Accurate Tensor Decomposition with Simultaneous Rank Approximation for Surveillance Videos

1st Ramin Goudarzi Karim
School of Business, Entrepreneurship, and CIS
Stillman College
Tuscaloosa, AL
rkarim@stillman.edu

2nd Guimu Guo
Department of Computer Science
University of Alabama at Birmingham
Birmingham, AL
guimuguo@uab.edu

3rd Da Yan
Department of Computer Science
University of Alabama at Birmingham
Birmingham, AL
yanda@uab.edu

4th Carmeliza Navasca
Department of Mathematics
University of Alabama at Birmingham
Birmingham, AL
cnavasca@uab.edu

2020 54th Asilomar Conference on Signals, Systems, and Computers | 978-0-7381-3126-9/20/$31.00 ©2020 IEEE | DOI: 10.1109/IEEECONF51394.2020.9443285

***Abstract*—Canonical polyadic (CP) decomposition of a tensor is a basic operation in a lot of applications such as data mining and video foreground/background separation. However, existing algorithms for CP decomposition require users to provide a rank of the target tensor data as part of the input and finding the rank of a tensor is an NP-hard problem. Currently, to perform CP decomposition, users are required to make an informed guess of a proper tensor rank based on the data at hand, and the result may still be suboptimal. In this paper, we propose to conduct CP decomposition and tensor rank approximation together, so that users do not have to provide the proper rank beforehand, and the decomposition algorithm will find the proper rank and return a high-quality result. We formulate an optimization problem with an objective function consisting of a least-squares Tikhonov regularization and a sparse ℓ_1-regularization term. We also test its effectiveness over real applications with moving object videos.**

***Index Terms*—H.8 - Speech, Image and Video Processing: Computer Vision, Image and Video Analysis**

I. Introduction

Canonical polyadic (CP) decomposition of a tensor is widely used in many real applications. For example, in neuroscience, data from EEG and fMRI usually naturally fit into a multi-way array and thus CP decomposition is a popular tool for analyzing them [8]. As another example, In data mining and machine learning, CP decomposition has been used for discussion detanglement in online chat rooms [1] and compressing deep learning models [2]. Albeit a fundamental operation, CP decomposition requires users to provide a rank of the target tensor data as part of the input.

Moreover, (Challenge I) finding the rank of tensors is NP-hard [11]. Currently, to perform CP decomposition, users are required to make an informed guess of a proper tensor rank based on the data at hand, and the result may still be suboptimal (Challenge II): [7] showed that tensors can be ill-posed and failed to have their best low-rank approximations. For example, there exists a tensor space in which no rank-3 tensor has an optimal rank-2 approximation [7]. As a result, an overestimated rank is usually provided by users.

In this paper, we tackle the problem of constructing CP decomposition and tensor rank approximation together. Our new formulation addresses the 2 challenges mentioned. Our main contributions are summarized as follows:

- We propose the problem of CP decomposition with simultaneous rank approximation, which addresses Challenges I and II described above.
- We formulate the CP decomposition problem with Tikhonov regularization to avoid ill-posed decomposition outputs and to help the convergence, and with an ℓ_1-regularization term for effective rank approximation.
- We design a block-coordinate descent algorithm to solve the new decomposition formulation, which uses a proximal alternating minimization technique for the rank and an alternating least-squares for the decomposition.
- We test its effectiveness in rank approximation and the quality of decomposition over real applications with moving object videos.

II. Background and Related Work

A. Tensor Notations

An N-th order tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ is a multidimensional array with entries $x_{i_1 i_2 \cdots i_N}$ for $i_k \in \{1, \ldots, I_k\}$ where $k \in 1, \ldots, N$. In particular, a third-order tensor $\mathcal{X} \in \mathbb{R}^{I \times J \times K}$ is a multidimensional array with entries x_{ijk} for $i \in \{1, \ldots, I\}$, $j \in \{1, \ldots, J\}$ and $k \in \{1, \ldots, K\}$.

The Kronecker product of two vectors $\mathbf{a} \in \mathbb{R}^I$ and $\mathbf{b} \in \mathbb{R}^J$ is denoted by $\mathbf{a} \otimes \mathbf{b} \in \mathbb{R}^{IJ}$:

$$\mathbf{a} \otimes \mathbf{b} = \left(a_1 \mathbf{b}^T, \ldots, a_I \mathbf{b}^T\right)^T.$$

The Khatri-Rao product of two matrices $A \in \mathbb{R}^{I \times J}$ and $B \in \mathbb{R}^{K \times J}$ is defined as

$$A \odot B = (\mathbf{a}_1 \otimes \mathbf{b}_1, \ldots, \mathbf{a}_J \otimes \mathbf{b}_J).$$

The outer product of three vectors $\mathbf{a} \in \mathbb{R}^I$, $\mathbf{b} \in \mathbb{R}^J$, $\mathbf{c} \in \mathbb{R}^K$ is a third-order tensor $\mathcal{X} = \mathbf{a} \circ \mathbf{b} \circ \mathbf{c}$ with the entries $x_{ijk} = a_i b_j c_k$.

