_1,\dots,B_N$ (having indices $\{i_1,j_1,k_1\},\dots,\{i_N,j_N,k_N\}$, and $N = I_1I_2I_3$), in such an order that the thresholded core $\tilde{\mathscr{B}}$ has coefficients $B_1,\dots,B_M$ with $M < N$ and zeros elsewhere. Then

$$\|\mathscr{A} - \tilde{\mathscr{A}}\|^2 = \langle \mathscr{A} - \tilde{\mathscr{A}}, \mathscr{A} - \tilde{\mathscr{A}} \rangle = \left\langle \sum_{n=M+1}^{N} B_n \cdot \boldsymbol{U}^{(1)}_{i_n} \circ \boldsymbol{U}^{(2)}_{j_n} \circ \boldsymbol{U}^{(3)}_{k_n}, \sum_{n=M+1}^{N} B_n \cdot \boldsymbol{U}^{(1)}_{i_n} \circ \boldsymbol{U}^{(2)}_{j_n} \circ \boldsymbol{U}^{(3)}_{k_n} \right\rangle = \tag{3}$$
$$= \sum_{n=M+1}^{N} \sum_{m=M+1}^{N} B_n B_m \cdot \left\langle \boldsymbol{U}^{(1)}_{i_n} \circ \boldsymbol{U}^{(2)}_{j_n} \circ \boldsymbol{U}^{(3)}_{k_n}, \boldsymbol{U}^{(1)}_{i_m} \circ \boldsymbol{U}^{(2)}_{j_m} \circ \boldsymbol{U}^{(3)}_{k_m} \right\rangle.$$

Because of Property 4, the above sum can be simplified to $\sum_{n=M+1}^{N} \|B_n\|^2$; i.e. to minimize the error, the selected elements should be precisely the smallest in norm. This manifests the sub-optimality of rank truncation: it removes entire slices of $\mathscr{B}$ of lowest norm, whereas the best choice is always the smallest individual coefficients. These two are correlated but in general far from coincident, as shown by the experiments in Section 6. This argument holds as long

as we employ a Tucker decomposition algorithm that yields orthonormal matrices, such as generated by HOOI.

We can compute an optimal solution curve with the same properties as the ones described for the truncated case. Such a computation is given by Algorithm 2. In this case, the curve (Figure 5) contains as many points as the original core, $I_1 I_2 I_3$.

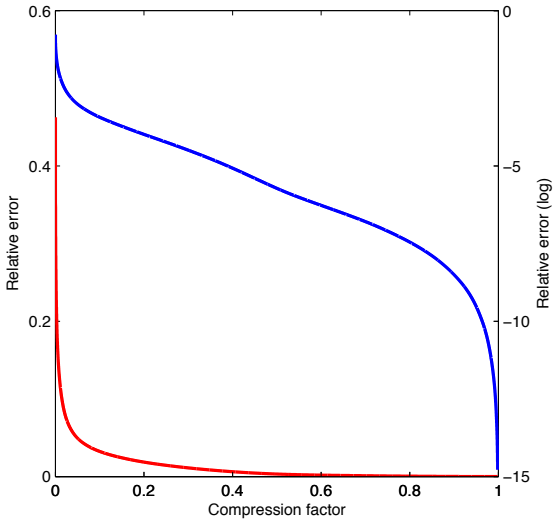


Fig. 5 In red, set of all 32768 thresholding choices for a full Tucker decomposition of the 32^3 brick in the center of the Bonsai. The error decreases logarithmically except in the left and right extremes, as shown by the blue curve.

Algorithm 2 Computing a set C of threshold choices for a Tucker decomposition of the input tensor $\mathscr{A}$ with core $\mathscr{B}$.

$C \leftarrow \{\}$
$B \leftarrow \text{sort}(\text{abs}(\text{vector}(\mathscr{B})))$ {The elements linearly arranged, in absolute decreasing order}
$B' \leftarrow \text{square}(B)$ {Each element is squared individually}
$B' \leftarrow \text{summedPrefixTable}(\text{square}(B))$ {A summed area table in the case of an array}
for $i = 1, \ldots, I_1 I_2 I_3$ **do**
  $F \leftarrow i/(I_1 I_2 I_3)$
  $\varepsilon \leftarrow \sqrt{\|\mathscr{A}\|^2 - B'_i}/\|\mathscr{A}\|$
  $\{R_1, R_2, R_3\} \leftarrow \text{boundingBox}(\text{threshold}(\mathscr{B}, B_i))$
  $C \leftarrow C \cup (F, \varepsilon, R_1, R_2, R_3)$
end for
return C

5.1 Computation Time

The most expensive part when obtaining the optimal curve cost in this case is sorting, as in the truncation algorithm. All the core coefficients must be handled, amounting again to $O(I_1 I_2 I_3 \log(I_1 I_2 I_3))$ operations for linear ordering.

