TENSOR VIDEO CODING

Abo Talib Mahfoodh and *Hayder Radha, IEEE Fellow*

Department of Electrical and Computer Engineering,
Michigan State University, East Lansing, MI, USA
Emails: {mahfoodh,radha}@msu.edu

ABSTRACT

In this paper, we exploit the intrinsic tensor-nature of video and propose a Tensor Video Coding (TVC) framework that is based on tensor decomposition. We develop a Progressive Canonical-decomposition Parallel-factor (PCP) framework that is tailored for video representation and coding. Our simulation results show that the proposed TVC approach outperforms state-of-the-art video coding methods that do not rely on motion estimation or compensation.

Index Terms—Tensor Decomposition, Video Coding, CP Tensor Decomposition, Rank-one Tensors

1. INTRODUCTION

In this paper, we propose a low-complexity video representation and coding scheme based on tensor factorization. We refer to our proposed framework, which does not rely on any form of Motion Estimation (ME) or Motion Compensation (MC), as Tensor Video Coding (TVC). The proposed TVC can be targeted for applications and devices that tolerate delay but require low-complexity at both the encoder and decoder. We show that TVC outperforms state-of-the-art video coding schemes that do not employ ME. Among such video coding schemes are Motion JPEG2000 [1] and H.264/AVC-Intra [2][3], both of which are based on coding each frame without exploiting temporal redundancy. H.264/AVC-no motion [2][4] codes a Group Of Pictures (GOP) such that the first frame is coded as a reference (i.e. I frame) and the other frames in the same GOP are coded as predicted pictures. This latter coding scheme exploits the temporal redundancy without employing any ME. Another set of low complexity video coding schemes are based on dimensionality reduction by means of tensor decomposition. 2DSVD video coding [5] is one of the recent developed frameworks in this family. Furthermore, [6–12] employed tensor factorization to represent visual data such as dynamic textures and image ensembles.

Tensor factorization, which is a generalization of matrix SVD decomposition, has received a great deal of attention recently [13–18]. There are two main tensor decomposition methods. The first one is known as Higher Order SVD (HOSVD) or Tucker decomposition [16]. The second decomposition, and arguably more popular, is known as Canonical-decomposition Parallel-factor (CP) method. This decomposition factorizes a tensor onto a number of rank-one tensors [16]. For more information on these and other tensor decomposition methods we refer to [13–18].

A key objective of the proposed TVC framework is to develop a CP-based tensor decomposition to represent and code video efficiently. However, the standard CP decomposition does not lead to an efficient representation of natural video. Consequently, we propose a new CP-based tensor decomposition, which we refer to as Progressive CP (PCP), and which lends itself to efficient video representation. The proposed PCP pursues a decomposition that leads to a small number of rank-one tensors. Similar to CP, the PCP rank-one tensors are represented using a set of 1D vectors. We refer to these 1D vectors as *eigenfibers*. (For tensors, a *fiber* is a generalization of row-wise or column-wise vector in matrices.) The proposed TVC framework represents and codes a video with a set of eigenfibers as explained in detail later in this paper. To the best of our knowledge, this is the first general coding framework based on employing rank-one tensor decomposition. Prior methods were employed only for dynamic textures or for very low-rank videos that have virtually identical or similar fibers in all spatiotemporal directions[6–8], [11], [12].

After a brief review of CP, Section 2 describes the proposed PCP approach. Section 3 presents our TVC framework, and Section 4 presents our simulation results.

2. CP-BASED TENSOR VIDEO DECOMPOSITION

2.1. CP Factorization

A grayscale video is a 3D tensor $\boldsymbol{\chi} \in \mathbb{R}^{v_1 \times v_2 \times v_3}$, where (v_1, v_2) are the spatial and (v_3) temporal dimensions. CP decomposes a 3D tensor $\boldsymbol{\chi}$ onto a number of rank-one tensors, each of which can be written as an outer product of three vectors [16]:

$$\hat{\boldsymbol{\chi}} = \sum_{r=1}^{R} \lambda_r \left(a_r^{(1)} \circ a_r^{(2)} \circ a_r^{(3)}\right) \quad (1)$$