B. CP Decomposition

In 1927, Hitchcock [12], [13] proposed the idea of the polyadic form of a tensor, i.e., expressing a tensor as the sum of a finite number of rank-one tensors. Today, this decomposition is called the canonical polyadic (CP), aka. CANDECOMP or PARAFAC. It is a basic operation in data mining and machine learning, and has been extensively applied to address various problems in science and engineering (see Introduction for some examples). For notation simplicity, we focus on third-order tensor in our presentation but our approach generalizes to higher-order tensors. Specifically, the CP decomposition of a given third-order tensor $\mathcal{X} \in \mathbb{R}^{I \times J \times K}$ factorizes it to a sum of rank-one tensors shown as follows:

$$\mathcal{X} \approx \sum_{r=1}^{R} \alpha_r (\mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r) \triangleq \hat{\mathcal{X}}, \tag{1}$$

where R is a user-provided rank parameter for $\mathcal{X}$. For simplicity, we use the notation $[A, B, C, \boldsymbol{\alpha}]_R$ to represent the sum on the right hand side of Equation (1), where $A \in \mathbb{R}^{I \times R}$, $B \in \mathbb{R}^{J \times R}$ and $C \in \mathbb{R}^{K \times R}$ are called factor matrices given below:

$$A = [\mathbf{a}_1, \ldots, \mathbf{a}_R], \quad B = [\mathbf{b}_1, \ldots, \mathbf{b}_R], \quad C = [\mathbf{c}_1, \ldots, \mathbf{c}_R].$$

Thus, we have $\hat{\mathcal{X}} = [A, B, C, \boldsymbol{\alpha}]_R$.

The CP decomposition problem can be formulated as an optimization problem. Given a rank parameter R, the goal is to find vectors $\mathbf{a}_r, \mathbf{b}_r, \mathbf{c}_r$, such that the distance between the tensor $\mathcal{X}$ and the sum of the outer products of $\mathbf{a}_r, \mathbf{b}_r, \mathbf{c}_r$ is minimized. Frobenius norm $\|.\|_F$ is usually used to measure this distance, or equivalently, for the residual tensor $(\mathcal{X} - \hat{\mathcal{X}})$ we minimize the sum of its entries squared:

$$\min_{A,B,C,\boldsymbol{\alpha}} \frac{1}{2} \|\mathcal{X} - [A, B, C, \boldsymbol{\alpha}]_R\|_F^2 \tag{2}$$

Let us rewrite the objective of Equation (2) as $f(.)$ below:

$$f(A, B, C, \boldsymbol{\alpha}) = \frac{1}{2} \|\mathcal{X} - [A, B, C, \boldsymbol{\alpha}]\|_F^2 \tag{3}$$

The optimization problem of minimizing $f(.)$ is non-convex, but if we fix three variables among A, B, C and $\boldsymbol{\alpha}$, and find the value of the other variable to minimize $f(.)$, the problem is actually a linear least-squares problem. Independently proposed by [6] and [10], Alternating Least-Squares (ALS) alternates among these least-squares problem to update one variable at a time until some convergence criterion is satisfied. ALS is the most popular method for CP decomposition thanks to its robustness and ease of implementation.

However, ALS can converge very slowly [16], and the tensor rank R should be provided by users. This paper addresses both problems.

C. Rank Approximation

The problem of finding the rank of a tensor can be formulated as a constrained optimization problem.

$$\min_{\boldsymbol{\alpha}} \|\boldsymbol{\alpha}\|_0 \ \text{ s.t. } \ \mathcal{X} = [A, B, C, \boldsymbol{\alpha}]_R$$

where $\|\boldsymbol{\alpha}\|_0$ is the ℓ_0-norm of $\boldsymbol{\alpha}$, i.e., the number of non-zero entries in $\boldsymbol{\alpha}$. Since ℓ_0-optimization problem is non-convex and difficult to solve, [4] proposed to use a solution to the convex ℓ_1-optimization problem as a good approximation, which is widely used in areas such as compressive sensing [5], [9].

We thus write the rank approximation problem as:

$$\min_{\boldsymbol{\alpha}} \|\boldsymbol{\alpha}\|_1 \ \text{ s.t. } \ \mathcal{X} = [A, B, C, \boldsymbol{\alpha}]_R$$

In order to obtain both a CP decomposition of the given tensor $\mathcal{X}$ as well as its rank approximation, the rank approximation problem can be formulated as follow:

$$\min_{A,B,C,\boldsymbol{\alpha}} \frac{1}{2} \|\mathcal{X} - [A, B, C, \boldsymbol{\alpha}]\|_F^2 + \gamma \|\boldsymbol{\alpha}\|_1 \tag{4}$$

where $\gamma > 0$ is the regularization parameter. In machine learning, ℓ_1-regularization (or lasso) is well-known to encourage the sparsity of optimization variable $\boldsymbol{\alpha}$.

