Pacific Graphics (2016)
E. Grinspun, B. Bickel, and Y. Dobashi (Editors)

Short Paper

Compressing Bidirectional Texture Functions via Tensor Train Decomposition

R. Ballester-Ripoll[1] and R. Pajarola[1]

[1]Department of Informatics, University of Zürich, Switzerland

Abstract

Material reflectance properties play a central role in photorealistic rendering. Bidirectional texture functions (BTFs) can faithfully represent these complex properties, but their inherent high dimensionality (texture coordinates, color channels, view and illumination spatial directions) requires many coefficients to encode. Numerous algorithms based on tensor decomposition have been proposed for efficient compression of multidimensional BTF arrays, however, these prior methods still grow exponentially in size with the number of dimensions. We tackle the BTF compression problem with a different model, the tensor train *(TT) decomposition. The main difference is that TT compression scales linearly with the input dimensionality and is thus much better suited for high-dimensional data tensors. Furthermore, it allows faster random-access texel reconstruction than the previous Tucker-based approaches. We demonstrate the performance benefits of the TT decomposition in terms of accuracy and visual appearance, compression rate and reconstruction speed.*

1. Introduction

Bidirectional reflectance distribution functions (BRDFs), bidirectional texture functions (BTFs) and time-varying light fields (TVLFs), among others, are used to accurately simulate material reflectance properties for realistic rendering. One way to represent such functions is to model them analytically, which is very space-efficient but relies on assumptions and prior knowledge of the material's physical properties. Data-driven rendering techniques, on the other hand, prioritize realism over storage requirements by making use of precomputed multidimensional tables. During rendering, the appropriate reflectance at each texel is read and interpolated from the table to produce high-quality results. Such tables are populated empirically: physical conditions (e.g. camera and light positions) are varied in a laboratory in order to measure each entry. The resulting data sets are typically very redundant and large in size, due to their high intrinsic dimensionality and often high sampling density. The need to cope with this challenge has sparked numerous preprocessing algorithms that rely on compressing the input down to a reduced representation while preserving main texture details. The processing pipeline is highly asymmetric: compression can be done in a slow offline step, whereas an efficient online decompression stage is critical. Attention must be paid to a) small resulting sizes, which allow for in-core solutions; and b) fast decompression.

Tensor decomposition constitutes a family of successful approaches for light field and BTF compression. Upon acquisition, the full precomputed data table is treated as an N-dimensional tensor (i.e. a scalar field discretized as a multiarray) and compressed by a multidimensional generalization of the singular value decomposition (SVD). Several such generalizations exist and have been proven viable for real-time photorealistic rendering, as summarized in the following section. In this paper we solve this problem for the first time with the *tensor train* (TT), a more recent decomposition that was specifically developed for spaces with a high number of dimensions. As opposed to other models, TT has not received attention from the graphics community yet and in particular has not been applied to data-driven rendering. We investigate the performance of TT-based BTF compression and its advantages with respect to its prior state-of-the-art counterparts.

2. Related Work

N-SVD compression The N-mode singular value decomposition (N-SVD), also known as Tucker decomposition, relies on *multilinear projections*: a set of separable orthogonal basis functions is defined onto which the input is projected. The higher-order orthogonal iteration (HOOI) is an algorithm to generate an N-SVD decomposition by successively computing the leading left singular vectors along each mode [dLdMV00]. This yields a dense core of size R^N (where R is the number of *tensor ranks*) and an orthonormal basis in the form of N *factor matrices*. The discrete cosine and Fourier transforms, as well as the separable orthogonal wavelet transforms, can be viewed as particular cases of N-SVD that use predefined factors [BRP15]. N-SVD is often used as a reference algorithm for comparison against other tensor-based methods.

Tensor-based data-driven rendering Several compression approaches that pursue more structured multilinear projections (such