where $\circ$ is an outer product, and λ_r is a normalization value that maintains an ℓ_2 unit norm for the vectors $a_r^{(d)}$, $d \in \{1,2,3\}$ [16]. Hence, the tensor $\boldsymbol{\chi}$ is approximated using a linear combination of rank-1 tensors $\left(a_r^{(1)} \circ a_r^{(2)} \circ a_r^{(3)}\right)$; and

the rank parameter R is the number of rank-one tensors used to approximate $\boldsymbol{\chi}$. Here, $a_r^{(d)}$ are column vectors of a corresponding set of matrices (i.e. $A^{(d)}$ where $d = 1,2,3$). For example, $A^{(1)} = [a_1^{(1)}\ a_2^{(1)} \dots a_R^{(1)}]$ is a $v_1 \times R$ matrix. In general, $A^{(d)} \in \mathbb{R}^{v_d \times R}$. These matrices can be found using:

$$A^{*(d)} = \underset{A^{(d)}}{\operatorname{argmin}} \left\| X_{(d)} - A^{(d)} \left(A^{(d_1)} \odot A^{(d_2)} \right)^T \right\|_F \quad (2)$$

Here, $\odot$ is Khatri-Rao product, $d \in \{1,2,3\}$, $d_1 \in \{1,2,3\} - \{d\}$, and $d_2 \in \{1,2,3\} - \{d, d_1\}$. $X_{(d)}$ is a matrix that results from unfolding the tensor $\boldsymbol{\chi}$ with respect to the d^{th} dimension. For example, $X_{(1)} \in \mathbb{R}^{v_1 \times (v_2 v_3)}$ is a matrix that results from unfolding the original tensor $\boldsymbol{\chi}$ with respect to the first dimension (i.e. v_1). Similarly, $X_{(2)}$ and $X_{(3)}$ are the unfolded original tensors with respect to the second (v_2) and third (v_3) dimensions [16]. For a given rank parameter R, the Alternative Least Square (ALS) [19] approach can be used to solve for the set of matrices in (2). It solves for $A^{(1)}$ by fixing $A^{(2)}$ and $A^{(3)}$ and similarly for $A^{(2)}$ and $A^{(3)}$:

$$A^{*(d)} = X_{(d)} \left(\mathcal{Z}^{(d)} \right)^T \left(\left(\mathcal{Z}^{(d)} \right)^T \left(\mathcal{Z}^{(d)} \right) \right)^{-1} \quad (3)$$

where $\mathcal{Z}^{(d)} = A^{(d_1)} \odot A^{(d_2)}$.

2.3. Progressive CP for Video Decomposition

A straightforward approach for tensor-based video representation is to directly employ the CP decomposition onto an original video sequence. However, such straightforward approach does not result in an efficient coding method (as we explain further below). Furthermore, independent of the form of tensor decomposition used, the rank parameter R should not be fixed throughout the tensor decomposition of the entire video tensor. In particular, the value of R directly influences the rate and efficiency of the video representation since it determines the number of rank-one tensors used for this representation. Meanwhile, different parts of the video tensor have different levels of spatial and temporal details. In our proposed CP-based solution we apply different tensor decompositions/ranks to different *3D blocks* of the original tensor signal. Hence, for the remainder of the paper, a 3D tensor $\boldsymbol{\chi}$ represents a 3D video-block. It is a cubic portion of a video that covers the two spatial plus temporal dimensions.

Now, we highlight some of the key characteristics of the proposed PCP-based decomposition. Unlike traditional CP, the proposed PCP has an extra loop to solve for each rank-one tensor progressively. In principle, PCP tries to estimate a given 3D video-block with a rank-one tensor. If the error is higher than a threshold ϵ_{max}, it will add another rank-one tensor to estimate the residual. This will continue until the error becomes smaller than ϵ_{max} or it reaches some maximum allowable rank R_{max}. PCP is different from traditional CP in that the later estimates all rank-one tensors at once, while PCP estimates them individually in a progressive manner. This represents a crucial and fundamental difference between the two frameworks.

