DYNAMIC UPDATING AND DOWNDATING MATRIX SVD AND TENSOR HOSVD FOR ADAPTIVE INDEXING AND RETRIEVAL OF MOTION TRAJECTORIES

Xiang Ma, Dan Schonfeld and Ashfaq Khokhar

Department of Electrical and Computer Engineering, University of Illinois at Chicago,
851 S Morgan St, Chicago, IL 60607, U.S.A.

ABSTRACT

Motion information is regarded as one of the most important cues for developing semantics in video data. Yet it is extremely challenging to build indexing and browsing tools for video data, particularly when it involves interactive motions of multiple objects. The problem is further complicated when the video archives are dynamically updated, and/or queries contains partial information. An efficient solution would require that the feature space used to represent the data be dynamically updated or downdated to allow frequent additions/deletions and matching of queries. Assuming Tensor HOSVD as the feature space, in this paper, we propose two novel algorithms, namely, Dynamic Tensor HOSVD Updating Algorithm ($DTSVD^{+}$) and Dynamic Tensor HOSVD Downdating Algorithm ($DTSVD^{-}$), for dynamically updating and downdating existing tensor HOSVD, without recalculating it from the raw data. The proposed algorithms are robustly applied to both full and partial multiple motion trajectories events with varying number of objects, trajectory lengths, and sampling rates. Simulations on real-world multiple motion trajectories data demonstrate the robustness and accuracy of the proposed algorithms.

Index Terms— Video retrieval, Motion trajectory analysis, Singular value decomposition, Multilinear algebra.

1. INTRODUCTION

Analysis of multiple motion trajectories has become an intensive area of research recently, due to its wide application in many areas in computer vision, such as activity recognition and video content analysis. However, the issues related to indexing and retrieval of video events with *multiple motion trajectories* are extremely challenging and have not yet been thoroughly addressed in the existing research work. Specifically, the practical utility of a robust indexing and retrieval system of multiple interacting motion trajectories must address three fundamental problems: (i) dynamic addition and deletion of entries in motion trajectory databases, (ii) dynamic matching of query and database entries when for a given motion event the number of trajectories in the query do not match with the corresponding event stored in the database, (iii) dynamic matching of query and database entries when the temporal length of the trajectories in the query event differ from the length of the trajectories in the corresponding event in the database. In our previous work [1], we presented a novel indexing and retrieval framework for compact and integrated representation of multiple interacting motion trajectories. Our key contribution was the formation of three new multi-linear algebraic structures –HOSVD being one of them– for compact and unified representation of events with multiple object trajectories in a reduced-dimension space. Three efficient algorithms for the indexing of multiple object trajectories corresponding to each of the algebraic structures based representations were proposed. For more details, please refer to [1]. However, this work did not consider dynamic updates and partial queries.

Also, most of the existing matrix SVD updating algorithms rely on the R-SVD algorithm [2]. However, it requires that the width of the matrix must be larger or equal to its height, which limits its applicability to arbitrary matrices. We extend these results by developing updating and downdating algorithms of SVD representation for arbitrary matrices. We subsequently rely on these methods as the foundation for developing new updating and downdating algorithms of HOSVD tensors.

2. MATRIX SVD AND TENSOR HOSVD

The classical matrix singular value decomposition (SVD) is a decomposition of the matrix A as product of singular value matrix and singular vector matrices.

$$A = U \times \Sigma \times V^{T}. \tag{1}$$

where in the above equation, U and V are left and right singular vector matrices, whose columns are left and right singular vectors of matrix A, while Σ is a diagonal matrix whose entries are eigen values of matrix A.

The tensor HOSVD, or higher-order SVD (HOSVD) [3] is a generalization of the classical matrix SVD to higher-order tensors. An N-th order tensor $\mathcal{A}$ is an N-dimensional matrix composed of N vector spaces. The tensor SVD, or HOSVD is aiming of find N orthonormal base matrices, also called

This work is funded in part of funding from NSF IIS-0534438.

mode matrices, $U^{(1)}, ..., U^{(n)}, ..., U^{(N)}$ which span these N spaces, respectively.