5.2 Space Requirements

Memory-wise, not only must the actual coefficient values of a sparse data set be stored, but also their positions must be recorded. At this point, we face the challenge of lossless compression of a volume consisting of binary elements (each element meaning either that the coefficient was thresholded, or that it stayed unchanged). Several approaches exist for sparse compression, including encoding algorithms and hierarchical strategies for pruning empty regions. In our case, there is one pattern in the spatial distribution of absent coefficients that can be exploited well. We observe that the core elements tend to be much larger at, or close to the hot corner (Property 5). In other words, most thresholded elements usually are far away from that hot corner, which suggests the usage of an element ordering that reflects this tendency. We implemented a 3D generalization of the commonly used zigzag scheme for image entropy encoding (as used in the JPEG standard [11]). We traverse the core $\mathscr{B}$ in diagonal slices, starting at the hot corner and progressively moving further away from it along the cube diagonal; each of these slices is in turn traversed in a zigzag fashion. This traversal is illustrated in Figure 6.

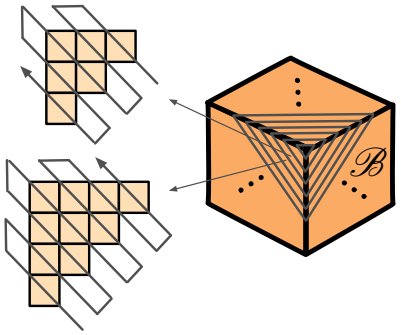


Fig. 6 Slice-based zigzag traversal of a Tucker core $\mathscr{B}$. In detail, the third and fourth slices are shown.

The result is a binary vector of length $R_1 R_2 R_3$ with elements 0 or 1 indicating absence or presence of individual coefficients. After this step, which takes advantage of coefficient locality, we apply a run length encoding step on that vector, followed by Huffman coding in order to compress adjacent blocks of information with the same binary value as much as possible. The Huffman coding provides a good compact representation of the presence bits (up to 8-fold compression in our experiments, depending on the data

set) while offering superior reconstruction speed than other methods such as arithmetic coding.

Figure 7 summarizes the algorithm steps detailed so far and the final space requirement depends on the effectivity of the zigzag run-length and entropy coding. Figure 8 depicts the resulting state of $\mathscr{B}$ and $\mathbf{U}^{(n)}$ after each of the 3 methods is applied at a compression factor of 0.25. Thresholding is the most adaptive algorithm, as it focuses on keeping the core coefficients with the highest energy. On the other hand, it must store a larger set of basis functions.

5.3 Reconstruction Time

As opposed to rank truncation, thresholding means eliminating coefficients from inner areas of the core. This results in a significantly larger bounding box, which in turn negatively impacts the reconstruction speed. Fortunately and due to the usual hot corner distribution, the discarded coefficients tend to lie in the outermost regions of the core. As a consequence, these regions can be trimmed conveniently when they become empty. We account for this in our implementation, and store the bounding box of the remaining coefficients.

6 Experimental Results

So far we have detailed the two main strategies that we wanted to analyze: non-symmetric truncation (NST) and thresholding (T). In this section, we conduct experimental measurements of their performance and compare them against each other. We also compare them against the traditional symmetric truncation (ST), which serves as a baseline algorithm. In order to conduct an as exhaustive exploration as possible, we truncate or threshold the decompositions to the fullest possible range: from discarding no coefficients ($F > 1$ and $\varepsilon = 0$) to discarding everything (which gives an empty tensor, $F = 0$ and $\varepsilon = 1$).

6.1 Hardware and Software

The experiments were run on a 16-core Intel Xeon CPU E5-2670 with 2.60GHz and 32GB of RAM. All algorithms were implemented in MATLAB 2014a, using the Tensor Toolbox [3]. MATLAB makes use of highly optimized, state-of-the-art routines from the BLAS and LAPACK libraries in order to perform tensor-times-matrix operations and computations of eigenvalues and singular vectors.

6.2 Input Data Sets

We have used four distinct data sets in our experiments: a *Bonsai* tree, a human *Foot* and *Skull*, and a piece of *Wood*; all are 8-bit computer tomography (CT) scans of size 256^3. The first three are standard volumes publicly available at volvis.org [1]. For our experiments, the data sets are split into $4^3 = 64$ bricks, each of size 64^3, which are all processed independently. The numerical results (compression error and needed time) are then averaged between all bricks of each test set. Empty bricks (those which contain only zeros) are not considered for the experiments: these are 1.56% of the total bricks for the Bonsai, 7.81% for the Foot and 0% for the Skull and the Wood. All algorithms take the same per-brick input core $\mathscr{B}$, obtained with the HOOI algorithm (3 iterations). The average per-brick decomposition time T_D was 129ms for the Bonsai, 134ms for the Foot and 136ms for the Skull and the Wood.

6.3 Results

Table 1 shows the resulting mean and median approximation error for every algorithm and quantization scheme used, for 4 different compressions F; Table 2 shows the mean and median tensor reconstruction time T_R in milliseconds. Figures 9 and 10 display the average across all data sets.

A substantial accuracy improvement can be observed between the non-global quantization employed in [22] and the proposed alternative which quantizes also the matrices. In the latter, the thresholding approach consistently attains a better compression performance than the other two core reduction strategies; this is achieved in exchange for a larger reconstruction time. In order to explore an exhaustive range of compression factors, we display the errors for each possible factor F as a rate distortion diagram in Figure 11. The three algorithms and both quantization strategies are considered.