Similar to CP, the approximated tensor ($\widehat{\boldsymbol{\chi}}$) under PCP is a sum of rank-one tensors. However, the PCP decomposition results in different rank-one tensors and corresponding vectors from what is generated by CP. We also use a different normalization as explained further below. To emphasize the difference between the two schemes, we express the PCP decomposition using different notations for the rank-one tensors and normalization parameters:

$$\widehat{\boldsymbol{\chi}} = \sum_{r=1}^{R} \beta_r (b_r^{(1)} \circ b_r^{(2)} \circ b_r^{(3)}) \quad (4)$$

where, under PCP, $R \in \{1,2, \dots, R_{max}\}$. Similar to CP, $b_r^{(d)}$ are column vectors of a corresponding set of matrices $B^{(d)}$ where $d = 1,2,3$. Under PCP though, there are two key differences:

(1) We solve for the vectors $b_r^{(d)}$ using:

$$b_r^{*(d)} = \underset{b_r^{(d)}}{\operatorname{argmin}} \left\| \left(X_{(d)} - \sum_{k=0}^{r-1} X'_{(d),k} \right) - b_r^{(d)} \left(z_r^{(d)} \right)^T \right\|_F \quad (5)$$

where $z_r^{(d)} = b_r^{(d_1)} \odot b_r^{(d_2)}$, $d \in \{1,2,3\}$, $d_1 \in \{1,2,3\} - \{d\}$, and $d_2 \in \{1,2,3\} - \{d, d_1\}$. $X'_{(d),k}$ is the k^{th} rank-one tensor unfolded over dimension d, and $k \in \{1,2, \dots, R\}$. $X'_{(d),0} = 0$, $X'_{(d),k} = \beta_k b_k^{(d)} \left(z_k^{(d)} \right)^T$. For a given rank parameter R_{max}, we modify the ALS approach to solve the minimization problem in (5). Similar to CP, we fix $b_r^{(2)}$ and $b_r^{(3)}$ and solve for $b_r^{(1)}$; and similarly for $b_r^{(2)}$ and $b_r^{(3)}$:

$$b_r^{*(d)} = \left(X_{(d)} - \sum_{k=0}^{r-1} X'_{(d),k} \right) \left(z_r^{(d)} \right)^T \left(\left(z_r^{(d)} \right)^T z_r^{(d)} \right)^{-1} \quad (6)$$

At each iteration r we calculate the error $\epsilon_r = MSE(\boldsymbol{\chi} - \widehat{\boldsymbol{\chi}_r})$, where $\widehat{\boldsymbol{\chi}_r}$ is obtained from (4). If the error is larger than the maximum acceptable value (i.e. $\epsilon > \epsilon_{max}$), then another rank-one tensor will be added to approximate the residual $\boldsymbol{\chi} - \widehat{\boldsymbol{\chi}_r}$ in the next iteration. This approximation results in a progressive decomposition of $\boldsymbol{\chi}$. The stop criterion is $\epsilon \leq \epsilon_{max}$ or $r > R_{max}$. This procedure aims at minimizing the global MSE while keeping it smaller than ϵ_{max} by forcing the condition to each 3D block; and meanwhile we are constraining the maximum rank for each 3D block by employing R_{max}. The later constraint effectively keeps the overall rate below a certain level. A more elaborate rate-control solution will be presented in a future paper.

(2) As mentioned above, under CP, the rank-one tensors are normalized by maintaining an ℓ_2 unit-norm vectors. This is captured through λ_r in (1). Under PCP, we employ an ℓ_∞ norm instead. This leads to the following: (a) the normalizing parameter β_r captures the maximum magnitudes of the entries of the corresponding vectors $b_r^{(d)}$, $d = 1,2,3$; and (b) the vectors $b_r^{(d)}$ have normalized values between -1 and $+1$.

As we show below, the proposed PCP lends itself to more efficient video coding when compared to traditional CP.