The tensor HOSVD is a decomposition of the tensor $\mathcal{A}$ as product of a core tensor and those bases matrices,

$$\mathcal{A} = \mathcal{B} \times_1 U^{(1)} \times_2 U^{(2)} \times_3 U^{(3)} ... \times_N U^{(N)}. \quad (2)$$

where in the above equation, $\mathcal{B}$ is a core tensor that controls the interactions among the mode matrices $U^{(1)}, ..., U^{(n)}, ..., U^{(N)}$. The core tensor can be obtained by

$$\mathcal{B} = \mathcal{A} \times_1 U^{(1)^T} \times_2 U^{(2)^T} ... \times_N U^{(N)^T}. \quad (3)$$

The HOSVD of a tensor is calculated by performing a matrix SVD on the unfolded matrix of tensor, which enables us to dynamically update or downdate the HOSVD of a tensor when arbitrary entries are added or deleted.

3. DYNAMIC UPDATING AND DOWNDATING OF MATRIX SVD AND TENSOR HOSVD

Given a three-dimensional tensor $\mathcal{A}$ consisting of multiple trajectories, for example, of size $I \times J \times K$. There are K entries of multiple trajectory data, each entry consists of J motion trajectories, and each trajectory is of length I, as shown in Fig 1. The Tensor HOSVD of the tensor $\mathcal{A}$ is

$$\mathcal{A} = \mathcal{B} \times U^{(1)} \times U^{(2)} \times ... \times U^{(N)}. \quad (4)$$

where tensor $\mathcal{B}$ is the core tensor, and $U^{(i)}, i = 1, 2, ..., N$ are the loading matrices. Inside the tensor $\mathcal{A}$, there is another tensor $\mathcal{A}'$, of size $I' \times J' \times K'$; in other words, tensor $\mathcal{A}'$ contains data the same as portion of tensor $\mathcal{A}$. The Tensor HOSVD of tensor $\mathcal{A}'$ is

$$\mathcal{A}' = \mathcal{B}^* \times U^{(1)*} \times U^{(2)*} \times ... \times U^{(N)*}. \quad (5)$$

The *tensor HOSVD downdating* problem arises when we know the Tensor HOSVD of the tensor $\mathcal{A}$ and want to know the Tensor SVD of the tensor $\mathcal{A}'$. And when we know the Tensor HOSVD of the smaller one, e.g. $\mathcal{A}'$, and we want to know the Tensor HOSVD of the one after more entries added, e.g. $\mathcal{A}$, that is called *tensor HOSVD updating*.

Figure. 1 demonstrate the structural relationship of tensor $\mathcal{A}$ (white) of size $I \times J \times K$ and tensor $\mathcal{A}'$ (red) of size $I' \times J' \times K'$, where $I' \leq I$, $J' \leq J$ and $K' \leq K$. We can view tensor $\mathcal{A}'$ is a portion of tensor $\mathcal{A}$, where the former consists portion of the same data from the letter. The unfolded matrices are also depicted to further clarify the relationship of data in tensor $\mathcal{A}$ and in tensor $\mathcal{A}'$. Let the left up corner of tensor $\mathcal{A}$ be the origin of a three-dimensional coordinate system (X, Y, Z), and the left up corner of tensor $\mathcal{A}'$ be at position (x, y, z), then if we know the position (x, y, z) and sizes of the tensor $\mathcal{A}$, i.e. I, J, K and sizes of the tensor $\mathcal{A}'$, i.e. $I', J' and K'$, then we can estimate the Tensor HOSVD of one tensor if we know the Tensor HOSVD of the other, these