Figure 12 visually shows the accuracy of each method by displaying their different error magnitudes, in absolute value.

Finally and in order to put the tested techniques in perspective, we conduct a comparison with DCT compression and several wavelet transform-based methods. Results are shown in Fig. 13 and Fig. 14 in terms of accuracy and reconstruction time, respectively. The DCT coefficients are non-symmetrically truncated and then quantized as in the method we exposed for the tensor case. The WT coefficients are compressed with a standard scalar quantization scheme as used in [25]: a quantization step h that determines the overall resulting compression rate is defined for the first wavelet level, and reduced to $h/2^l$ for any other level l as average coefficient energy tends to decrease in subsequent levels. The quantization function is $f(x) = \text{sign}(x) \cdot \text{round}(|x|/h)$. The resulting integers are then concatenated in scan-line order, and compressed via run-length encoding followed by entropy encoding (with the Huffman algorithm).

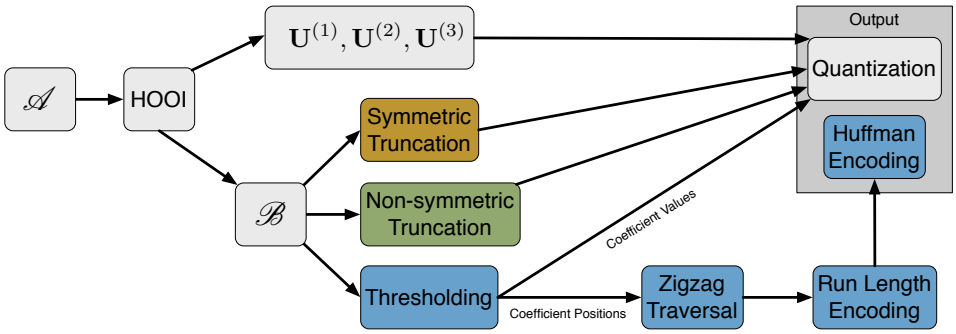
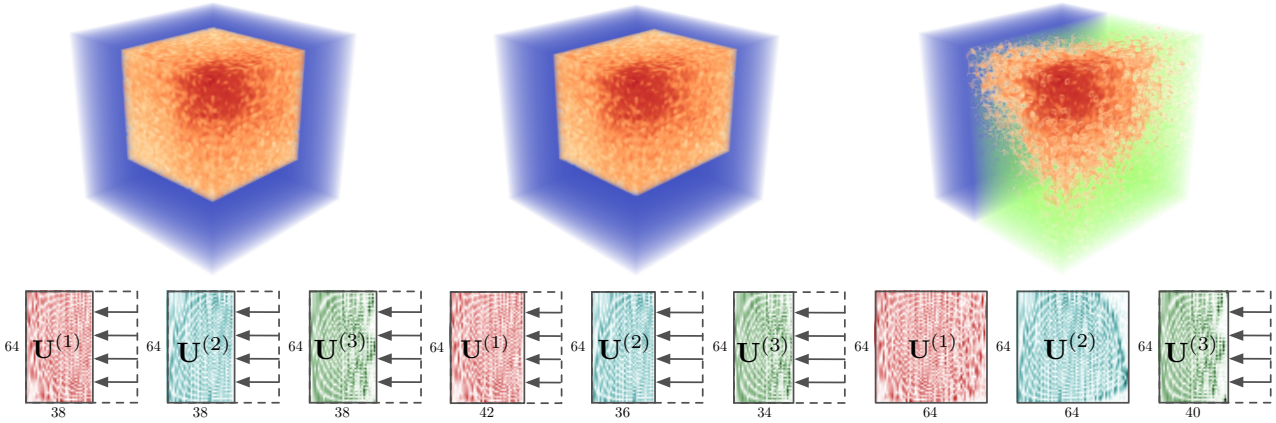


Fig. 7 Compression algorithm flow chart. In color, the three alternative coefficient reduction techniques discussed.



(a) Symmetric truncation (ST) (b) Non-symmetric truncation (NST) (c) Thresholding (T)

Fig. 8 Resulting core and factor matrices for the 3 analyzed strategies. Non-zero core coefficients are mapped to a orange-red logarithmic scale, while blue represents the zone outside the bounding boxes. Thresholded coefficients are shown in green. The data set employed is a sub-brick of size 64^3 lying in the center of the Bonsai, and the compression factor $F = 0.25$ was chosen. The hot corner distribution is clearly visible in the red core regions.