3. TENSOR VIDEO CODING WITH PCP

We refer to the normalized vectors $b_r^{(d)}$ resulting from the proposed PCP decomposition as *eigenfibers*. Once evaluated for all 3D blocks of a video, we can arrange these eigenfibers onto the columns of a 2D matrix B. Thus, we can apply a 2D compression scheme to this matrix B (i.e., treating it as an image); and subsequently, we can decode its columns, which represent the eigenfibers $b_r^{(d)}$, to reconstruct the original video. Two important questions need to be answered: (1) How should the eigenfibers $b_r^{(d)}$ be arranged within the matrix B? And (2) How much correlation does exist among these eigenfibers? To address these questions, we need to introduce an index for the 3D blocks within a video tensor. A given 3D video-block is indexed by (j).

Recall that under PCP each block has its own rank, and hence we denote R_j to represent the number of rank-one tensors used for approximating video-block j. Consequently, the total number of rank-one tensors used for approximating the whole video is: $\sum_{j=1}^{N_B} R_j$, where N_B is the total number of 3D blocks in the video. Since each rank-one tensor requires three eigenfibers: $(b_{r,j}^{(1)} \circ b_{r,j}^{(2)} \circ b_{r,j}^{(3)})$, for $r = 1, \dots R_j$ and $j = 1, \dots N_B$; then we have a total of $3\sum_{j=1}^{N_B} R_j$ eigenfibers to code. There are many options for arranging these eigenfibers $b_{r,j}^{(d)}$ onto the matrix B that we plan to compress as a 2D image. Here, we employ the arrangement shown in the example of Fig. 1. In this example, we generated the eigenfibers of 180 frames of the Container CIF video. We divided the video into 16×16×180 3D blocks, which results into 396 tensor blocks. (For clarity and ease-of-illustration purposes, we are only showing the eigenfibers for the first 72 blocks chosen in a raster-scan order.) All eigenfiber values are mapped from their normalized $[-1, +1]$ range onto the traditional $[0, 255]$ pixel values. As shown in the figure, we employ the following arrangement:

(1) *Vertical arrangement*: The eigenfibers $b_{r,j}^{(1)}$ are put at the top of the 2D image; each fiber is of height 16. Next, the second eigenfibers $b_{r,j}^{(2)}$, also with height 16, are placed below $b_{r,j}^{(1)}$. These two groups of eigenfibers capture the 16×16 spatial information of the video-blocks. Meanwhile, the third eigenfibers $b_{r,j}^{(3)}$ with height 180 are placed below; and these later eigenfibers capture the temporal information of the 3D video-blocks.

(2) *Horizontal arrangement*: More importantly, we separate the eigenfibers associated with the first rank-one tensors (i.e., for $r = 1$) from the rest of all other eigenfibers with higher rank index (i.e. for $r > 1$). This separation is analogous to differentiating between "DC" and "AC" coefficients in traditional image and video coding. For higher rank indices, $r > 1$, we simply place the eigenfibers according to the blocks they belong to in a raster-scan order. We have also experimented with other horizontal arrangements for eigenfibers with $r > 1$.

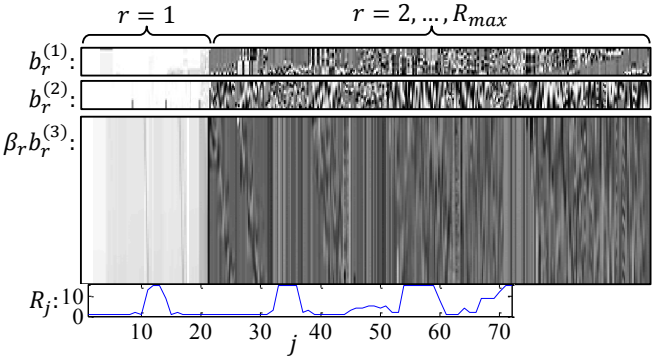

Fig. 1. Eigenfibers of the PCP tensor decomposition and the rank values R for 72 blocks of the Container CIF video with 180 frames.