DOI: 10.2312/pg.20161329

as sparse or clustered representations) have been described in the literature, often based on PCA and the N-SVD, with early works including [VT04], [HWL05] and [WWS*05]. Ruiters and Klein [RK09] propose a sparse multilinear decomposition achieved through a sequence of K-SVD dictionary learning steps that yields better compression than both N-SVD and clustered principal components analysis (PCA). Wu et al. [WXC*08] define a hierarchical Tucker decomposition by recursive approximation of the residuals; their method compares favorably against wavelets and N-SVD for BTF compression. Bilgili et al. [BÖK11] and Ruiters et al. [RSK12] tackle BRDF compression using N-SVD and the CANDECOMP/PARAFAC model (CP) respectively. Tsai and Shih [TS12] construct a K-clustered tensor approximation (K-CTA): the tensor is split along a mode and each part is assigned to a cluster; clusters are then compressed independently via N-SVD. This yields a better compression rate and reconstruction speed compared to the raw N-SVD. The most recent contribution in this direction, MK-CTA, is due to Tsai [Tsa15] and is a multiway clustering extension of K-CTA. By taking into account all modes for defining clusters (instead of only one), this version leads to substantial improvements over K-CTA, also when handling time-varying light fields. Although clustering strategies can significantly enhance BTF compression rates, N-mode projections still suffer from the curse of dimensionality in the sense that the resulting cores are dense and still keep the same number of dimensions. This motivates using a different decomposition type as we argue next.

3. Tensor Train BTF Decomposition

The *tensor train* (TT) representation was proposed by Oseledets [Ose11] to better overcome the curse of dimensionality. Its number of coefficients, $O(NIR^2)$, increases linearly with respect to the number of dimensions N (where I is the data tensor size along each dimension). This is in contrast with previously used models; for example, the pure N-SVD requires $O(R^N + NIR)$ coefficients. For visual data and $N \geq 4$, TT quickly outperforms it in terms of compression accuracy. TT has an additional, particularly important advantage in the context of this paper. In rendering applications, typically only a small percentage of the total BTF tensor elements need to be reconstructed at each frame via random accesses. With an N-SVD decomposition one must always traverse all R^N core elements, no matter if the whole tensor or a single value are being reconstructed. On the other hand, subspace reconstructions in the TT format operate only on one or a few slices per core and are thus comparatively inexpensive (Fig. 1, see also [Ose11]). In particular, retrieving color components from a single random-access texel needs only $O(NR^2)$ operations (see also Sec. 3.2).

Computing the TT Decomposition The Frobenius norm of a tensor $\|\mathscr{A}\|_F$, that we denote just as $\|\mathscr{A}\|$, is the norm of its vectorized representation: $\sqrt{\sum_i \mathscr{A}(i)^2}$. To measure the quality of an approximated tensor $\widetilde{\mathscr{A}}$ with respect to the original $\mathscr{A}$ we use the *relative error*, defined as $\|\widetilde{\mathscr{A}} - \mathscr{A}\|/\|\mathscr{A}\|$. Given a prescribed tolerable error $\varepsilon_{\max}$ and a tensor $\mathscr{A}$, the so-called *TT-SVD* algorithm computes its TT decomposition by successive *tensor unfoldings* interleaved with SVD low-rank truncation steps. TT ranks are determined adaptively and are equal to these SVD ranks; the algorithm guarantees that the resulting ε will not exceed the target $\varepsilon_{\max}$, and in fact it is usually lower. One key feature of TT-SVD (compared to N-SVD decomposition procedures) is that dimensions are separated one at a time and the SVD truncations are performed on increasingly smaller matrices. For a detailed description of TT-SVD and its theoretical error bounds we refer the reader to [Ose11].

3.1. BTF Compression

Since view and light coordinates each lie on a 2D manifold, a BTF can be regarded as a 7-dimensional tensor: texture coordinates are given by (x,y), color channel by c, view direction by (v_x,v_y) and light direction by (l_x,l_y) (alternatively, one may use angular coordinates). In the data sets used in our experiments, however, view and light coordinates were not sampled following a 2D parameterization and are given in a sequential order. In other words, the dimensions (v_x,v_y) are given as a single dimension v, and equivalently for (l_x,l_y) into l. We do not separate v and l back into 2D, as this would require resampling and introduce additional inaccuracies.

Dimension ordering The TT format is dimensionally asymmetric: the initial ordering of dimensions can affect the number of ranks that are needed to achieve the user-defined accuracy. This property is distinct to N-SVD, whose accuracy is invariant to mode permutation. Since TT ranks tend to be larger around central dimensions (see Tab. 1), and each core $\mathscr{G}_n$ has size $R_n \times I_n \times R_{n+1}$, it is generally best to place large dimensions in the middle in order to minimize the final size. Our experiments indicate that the ordering (x,y,c,v,l) generally performs well for BTF compression.