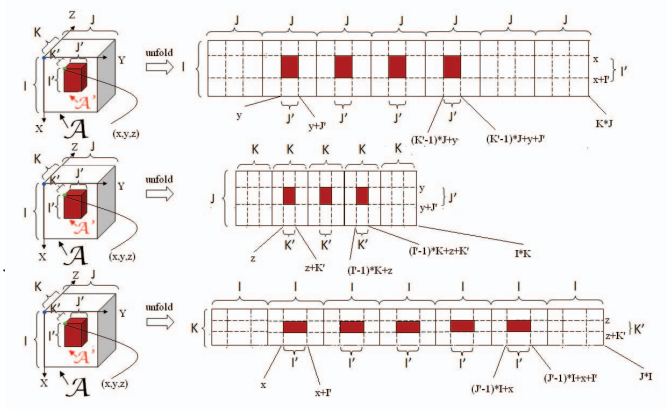

Fig. 1. Dynamic updating and downdating of Tensor HOSVD

are *the tensor HOSVD updating* and *tensor HOSVD downdating* problems.

Traditionally, if we want to determine the Tensor HOSVD of the updated or downdated tensor, we need to follow a tedious procedure to recalculate the Tensor HOSVD from the updated raw data. Here, we propose two novel algorithms for dynamic updating and downdating of tensor HOSVDs, which adaptively calculate the new Tensor HOSVD based on existing Tensor HOSVD of the previous data.

3.1. Dynamic Tensor HOSVD Updating Algorithm

We utilize the proposed matrix SVD updating algorithm, namely SVD^+, which consists of two algorithms: SVD^{+C}, to update the matrix SVD when arbitrary columns are added; and SVD^{+R}, to update the matrix SVD when arbitrary rows are added. The combination of SVD^{+C} and SVD^{+R} algorithms is used to dynamically update the SVD of a matrix when arbitrary entries are added. This can be done first by locating the positions of entries that are to be added and then performing a two step dynamic updating of matrix SVD, either by applying SVD^{+C} followed by SVD^{+R}, or by applying SVD^{+R} followed by SVD^{+C}. Due to space limitations, these algorithms are not listed here, and will be presented in a detailed version of this work some where else.

3.2. Dynamic Tensor HOSVD Downdating Algorithm

In this subsection, we propose a novel dynamic tensor HOSVD downdating algorithm ($DTSVD^-$) which solves the problem of estimating Tensor HOSVD of a tensor when entries are deleted, or searching for a specific location of a partial tensor. In our $DTSVD^-$ algorithm, we utilize a novel SVD^-, which consists of two algorithms, namely, SVD^{-C} and SVD^{-R}. The combination of SVD^{-C} and SVD^{-R} algorithms is used to dynamically downdating the SVD of a matrix when arbitrary entries are deleted. This can be done by firstly locating the positions of entries that are to be kept and then performing a two step dynamic downdating of matrix

Algorithm 1 Dynamic tensor HOSVD updating algorithm $DTSVD^{+}$; $[\mathcal{B}^{*}, U^{(1)*}, U^{(2)*}, U^{(3)*}, ..., U^{(N)*}] = DTSVD^{+}[\mathcal{B}, U^{(1)}, U^{(2)}, U^{(3)}, ..., U^{(N)}, \{x, y, z\}_{x\in X, y\in Y, z\in Z}]$

Input: HOSVD of original tensor $\mathcal{A}$ i.e. $\{\mathcal{B}, U^{(1)}, U^{(2)}, ..., U^{(N)}\}$ and positions of entries that need to be added $\{x, y, z\}_{x\in X, y\in Y, z\in Z}$.

Output: HOSVD of updated tensor $\mathcal{A}^{*}$, i.e. $\{\mathcal{B}^{*}, U^{(1)*}, U^{(2)*}, ..., U^{(N)*}\}$.