		$\mathscr{B}$ **quantized**				$\mathscr{B}$, $\mathbf{U}^{(n)}$ **quantized**			
		$F = 0.1$	$F = 0.25$	$F = 0.5$	$F = 0.75$	$F = 0.1$	$F = 0.25$	$F = 0.5$	$F = 0.75$
Bonsai	**ST**	0.2312	0.1449	0.0888	0.0616	0.1832	0.1198	0.0785	0.0517
	NST	0.2164	0.1231	0.0565	0.0181	0.1684	0.0972	0.0394	0.0116
	T	0.2650	0.1082	0.0317	0.0094	0.1417	0.0625	0.0216	0.0109
Foot	**ST**	0.2624	0.1790	0.1103	0.0752	0.2186	0.1495	0.0960	0.0618
	NST	0.2485	0.1601	0.0944	0.0566	0.2048	0.1339	0.0780	0.0461
	T	0.2881	0.1448	0.0599	0.0280	0.1677	0.0890	0.0422	0.0207
Skull	**ST**	0.1495	0.1004	0.0710	0.0548	0.1209	0.0874	0.0646	0.0475
	NST	0.1474	0.0986	0.0714	0.0539	0.1207	0.0878	0.0643	0.0469
	T	0.1675	0.1041	0.0591	0.0350	0.1143	0.0748	0.0456	0.0235
Wood	**ST**	0.2449	0.1763	0.1172	0.0872	0.2109	0.1500	0.1046	0.0744
	NST	0.2395	0.1747	0.1144	0.0733	0.2097	0.1526	0.0962	0.0605
	T	0.2683	0.1767	0.0894	0.0485	0.1955	0.1203	0.0655	0.0318

Table 1 Average relative approximation error ε per brick for each algorithm and quantization approach.

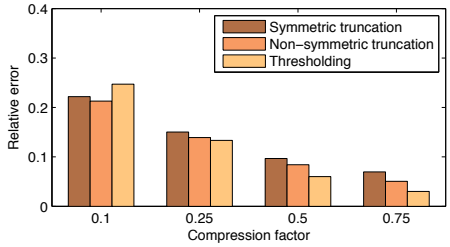
7 Discussion and Future Work

In the view of the presented experiments, the following observations can be made:

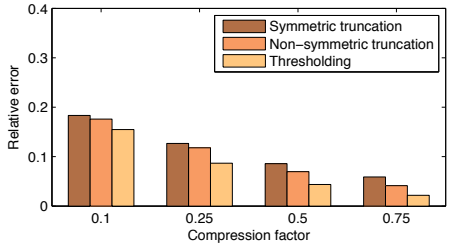
- The non-symmetric truncation approach almost always attains a higher approximation accuracy than the symmetric one, to an extent which depends on the data set.
- When global quantization is used, the thresholding technique offers superior compression rate distortion quality compared to the standard Tucker rank reduction methods, including both symmetric [22] and non-symmetric [8] truncations. At high compression factors it is competitive with respect to transform coding approaches such as wavelets, supporting previous results

		$\mathscr{B}$ quantized				$\mathscr{B}$, $\mathbf{U}^{(n)}$ quantized			
		$F = 0.1$	$F = 0.25$	$F = 0.5$	$F = 0.75$	$F = 0.1$	$F = 0.25$	$F = 0.5$	$F = 0.75$
Bonsai	**ST**	3.8446	4.2959	5.2134	5.5450	4.1429	4.5262	5.5114	5.5284
	NST	3.7657	4.0305	4.4759	4.8817	3.7997	4.1117	4.7677	5.0552
	T	8.9881	8.4843	7.2308	7.6654	8.2212	8.0934	7.2610	7.1423
Foot	**ST**	3.8770	4.4725	5.1247	5.4767	4.1748	4.6117	5.2687	5.4726
	NST	3.9726	4.4260	5.0217	5.3732	4.2434	4.5406	5.1138	5.3667
	T	9.1605	8.0424	7.1200	7.3665	7.8778	7.8396	7.0456	7.2213
Skull	**ST**	3.9501	4.4630	5.1294	5.4551	4.1137	4.6243	5.3222	5.3470
	NST	3.8743	4.3644	5.3227	5.3806	4.1097	4.6530	5.2909	5.4483
	T	8.9021	8.1639	6.6944	7.2219	7.9507	7.2108	6.9709	6.9563
Wood	**ST**	3.8421	4.4615	5.2555	5.2811	4.2080	4.4708	5.2963	5.2936
	NST	3.6515	4.3544	5.4385	5.5938	3.9827	4.6037	5.6178	5.7099
	T	8.8999	8.0792	6.7928	7.1857	8.1835	7.5583	6.8774	7.2586

Table 2 Average reconstruction time T_R (in milliseconds) per brick for each algorithm and quantization approach.

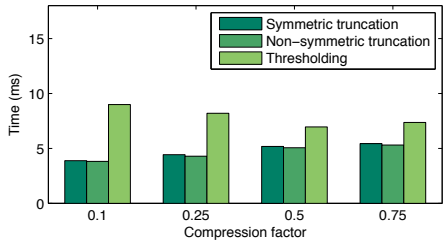


(a) $\mathscr{B}$ quantized

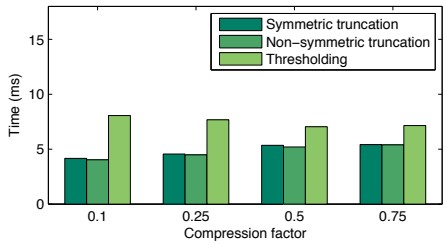


(b) $\mathscr{B}$, $\mathbf{U}^{(n)}$ quantized

Fig. 9 Average (for the 4 data sets) per-brick relative approximation error ε, for each algorithm and quantization approach.