3.2. BTF Decompression

Notation-wise we assume that $1 \leq x \leq X, 1 \leq y \leq Y$, $C = 3$ is the number of color channels, and $1 \leq v \leq V, 1 \leq l \leq L$. To reconstruct one element from a TT decomposition it is sufficient to perform a sequence of matrix-matrix multiplications. The first and last matrices are a row and a column vector, and the product is thus a scalar in the end. This also means that only vector-matrix products have to be computed throughout the whole sequence, for a total of $\sum_{i=1}^{4}(R_i - 1)R_{i+1}$ and $\sum_{i=1}^{4} R_i R_{i+1}$ floating-point sums and products, respectively, per reconstructed element. The symbols $R_1,\ldots,R_5$ denote TT ranks and are different in general. The formula to retrieve all color channels for one texel is:

$$\sum_{\alpha_1,\alpha_2,\alpha_3,\alpha_4} \mathscr{G}_1(x,\alpha_1)\mathscr{G}_2(\alpha_1,y,\alpha_2)\mathscr{G}_3(\alpha_2,:,\alpha_3)\mathscr{G}_4(\alpha_3,v,\alpha_4)\mathscr{G}_5(\alpha_4,l)$$

where $1 \leq \alpha_n \leq I_n$. The *tensor cores* $\mathscr{G}_i$ are illustrated in Fig. 1.

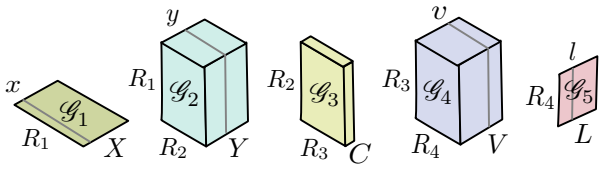

Figure 1: TT compression of a 5D BTF, with the smaller color channel dimension $C = 3$ in the center. The vectors and matrices shadowed in gray are used to reconstruct a texture element (x,y) from view v and light l.

4. Results

We have tested the proposed TT-based compression for BTFs on the UBO2003 data set [SSK03], which comprises 6 different materials. Each of them is organized as a 5-dimensional tensor consisting of 256×256 texels acquired over 81 view and 81 light conditions. They are 8-bit RGB and amount to 1.2GB each. Our benchmark tests were written in MATLAB, and use:

- the Tensor Toolbox [BK*15] for the N-SVD compression. It implements the popular HOOI algorithm, that we use for the decomposition stage;
- the TT-Toolbox [O*] for TT, which provides an implementation of the TT-SVD compression algorithm as well as reconstruction procedures for arbitrary subspaces of TT tensors.

Tab. 1 shows the numerical results of the proposed method compared to N-SVD, which is often used as a simple baseline algorithm to compare to. TT is tested twice: (1) aiming for the same accuracy as with N-SVD, and (2) aiming for the same compression rate. For the N-SVD decomposition we found that 3 iterations of the HOOI were sufficient to converge and we initialized its factor matrices via the higher-order singular value decomposition (HOSVD) [dLdMV00]. All coefficients of both methods were stored as 16-bit floating point numbers as in [Tsa15]. The number of operations for all methods (second-to-last column) was computed accounting for both sums and products. Example reconstructed slices $\mathscr{A}(:,:,:,V/2,L/2)$ are shown in Fig. 2 for visual comparison between both algorithms over all BTF data sets. The images correspond to the results from TT (2) in Tab. 1.

We can make a relative analysis of our (TT) method as it compares to the most recent state-of-the-art algorithm MK-CTA [Tsa15] by relating both to the N-SVD baseline. For MK-CTA, a similar or moderately higher quality compared to N-SVD was reported when compression rates are set equal for both methods. In terms of speed, MK-CTA takes longer to decompose than N-SVD (as TT does), but is faster than N-SVD for reconstruction by a factor up to 1:5 [Tsa15]. In our experiments, the TT approach achieves similarly better quality compared to N-SVD for the same compression rates (by up to 5db PSNR), as shown for TT (2) in Tab. 1. Furthermore, TT's major advantage is its reconstruction time, with speed-up factors that reach up to 1:25, which is significantly higher than the relative reported speed-up of 1:5 of MK-CTA. Our measurements are consistent with the expected number of operations. The advantage stems from the fact that, even though TT often requires more ranks than N-SVD, only a few slices of the compressed BTF need to be accessed for a random-access decompression (as argued in Sec. 3.2).