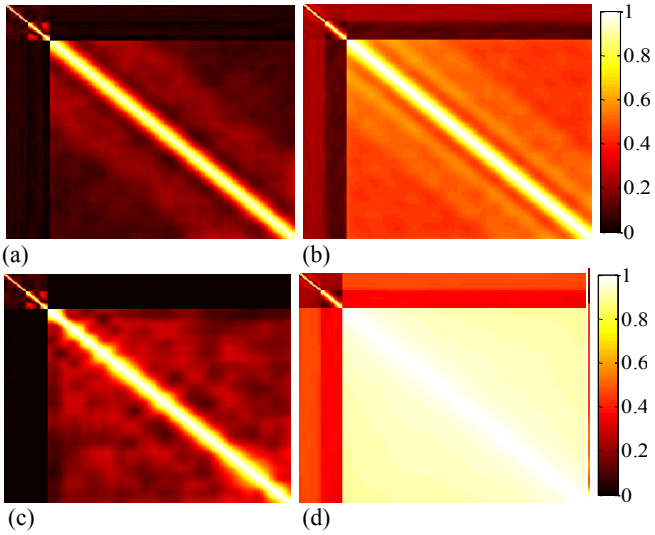

Fig. 2. The autocorrelation among (a) The CP vectors; (b) The proposed PCP eigenfibers; (c) After λ_r absorption onto $A^{(3)}$; and (d) After β_r absorption onto $B^{(3)}$. The video is Container CIF.

For example, one can group eigenfibers with $r = 2$, followed by ones with $r = 3$, and so on. We observed little improvement in coding efficiency while using such arrangement; meanwhile it increases the complexity due to the need of performing matrix permutations at both the encoding and decoding sides.

(3) *Absorbing the normalization parameters* $\beta_{r,j}$: As shown in Fig. 1, the temporal eigenfibers $b_{r,j}^{(3)}$ are multiplied with the corresponding parameters $\beta_{r,j}$. There are two benefits for absorbing these parameters onto the eigenfibers. First, we eliminate the need for coding these parameters separately. Second, this multiplication process improves the correlation among the eigenfibers within the 2D image as we discuss below.

It is important to note that for a given 3D block, the three eigenfibers $(b_r^{(1)}, b_r^{(2)}, b_r^{(3)})$ are expected to be uncorrelated. However, if we consider different 3D video-blocks of similar spatial and temporal characteristics, then we anticipate that the eigenfibers across such blocks to be correlated. These correlated eigenfibers across similar spatiotemporal 3D video-blocks represent a key aspect for

achieving efficient tensor representation and coding of the underlying video. Moreover, the entries within the eigenfibers can also be correlated. In other words, if we consider the first entry of an eigenfiber $b_r^{(d)}$, then it will be desired to have this entry correlated with other entries within the same eigenfiber. Such *intra*-eigenfiber correlation is captured by the correlation among the rows of the eigenfiber matrix shown in Fig.1. The proposed PCP decomposition provides higher intra-eigenfibers correlation (arguably significantly higher) than what can be achieved under CP. Fig. 2 shows the autocorrelation among the entries of factored vectors $(a_{r,j}^{(1)}, a_{r,j}^{(2)}, a_{r,j}^{(3)})$ of the Container video using CP and the autocorrelation among the entries of the eigenfibers $(b_{r,j}^{(1)}, b_{r,j}^{(2)}, b_{r,j}^{(3)})$ under the proposed PCP decompositions. By comparing figures 2(a) and 2(b), it should be clear that PCP provides eigenfibers with higher autocorrelation. Figures 2(c) and 2(d) also show the impact of absorbing the normalization parameters $\lambda_{r,j}$ (for CP) and $\beta_{r,j}$ (for PCP) within the corresponding eigenfibers.

The final step in TVC is to code the rank parameters R_j for all 3D blocks. We simply arrange these values onto a vector and entropy code them in a lossless manner.

4. SIMULATION RESULTS

In our simulations, we simply employ JPEG2000 to compress the eigenfibers' matrix. Other 2D compression methods can be considered in the future. Huffman and run-length coding are applied to compress the rank-values R_j.