1. for n=1:N,
2. Compute the mode-n unfolding matrix $A(n)$ of tensor $\mathcal{A}$, such that $T(n) = U_n \times \Sigma_n \times V_n$.
3. Update the HOSVD of $T(n)$ using SVD^{+} algorithm. $[U_n^{*}, \Sigma_n^{*}, V_n^{*}] = SVD^{+}[U_n, \Sigma_n, V_n, \{K_j\}]$, where $\{K_j\}$ is decided by $\{x, y, z\}_{x\in X, y\in Y, z\in Z}$ (e.g., as shown in Figure 1 for three dimensional tensor).
4. end.
5. Set the mode-n loading matrix $U^{(n)*}$ of the downdated tensor $\mathcal{A}^{*}$ as the left singular vector matrix U_n^{*} of the mode-n unfolding matrix.
6. The updated core tensor $\mathcal{B}^{*}$ of the updated tensor $\mathcal{A}^{*}$ can be calculated as $\mathcal{B}^{*} = T^{*} \times_1 U^{(1)*} \times_2 U^{(2)*} \times_3 U^{(3)*} ... \times_N U^{(N)*}$.

Algorithm 2 Dynamic tensor HOSVD downdating algorithm $DTSVD^{-}$; $[\mathcal{B}^{*}, U^{(1)*}, U^{(2)*}, U^{(3)*}, ..., U^{(N)*}] = DTSVD^{-}[\mathcal{B}, U^{(1)}, U^{(2)}, U^{(3)}, ..., U^{(N)}, \{x, y, z\}_{x\in X, y\in Y, z\in Z}]$

Input: HOSVD of original tensor $\mathcal{A}$ i.e. $\{\mathcal{B}, U^{(1)}, U^{(2)}, ..., U^{(N)}\}$ and positions of entries that need to be kept $\{x, y, z\}_{x\in X, y\in Y, z\in Z}$.

Output: HOSVD of downdated tensor $\mathcal{A}^{*}$, i.e. $\{\mathcal{B}^{*}, U^{(1)*}, U^{(2)*}, ..., U^{(N)*}\}$.

1. for n=1:N,
2. Compute the mode-n unfolding matrix $A(n)$ of tensor $\mathcal{A}$, such that $A(n) = U_n \times \Sigma_n \times V_n$.
3. Downdate the HOSVD of $A(n)$ using SVD^{-} algorithm. $[U_n^{*}, \Sigma_n^{*}, V_n^{*}] = SVD^{-}[U_n, \Sigma_n, V_n, \{K_j\}]$, where $\{K_j\}$ is decided by $\{x, y, z\}_{x\in X, y\in Y, z\in Z}$ (e.g., as shown in Figure 1 for three dimensional tensor)
4. end.
5. Set the mode-n loading matrix $U^{(n)*}$ of the downdated tensor $\mathcal{A}^{*}$ as the left singular vector matrix U_n^{*} of the mode-n unfolding matrix.
6. The downdated core tensor $\mathcal{B}^{*}$ of the downdated tensor $\mathcal{A}^{*}$ can be calculated as $\mathcal{B}^{*} = \mathcal{A}^{*} \times_1 U^{(1)*} \times_2 U^{(2)*} \times_3 U^{(3)*} ... \times_N U^{(N)*}$.

SVD, either first applying SVD^{-C} then applying SVD^{-R}, or vice versa.

4. EXPERIMENTAL RESULTS

We test the effectiveness of the proposed algorithms in an indexing and retrieval framework where video events consisting of multiple motion trajectories are stored and querried in a query-by-example paradigm. The experiments are performed using Matlab on a 2.0 GHz Pentium Dual Core Laptop with 2GB memories, without code optimization.

The accuracy of our $DTSVD^{-}$ applied to high-dimensional tensor is shown in the following experiment. A three dimensional tensor of size $20 \times 15 \times 10$ is randomly generated, as shown in Figure 2 (a). The Tensor HOSVD of A is shown in Figures 2 (b)-(e). We extract a new tensor A^{*} from A, of size $7 \times 10 \times 5$, as shown in Figure 2 (f)(the relative position of A^{*} in A is highlighted in yellow). Given the tensor HOSVD of A, we estimate the tensor HOSVD of A^{*}, using our $DTSVD^{-}$ algorithm, which prevents the re-calculating of tensor HOSVD. The tensor HOSVD of A^{*} are depicted in Figures 2 (g)-(j). The estimation results are shown in Table 1. The true values are obtained by direct Tensor SVD calculation while the estimated values are acquired using our $DTSVD^{-}$ algorithm. We can see that the estimation errors are at machine precision level thus are actually zeros.