5. Conclusions

We have studied and demonstrated the performance of BTF compression using the TT decomposition, a model not exploited in computer graphics as of yet. It offers a competitive compression quality and is simple to implement: decomposition is a straightforward application of the TT-SVD algorithm (paying attention to the dimension ordering), while random-access texture reconstruction follows from plain vector-matrix products. Another significant advantage is its reconstruction time, which we both estimated theoretically (asymptotic complexity and number of operations) and measured experimentally. As with other tensor decompositions, parallel and GPU solutions are possible for the TT reconstruction stage since only basic linear algebra operations are performed. Even though we only tested 5D compression in this paper, larger advantages may be expected for higher-dimensional tensors, e.g. time-varying data, reflectance fields, or BTFs whose view and light directions are parameterized into 2-dimensions.

References

[BK*15] BADER B. W., KOLDA T. G., ET AL.: MATLAB Tensor Toolbox version 2.6. Available online, http://www.sandia.gov/~tgkolda/TensorToolbox/, February 2015. 3

[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

[BRP15] BALLESTER-RIPOLL R., PAJAROLA R.: Lossy volume compression using Tucker truncation and thresholding. *The Visual Computer* (2015), 1–14. 1

[dLdMV00] DE LATHAUWER L., DE MOOR B., VANDEWALLE J.: On the best rank-1 and rank-$(R_1, R_2, ..., R_N)$ approximation of higher-order tensors. *SIAM Journal of Matrix Analysis and Applications 21*, 4 (2000), 1324–1342. 1, 3

[HWL05] HO P.-M., WONG T.-T., LEUNG C.-S.: Compressing the illumination-adjustable images with principal component analysis. *IEEE Transactions on Circuits and Systems for Video Technology 15*, 3 (2005), 355–364. 2

[O*] OSELEDETS I. V., ET AL.: Tensor Train Toolbox. Github repository, https://github.com/oseledets/TT-Toolbox/. 3

[Ose11] OSELEDETS I. V.: Tensor-train decomposition. *SIAM Journal on Scientific Computing 33*, 5 (2011), 2295–2317. 2

[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

[SSK03] SATTLER M., SARLETTE R., KLEIN R.: Efficient and realistic visualization of cloth. In *Proceedings Eurographics Workshop on Rendering* (Jun 2003), pp. 167–177. 3

[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 (June 2012), 1–17. 2

[Tsa15] TSAI Y.-T.: Multiway K-clustered tensor approximation: Toward high-performance photorealistic data-driven rendering. *ACM Transactions on Graphics 34*, 5 (October 2015), 157:1–15. 2, 3

[VT04] VASILESCU M. A. O., TERZOPOULOS D.: TensorTextures: multilinear image-based rendering. *ACM Transactions on Graphics 23*, 3 (2004), 336–342. 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 (July 2005), 527–535. 2

[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 (January/February 2008), 186–199. 2

BTF name	Algorithm	Space (MB)	Tensor ranks	T_D (s)	T_R (ms)	Operations	PSNR (dB)
Corduroy	**N-SVD**	47.6	96, 96, 3, 30, 30	224.3291	6.8820	50028537	27.6246
	TT (1)	15.4	61, 488, 131, 23	419.1781	0.1195	450467	27.7397
	TT (2)	49.0	76, 1173, 441, 36	936.6653	0.2519	3318798	29.7186
Impalla	**N-SVD**	14.0	71, 71, 3, 22, 22	166.9500	2.0003	14743781	23.4342
	TT (1)	1.0	41, 44, 19, 38	196.6988	0.0580	10281	23.6490
	TT (2)	14.6	76, 345, 143, 67	894.1605	0.1282	369260	28.9794
Proposte	**N-SVD**	44.6	93, 93, 3, 30, 30	207.5554	4.9417	46958637	26.3326
	TT (1)	5.1	55, 177, 62, 28	379.7756	0.1097	89468	26.5019
	TT (2)	43.4	89, 867, 427, 54	947.7406	0.2092	2426798	30.9604
Pulli	**N-SVD**	86.5	111, 111, 3, 35, 35	258.0855	9.2653	90971227	27.2433
	TT (1)	15.0	51, 535, 266, 21	357.2561	0.1662	922782	27.4470
	TT (2)	85.9	84, 1690, 1004, 44	940.3235	1.1698	10565189	31.8687
Wallpaper	**N-SVD**	16.7	74, 74, 3, 23, 23	159.1643	2.4571	17500030	28.0912
	TT (1)	1.1	60, 36, 13, 27	402.9035	0.0542	7967	28.2279
	TT (2)	17.2	124, 269, 79, 57	1004.0195	0.1092	204052	32.2044
Wool	**N-SVD**	21.8	78, 78, 3, 25, 25	183.0543	2.7699	22963322	27.6590
	TT (1)	14.9	62, 466, 141, 18	503.2596	0.1218	458595	27.7264
	TT (2)	21.7	67, 618, 205, 23	652.5150	0.1362	854625	28.3385