We evaluated TVC using more than 10 video test sequences, and we compare its performance with five low complexity encoders described in [5]. Here, we present the results of TVC coding over 180 frames of the Container and Silent CIF videos only (at 30 frames per second) due to space limitation and more importantly to be consistent with [5]. The PSNR plots are shown in Fig. 3. The results of the first five encoders are extracted from [5]. TVC outperforms the other five encoders over a wide range of bitrates, and it results in higher PSNR than the best competing method by up to 2 db. Note that we changed the block size and ϵ_{max} accordingly to obtain the highest possible PSNR at different bit rates. We are investigating a TVC that has variable block sizes.

To compare the complexity aspects, we coded 180 frames of Container and Silent CIF videos using four different encoders. In this experiment we kept the PSNR values close together to compare the bit rates and the encoding/decoding time. The four encoders are:

1- H.264/AVC-no motion, main profile, GOP of size 24 and number of reference frames equals to one. The JM18.4 version implemented in C/C++ was used.

2- H.264/AVC-Intra, main profile, and the period of I-pictures was set to one (JM18.4 version).

3- DISCOVER DVC codec [20] with GOP size of two. The C/C++ implementation available in [21] was used.

4- The proposed TVC with block size 16×16×180, $\epsilon_{max} = 11$, and $R_{max} = 65$ for container video and block size 32×32×180, $\epsilon_{max} = 11$, and $R_{max} = 50$ for silent video. The TVC is implemented in MATLAB. Hence, the implementation can be optimized further.

Table 1 shows the results, where all codecs are evaluated at a desktop computer with 12 GB of memory and an Intel Core i7 2600 CPU (8MB Cache, 3.4 GHz). It is very clear that TVC provides the best bitrate results and very competitive encoder/decoder times. As expected, DVC provided the best encoding times but at a significant penalty at the decoder side. Overall, TVC provides a good balance of coding efficiency and low-complexity (despite its MATLAB implementation).

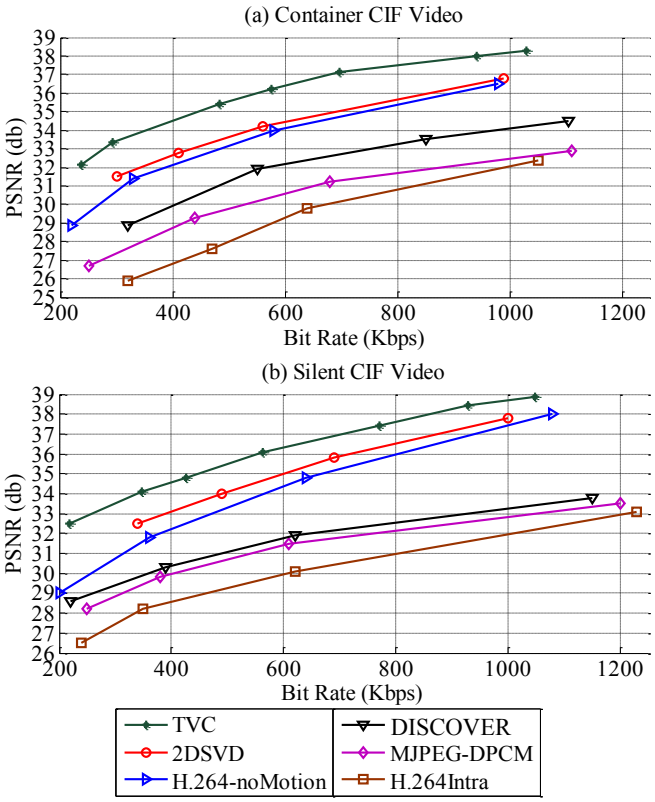

Fig. 3. PSNR plots of six low complexity video coders for (a) Container (b) Silent CIF videos.

Table 1. The encoding/decoding time, PSNR and bit rate of Container and Silent CIF videos with 180 frames using four coding methods.