For multiple motion trajectory retrieval, figure 3 (a) shows a 2-traj query, figures 3(b)-(c) show the retrieved top two most-similar 3-trajectory entries; while figure 3(d) depicts another query of temporal length 300 sec., and figure 3(e)-(f) show the retrieved top two most-similar entries. We can see the retrieved multiple trajectories are visually quite similar with the queries in both cases.

Algorithm 3 Matrix SVD downdating algorithm SVD^{-R}; $[U^{*}, \Sigma^{*}, V^{*}] = SVD^{-R}[U, \Sigma, V, \{L_m\}_{(m=1,2,...,I)}]$

Input: SVD of original matrix B: $\{U, \Sigma, V\}$, and the I sets of row vectors to be kept: $\{L_m\}, m = 1, 2, ..., I$.

Output: SVD of downdated matrix B^{*} after deleting arbitrary row entries: $\{U^{*}, \Sigma^{*}, V^{*}\}$.

$$[V^{*}, \Sigma^{*T}, U^{*}] = SVD^{-C}[V, \Sigma^{T}, U, \{L_m\}_{(m=1,2,...,I)}]. \quad (6)$$

Table 1. Estimation errors for tensor HOSVD downdating

$\lvert S^{*}(true) - S^{*}(est)\rvert$	$\lvert U_1^{*}(true) - U_1^{*}(est)\rvert$
0	3.3388×10^{-14}
$\lvert U_2^{*}(true) - U_2^{*}(est)\rvert$	$\lvert U_3^{*}(true) - U_3^{*}(est)\rvert$
1.0586×10^{-14}	1.9201×10^{-14}

5. CONCLUSION

In this paper, we have presented two novel algorithms, namely, $DTSVD^{+}$ and $DTSVD^{-}$, for dynamic updating and downdating of existing tensor HOSVD, when new data is added or deleted, without recalculating of the raw data. They have been successfully used to solve three fundamental problems related to searching, indexing and retrieval of multiple motion trajectories. Simulations results on tensor and multiple motion trajectories data demonstrate the robustness

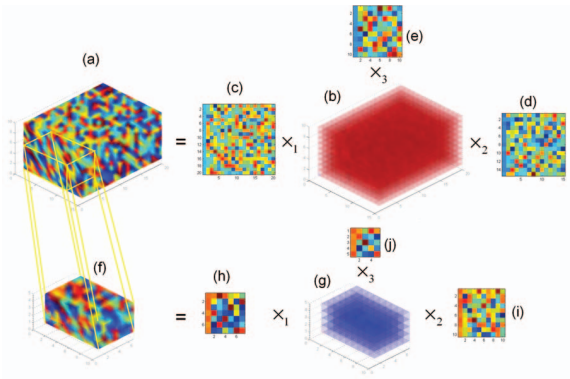

Fig. 2. Tensor HOSVD Downdating: (a) Tensor A of size $20 \times 15 \times 10$ (b) Core tensor S of A. (c) Basis Matrix U_1 (d) Basis Matrix U_2 (e) Basis Matrix U_3 (f) Tensor A^* extracted from A, of size $7 \times 10 \times 5$. (g) Core tensor S^* of A^* (h) Basis Matrix U_1^* (i) Basis Matrix U_2^* (j) Basis Matrix U_3^*

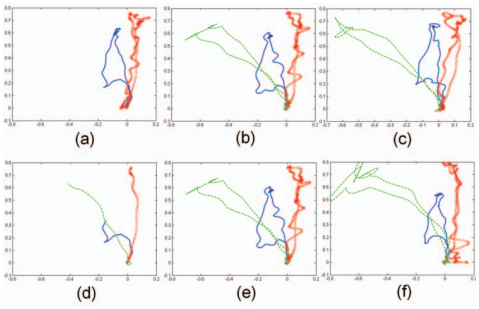

Fig. 3. Multiple motion trajectory retrieval using tensor HOSVD downdating: (a) A 2-traj query (b) The retrieved most-similar 3-traj entry (c) The retrieved second most-similar 3-traj entry (d) A 3-traj query of length 300 sec. (e) The retrieved most-similar entry of length 1024 sec. (f) The retrieved second most-similar entry of length 1024 sec.

and accuracy of the proposed algorithms.