Table 1: Experimental results comparing N-SVD with the proposed TT-based BTF compression, tested over all 6 BTFs from the UBO2003 database. For the rows labeled TT (1) we set the PSNR to be approximately that of N-SVD; for the rows labeled TT (2) we set the compression rate to be that of N-SVD's instead. Decomposition time is denoted by T_D. To compute T_R (reconstruction time) we averaged the decompression time of 1000 elements (each with 3 color channels) at random positions in the tensor.

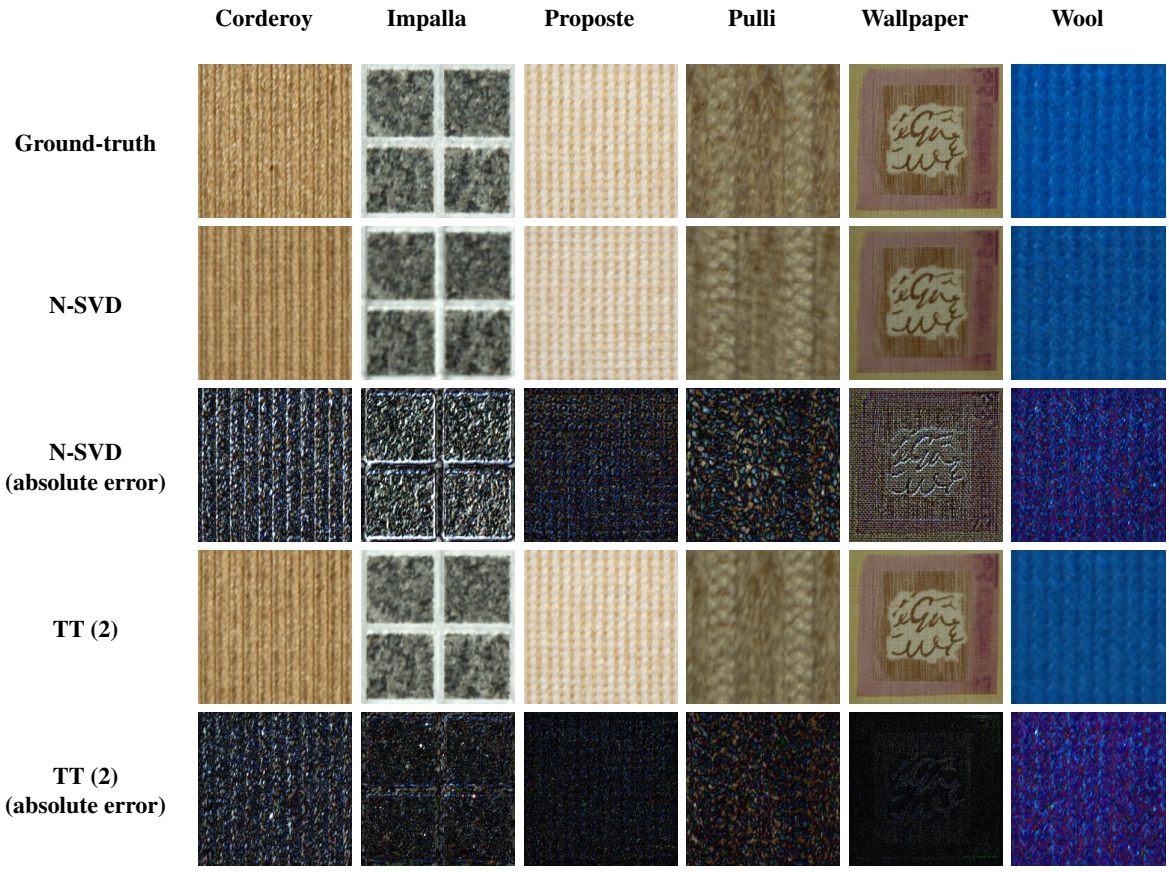

Figure 2: Visual results corresponding to Tab. 1: in each case, the slice lying at $v = V/2$ and $l = L/2$ is shown.