Method	Video	PSNR (db)	bit rate (Kbps)	Encode time (s)	Decode time (s)
H.264/AVC-no motion	Container	36.37	1008	101	5
	Silent	34.54	495	90	**4.47**
H.264/AVC-Intra	Container	36	2092	114	9.2
	Silent	34.67	1800	114	9.3
DISCOVER	Container	36.67	1298	**95**	1440
	Silent	34.19	1092	**65**	1929
TVC	Container	36.4	**604**	134	**4.2**
	Silent	34.67	**379**	150	4.9

6. REFERENCES

[1] "Information technology -- JPEG 2000 image coding system: Motion JPEG 2000: ISO/IEC 15444-3:2007." .

[2] JV Team, "Advanced video coding for generic audiovisual services," *ITU-T Rec. H*, 2012.

[3] T. Wiegand, G. J. Sullivan, G. Bjøntegaard, and A. Luthra, "Overview of the H . 264 / AVC Video Coding Standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 7, pp. 560–576, 2003.

[4] R. Martins, C. Brites, J. Ascenso, and F. Pereira, "Refining Side Information for Improved Transform Domain Wyner – Ziv Video Coding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 19, no. 9, 2009.

[5] Z. Gu, W. Lin, B. Lee, and C. Lau, "Low-complexity video coding based on two-dimensional singular value decomposition.," *IEEE transactions on image processing : a publication of the IEEE Signal Processing Society*, vol. 21, no. 2, pp. 674–687, Feb. 2012.

[6] B. Zhou, F. Zhang, and L. Peng, "Compact Representation for Dynamic Texture Video Coding using Tensor Method," 2004.

[7] B. Zhou, F. Zhang, and L. Peng, "Video dimension reduction and coding using Multiple Tensor Rank-R Decomposition," *4th International Congress on Image and Signal Processing*, pp. 330–334, Oct. 2011.

[8] C. Ding, H. Huang, and D. Luo, "Tensor Reduction Error Analysis – Applications to Video Compression and Classification," *Computer Vision and Pattern Recognition*, 2008.

[9] M. Vasilescu and D. Terzopoulos, "Multilinear Analysis of Image Ensembles : TensorFaces," *European Conference on Computer Vision*, pp. 447–460, 2002.

[10] H. Wang and N. Ahuja, "Rank-R Approximation of Tensors Using Image-as-Matrix Representation," *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2, pp. 346–353, 2005.

[11] R. Costantini, L. Sbaiz, and S. Süsstrunk, "Higher order SVD analysis for dynamic texture synthesis.," *IEEE transactions on image processing : a publication of the IEEE Signal Processing Society*, vol. 17, no. 1, pp. 42–52, Jan. 2008.

[12] H. Wang and N. Ahuja, "Compact representation of multidimensional data using tensor rank-one decomposition," *Proceedings of the 17th International Conference on Pattern Recognition*, vol. 1, pp. 44–47, 2004.

[13] R. M. Bowen and C.-C. Wang, *Introduction to Vectors and Tensors*, vol. 2. Dover Publications, 2009.

[14] A. Cichocki, R. Zdunek, A. Phan, and S. Amari, *Nonnegative matrix and tensor factorizations: applications to exploratory multi-way data analysis and blind source separation*. Wiley, 2009.

[15] P. Comon, "Tensor decompositions," *Mathematics in Signal Processing V*, 2002.

[16] T. G. Kolda and B. W. Bader, "Tensor decompositions and applications," *SIAM review*, vol. 51, no. 3, pp. 455–500, 2009.

[17] W. Hackbusch, *Tensor spaces and numerical tensor calculus*. Springer, 2012.

[18] K. Dullemond and K. Peeters, *Introduction to Tensor Calculus*. 2010.

[19] J. Carroll and J. Chang, "Analysis of individual differences in multidimensional scaling via an N-way generalization of 'Eckart-Young' decomposition," *Psychometrika*, vol. 35, no. 3, pp. 283–319, 1970.

[20] X. Artigas, J. Ascenso, M. Dalai, S. Klomp, D. Kubasov, and M. Ouaret, "The DISCOVER codec: architecture, techniques and evaluation," *Picture Coding*, vol. 17, no. 9, pp. 1103–1120, 2007.

[21] "DISCOVER." [Online]. Available: http://www.discoverdvc.org/. [Accessed: 20-Nov-2012].