(a) $\mathscr{B}$ quantized



(b) $\mathscr{B}$, $\mathbf{U}^{(n)}$ quantized

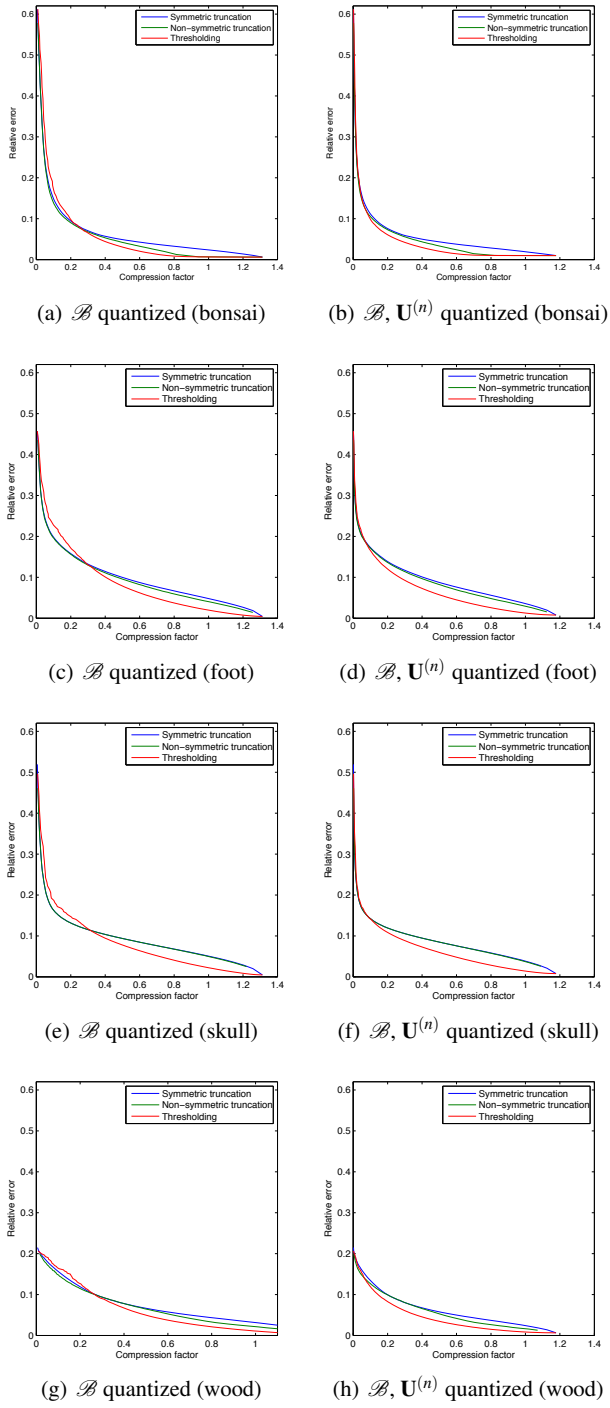
Fig. 10 Average (for the 4 data sets) per-brick reconstruction time T_R (in milliseconds), for each algorithm and quantization approach.

which have highlighted the advantages of tensor-based compression techniques in terms of perceptual quality and data feature preservation [33,23].

- In terms of reconstruction time, non-symmetric truncation achieves a similar or even slightly better performance when compared to its symmetric counterpart. Thresholding is slower: the number of resulting ranks is higher, entailing more expensive tensor-times-matrix operations for the reconstruction stage.
- The coefficient layout in the Tucker core is typically denser near the hot corner, and can be well entropy-encoded by means of a zigzag slice-based traversal.
- Quantization of the basis factor matrices in addition to the core tensor provides better rate distortion in all cases. It is particularly beneficial for the thresholding method.

We have tested a limited set of 3D CT-scans, but the results shown are already promising; in the future, more extensive experiments will be required to increase their statistical significance. Two other interesting future work lines of research exist:

1. Hybrid strategies between truncation and thresholding can be also explored: removing coefficients from inside the core that have a very small contribution, while truncating the sparsest outer parts of it. Such a mixture is potentially well suited to further refine a compromise between tensor compression accuracy and speed.
2. The ideas detailed in [2] for 2D SVD coding can be considered for higher dimensions. Unlike SVD, in the Tucker model the majority of transform coefficients lie in the core, rather than in the factor matrices. However,



(a) $\mathscr{B}$ quantized (bonsai) (b) $\mathscr{B}$, $\mathbf{U}^{(n)}$ quantized (bonsai)

(c) $\mathscr{B}$ quantized (foot) (d) $\mathscr{B}$, $\mathbf{U}^{(n)}$ quantized (foot)

(e) $\mathscr{B}$ quantized (skull) (f) $\mathscr{B}$, $\mathbf{U}^{(n)}$ quantized (skull)

(g) $\mathscr{B}$ quantized (wood) (h) $\mathscr{B}$, $\mathbf{U}^{(n)}$ quantized (wood)

Fig. 11 Compression quality, rate distortion performance for the considered methods when applied on a data brick example of size 64^3 lying in the center of each data set. Global quantization (bottom) causes less error, except at the very right end.

in the same way we showed matrix quantization can significantly improve the attained accuracy, other singular vector coding strategies are worth exploring. In particular, a) exploiting matrix orthonormality to remove redundant elements, and b) coding the difference between adjacent values of the 1D Tucker basis functions (differential pulse-code modulation, or DPCM). Care must be taken if coupling these strategies with quantization, as the combined error should be minimized.