6. REFERENCES

[1] X. Ma, F. Bashir, A. A. Khokhar and D. Schonfeld, "Event Analysis Based on Multiple Interactive Motion Trajectories", IEEE Trans. on Circuits and Systems for Video Technology, to be published.

[2] A. Levy and M. Lindenbaum, "Sequential Karhunen-Loeve Basis Extraction and its Application to Images", IEEE Trans. on Image Processing, Vol. 9, No. 8, pp. 1371-1374, 2000.

[3] L. Lathauwer, B. D. Moor and J. Vandewalle, "A multilinear singular value decomposition", SIAM Journal on Matrix Analysis and Applications (SIMAX), pp. 1253-1278, 2000.

Algorithm 4 Matrix SVD downdating algorithm SVD^{-C}; $[U^*, \Sigma^*, V^*] = SVD^{-C}[U, \Sigma, V, \{K_n\}_{(n=1,2,...,J)}]$

Input: SVD of original matrix B: $\{U, \Sigma, V\}$, and the J sets of numbers for column vectors to be kept: $\{K_n\}, n = 1, 2, ..., J$.

Output: SVD of downdated matrix B^* after deleting arbitrary column entries: $\{U^*, \Sigma^*, V^*\}$.

1. Define a transform operator P, such that the transpose of the original right singular vector matrix V is transformed into the following form, where columns in positions corresponding to the deleted entries are transformed to unit vectors that form identity matrices. One possible construction of P is the product of series of Householder Transform operators, where $P = \prod_{j_d} P_{j_d}$, and j_d belongs to the set of indices of columns that would be deleted, P_{j_d} is householder transform matrix that transforms j_dth column to unit vector.
2. Define the column selecting operator S_c as

$$S_c = [e_{C_1}|e_{C_2}|...|e_{C_m}]; \tag{7}$$

where $e_{C_m} = [0, 0, ..., 1, 0, 0...0]^T$ is column vector with 1 at C_mth position and all 0s otherwise. And define the row selecting operator S_r as

$$S_r = \begin{bmatrix} e_{R_1}^T \\ e_{R_1}^T \\ ... \\ e_{R_n}^T \end{bmatrix} \tag{8}$$

where $e_{R_m} = [0, 0, ..., 1, 0, 0...0]^T$ is column vector with 1 at R_mth position and all 0s otherwise.
3. Construct $\widetilde{V}$ as $\widetilde{V} = Sr(PV^T)Sc$.
4. Calculate an intermediate core matrix $\widetilde{\widetilde{S}}$,

$$[\widetilde{\widetilde{U}}, \widetilde{\widetilde{S}}, \widetilde{\widetilde{V}}] = diagonalize(\widetilde{V}^T B^{*T} B^* \widetilde{V}). \tag{9}$$

.
5. Estimate the singular value decomposition

$$\Sigma^* = \begin{cases} \begin{bmatrix} diag(\sqrt{(\widetilde{\widetilde{S}})}) \\ 0_{(M-\sum_{n=1}^{J} K_n)(\sum_{n=1}^{J} K_n)} \end{bmatrix} & \text{if } \sum_{n=1}^{J} K_n \leq M \\ diag(\sqrt{(\widetilde{\widetilde{S}})})(1:M, \sum_{n=1}^{J} K_n) & \text{otherwise} \end{cases}$$

6. $V^* = \widetilde{V} \times \widetilde{\widetilde{V}}$.
7. $U^* = B^* \times V^* \times \Sigma^{*+}$, where Σ^{*+} is pseudo inverse of Σ^*.