8 Conclusions

In this paper, we have improved the standard Tucker decomposition based compression in two ways. First, we provided a method for obtaining a set of optimal core truncation solutions. We believe this concept is useful for successfully retrieving the most relevant ranks out for each dimension of a Tucker decomposition, and it helps to summarize the information distribution of the input data. In the context of multiway data analysis, the technique shows potential for assessing the degree of data complexity contained along each dimensional axis. Regarding data compression applications over regular grids, it allows for a more informed choice of the size for each basis, which offers a lower approximation error.

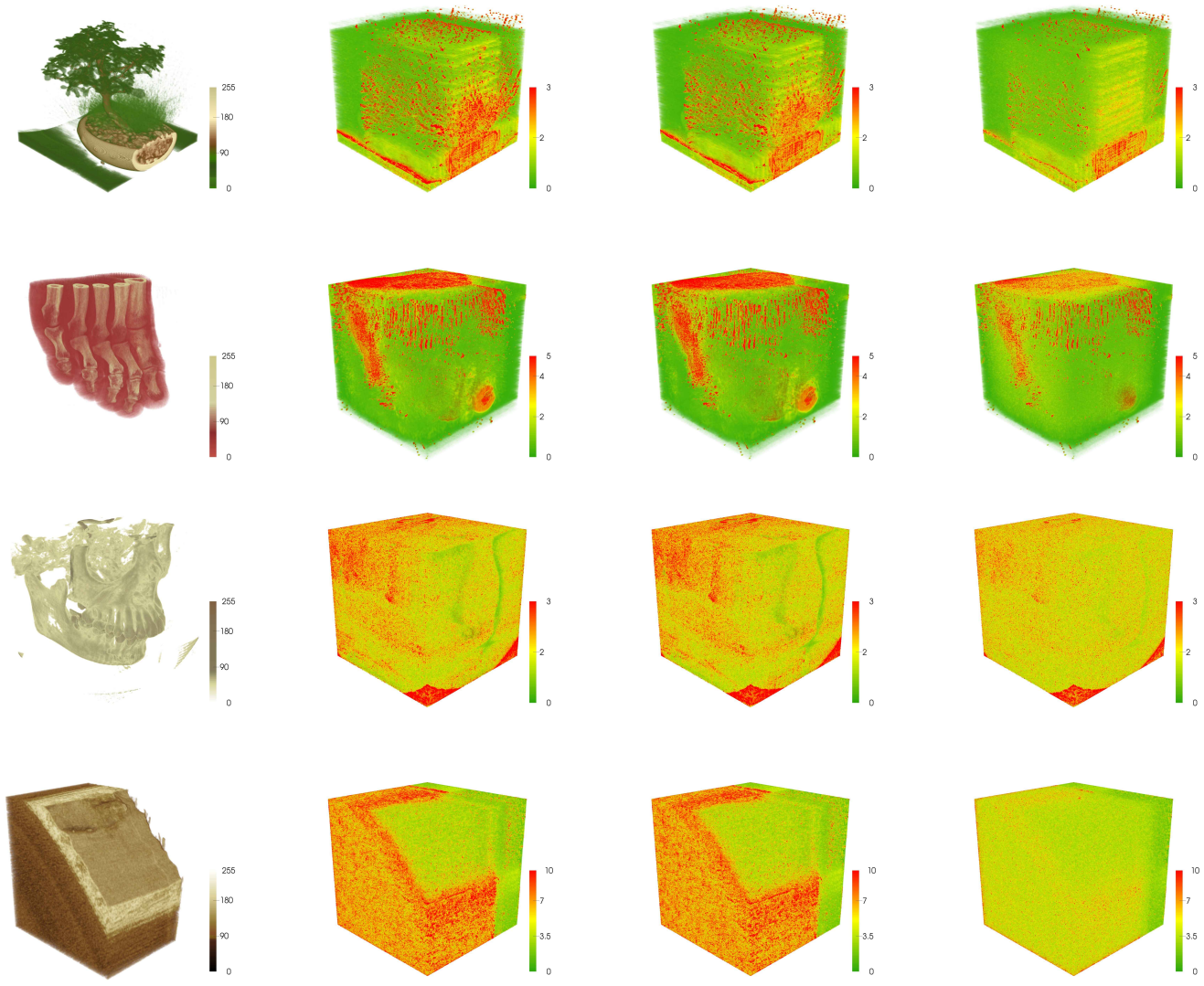
We also studied Tucker coefficient thresholding, and further improved the compression factor at the expense of a higher reconstruction time cost. It is thus preferable when storage or data transfer limitations outweigh the available computational power. Core truncation and thresholding are two opposite extremes of this trade-off, each of them optimizing a different property of the decomposition.

We tested and compared the employed techniques over a set of scanned real-world volumes. We showed, both numerically and visually, the advantages of the proposed thresholding method over previous Tucker compression schemes in terms of compression accuracy.

Acknowledgements We acknowledge the Computer-Assisted Paleoanthropology group and the Visualization and MultiMedia Lab at University of Zurich for the acquisition of the Wood micro-CT dataset.

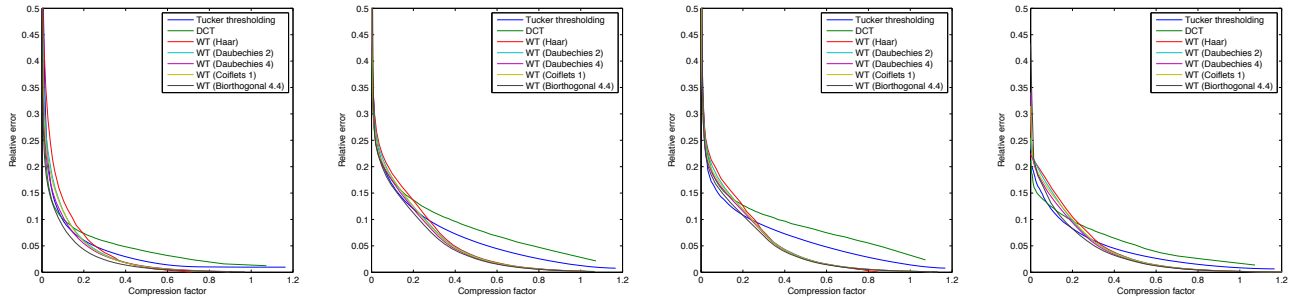
References

1. Real world medical datasets. http://volvis.org/ (2014)
2. Andrews, H.C., Patterson C., I.: Singular value decomposition (SVD) image coding. Communications, IEEE Transactions on **24**(4), 425–432 (1976)
3. Bader, B.W., Kolda, T.G., et al.: MATLAB tensor toolbox version 2.5. Available online (2012). URL http://www.sandia.gov/ tgkolda/TensorToolbox/
4. Ballester-Ripoll, R., Suter, S.K., Pajarola, R.: Analysis of tensor approximation for compression-domain volume visualization. Computers & Graphics **47**, 34–47 (2015)
5. Bilgili, A., Öztürk, A., Kurt, M.: A general BRDF representation based on tensor decomposition. Computer Graphics Forum **30**(8), 2427–2439 (2011)
6. Carroll, J.D., Chang, J.J.: Analysis of individual differences in multidimensional scaling via an n-way generalization of "Eckart–Young" decompositions. Psychometrika **35**(3), 283–319 (1970)



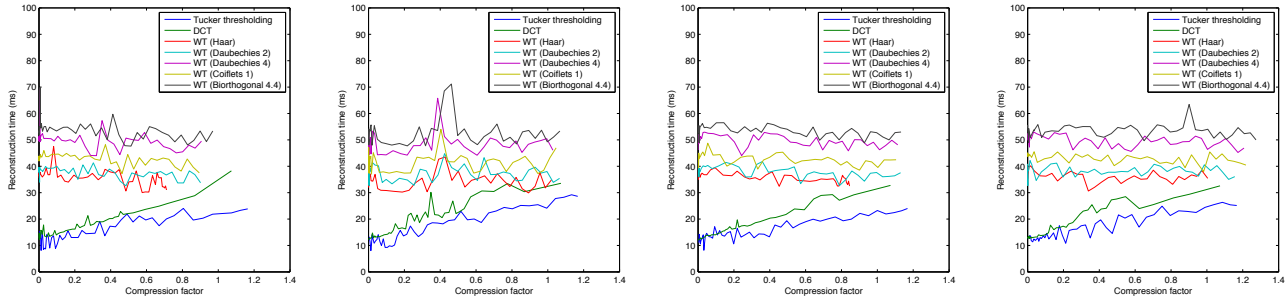
a) Original b) Error (symmetric truncation) c) Error (non-symmetric truncation) d) Error (thresholding)

Fig. 12 a) Original 256^3 Bonsai, Foot, Skull and Wood volumes. b-d) Errors expressed as the absolute value of the difference between the original and its approximation. The compression factor F is 0.25 and we employ global quantization.



(a) Bonsai (b) Foot (c) Skull (d) Wood

Fig. 13 Compression quality comparison between Tucker core thresholding (global quantization) and other methods (DCT compression and several wavelet transforms), applied on a data brick of size 64^3 lying in the center of each of the 4 data sets. The proposed method is competitive or even the best for compression factors 0.3 and lower, which is the more important target range for most lossy compression applications.



(a) Bonsai (b) Foot (c) Skull (d) Wood

Fig. 14 Reconstruction times for Tucker core thresholding (global quantization) and other methods, applied on a data brick of size 64^3 lying in the center of each of the 4 data sets. Tensor reconstruction is faster, especially for higher compression rates.

7. Chan, T.F., Zhou, H.M.: Total variation improved wavelet thresholding in image compression. In: ICIP, pp. 391–394 (2000)
8. Chen, H., Lei, W., Zhou, S., Zhang, Y.: An optimal-truncation-based tucker decomposition method for hyperspectral image compression. In: IGARSS, pp. 4090–4093 (2012)
9. Gandy, S., Recht, B., Yamada, I.: Tensor completion and low-n-rank tensor recovery via convex optimization. Inverse Problems **27**(2), 025010 (2011)
10. Hackbusch, W.: Tensor spaces and numerical tensor calculus, *Springer series in computational mathematics*, vol. 42. Springer, Heidelberg (2012). DOI 10.1007/978-3-642-28027-6
11. International Organization for Standardization: ISO/IEC 10918-1:1994: Information technology — Digital compression and coding of continuous-tone still images: Requirements and guidelines. International Organization for Standardization, Geneva, Switzerland (1994)
12. Kolda, T.G., Bader, B.W.: Tensor decompositions and applications. SIAM Review **51**(3), 455–500 (2009)
13. Kressner, D., Steinlechner, M., Vandereycken, B.: Low-rank tensor completion by Riemannian optimization. BIT Numerical Mathematics **54**(2), 447–468 (2014)
14. Kurt, M., Öztürk, A., Peers, P.: A compact tucker-based factorization model for heterogeneous subsurface scattering. In: Proceedings of the 11th Theory and Practice of Computer Graphics, TPCG '13, pp. 85–92 (2013)
15. de Lathauwer, L., de Moor, B., Vandewalle, J.: A multilinear singular value decomposition. SIAM Journal of Matrix Analysis and Applications **21**(4), 1253–1278 (2000)
16. 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), 1324–1342 (2000)
17. Pajarola, R., Suter, S.K., Ruiters, R.: Tensor approximation in visualization and computer graphics. In: Eurographics 2013 - Tutorials, t6. Eurographics Association, Girona, Spain (2013)
18. Rajwade, A., Rangarajan, A., Banerjee, A.: Image denoising using the higher order singular value decomposition. IEEE Trans. Pattern Anal. Mach. Intell. **35**(4), 849–862 (2013)
19. Rövid, A., Rudas, I.J., Sergyán, S., Szeidl, L.: Hosvd based image processing techniques. In: Proceedings of the 10th WSEAS International Conference on Artificial Intelligence, Knowledge Engineering and Data Bases, AIKED'11, pp. 297–302. World Scientific and Engineering Academy and Society (WSEAS), Stevens Point, Wisconsin, USA (2011)
20. Sun, X., Zhou, K., Chen, Y., Lin, S., Shi, J., Guo, B.: Interactive relighting with dynamic brdfs. ACM Trans. Graph. **26**(3) (2007)
21. Suter, S.K., Iglesias Guitián, 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), 2135–2143 (2011)
22. Suter, S.K., Makhynia, M., Pajarola, R.: TAMRESH: Tensor approximation multiresolution hierarchy for interactive volume visualization. Computer Graphics Forum **32**(3), 151–160 (2013)
23. Suter, S.K., Zollikofer, C.P., Pajarola, R.: Application of tensor approximation to multiscale volume feature representations. In: Proceedings Vision, Modeling and Visualization, pp. 203–210 (2010)
24. Tan, H., Cheng, B., Feng, J., Feng, G., Wang, W., Zhang, Y.J.: Low-n-rank tensor recovery based on multi-linear augmented lagrange multiplier method. Neurocomputing **119**(0), 144 – 152 (2013). Intelligent Processing Techniques for Semantic-based Image and Video Retrieval
25. Treib, M., Bürger, K., Reichl, F., Meneveau, C., Szalay, A., Westermann, R.: Turbulence visualization at the terascale on desktop PCs. IEEE Transactions on Visualization and Computer Graphics (Proc. Scientific Visualization 2012) **18**(12), 2169–2177 (2012)
26. 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), 967–976 (2006)
27. Tsai, Y.T., Shih, Z.C.: K-clustered tensor approximation: A sparse multilinear model for real-time rendering. ACM Transactions on Graphics **31**(3) (2012)
28. Vannieuwenhoven, N., Vandebril, R., Meerbergen, K.: A new truncation strategy for the higher-order singular value decomposition. SIAM J. Scientific Computing **34**(2), 1027–1052 (2012)
29. Vasilescu, M.A.O., Terzopoulos, D.: TensorTextures: Multilinear image-based rendering. ACM Transactions on Graphics **23**(3), 336–342 (2004)
30. Wang, H., Ahuja, N.: Rank-R approximation of tensors: Using image-as-matrix representation. In: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pp. 346–353 (2005)
31. 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), 527–535 (2005)
32. Wu, Q., Chen, C., Yu, Y.: Wavelet-based hybrid multilinear models for multidimensional image approximation. In: Proceedings IEEE International Conference on Image Processing, pp. 2828–2831 (2008)
33. 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), 186–199 (2008)
34. Wu, Q., Xia, T., Yu, Y.: Hierarchical tensor approximation of multidimensional images. In: Proceedings of the IEEE International Conference in Image Processing, vol. 4, pp. IV–49–IV–52 (2007)

35. Zaid, A., Olivier, C., Alata, O., Marmoiton, F.: Transform image coding with global thresholding: application to baseline jpeg. In: Proceedings of the XIV Brazilian Symposium on Computer Graphics and Image Processing, pp. 164–171 (2001)