There are several works on low rank tensor decomposition and sparsity; for example, see [15], [19], [20]. However, most of the work in the literature are in the Tucker (HOSVD) decompostion. In the paper of [19], a convex optimization is formulated by using the structured Schatten norm regularization using the Tucker decomposition.

III. Problem Formulation

Recall that the objective function in Equation (4) is a composition of a smooth and non-smooth functions. Moreover, it is known that CP decomposition is invariant to scaling and permutation of factor matrices. In order to overcome the scaling indeterminacy, we add a Tikhonov-type regularization term [14] to our objective function.

Recall f from Equation (3). We also define g as follow to simplify notations in our later discussion:

$$g(\boldsymbol{\alpha}) = \gamma \|\boldsymbol{\alpha}\|_1. \tag{5}$$

Then our rank approximation problem can be formulated as

$$\min_{A,B,C,\boldsymbol{\alpha}} f(A, B, C, \boldsymbol{\alpha}) + \frac{\lambda}{2} (\|A\|_F^2 + \|B\|_F^2 + \|C\|_F^2) + g(\boldsymbol{\alpha}). \tag{6}$$

For ease of presentation, we further denote the objective function in Equation (6) by $\Psi(A, B, C, \boldsymbol{\alpha})$, and define $\omega = (A, B, C, \boldsymbol{\alpha})$. For simplicity, when $B, C, \boldsymbol{\alpha}$ are fixed, we represent $f(\omega)$ by $f(A)$ and $\Psi(\omega)$ by $\Psi(A)$. We define $f(B)$, $f(C)$, $f(\boldsymbol{\alpha})$ and $\Psi(B)$, $\Psi(C)$, $\Psi(\boldsymbol{\alpha})$ similarly.

IV. Algorithm

In this section, we propose a block coordinate descent algorithm for solving the problem of Equation (6). We consider four (blocks of) variables A, B, C and $\boldsymbol{\alpha}$. In particular, at each inner iteration, we solve the following minimization problems, where superscript k denotes the iteration number (e.g., A^k denotes the value of A after Iteration k):

$$A^{k+1} = \text{argmin}_A \{ f(A, B^k, C^k, \boldsymbol{\alpha}^k) + \frac{\lambda}{2} \|A\|_F^2 \}, \tag{7}$$

$$B^{k+1} = \text{argmin}_B \{ f(A^{k+1}, B, C^k, \boldsymbol{\alpha}^k) + \frac{\lambda}{2} \|B\|_F^2 \}, \tag{8}$$

$$C^{k+1} = \text{argmin}_A\{f(A^{k+1}, B^{k+1}, C, \boldsymbol{\alpha}^k) + \frac{\lambda}{2}\|C\|_F^2\}, \quad (9)$$

and

$$\boldsymbol{\alpha}^{k+1} = \text{argmin}_{\boldsymbol{\alpha}}\{L_{\beta f}^k(A^{k+1}, B^{k+1}, C^{k+1}, \boldsymbol{\alpha}) + g(\boldsymbol{\alpha})\}, \quad (10)$$

where $L_{\beta f}^k(\boldsymbol{\alpha})$ represents the proximal linearization [3] of $f(\boldsymbol{\alpha})$ given by the equation below; note that here we omitted A^{k+1}, B^{k+1} and C^{k+1} that are fixed in $L_{\beta f}^k(.)$ and $f(.)$ for notation simplicity.

$$L_{\beta f}^k(\boldsymbol{\alpha}) = \langle \boldsymbol{\alpha} - \boldsymbol{\alpha}^k, \nabla_{\boldsymbol{\alpha}} f(\boldsymbol{\alpha}^k)\rangle + \frac{1}{2\beta}\|\boldsymbol{\alpha} - \boldsymbol{\alpha}^k\|_2^2, \quad (11)$$

and $\beta > 0$ is a fixed step size parameter, and $\langle \cdot, \cdot \rangle$ is the inner product operation.

Since each of the minimization problems in Equations (7)-(10) is strictly convex, $A, B, C, \boldsymbol{\alpha}$ are uniquely determined at each iteration.

Updating A, B and C. The subproblems in Equations (7)-(9) are standard linear least-squares problems with an additional Tikhonov regularization term. Without loss of generality, consider the subproblem of Equation (7).

Taking the objective's gradient with respective to A, we have:

$$\nabla_A f(A, B^k, C^k, \boldsymbol{\alpha}^k) + \lambda A,$$

Setting the gradient to 0, we obtain:

$$A(E^k(E^k)^T + \lambda I) = \mathcal{X}_{(1)}(E^k)^T, \quad (12)$$

where $E^k = \text{diag}(\boldsymbol{\alpha}^k)(C^k \odot B^k)^T$. Similarly, we can obtain

$$B(F^k(F^k)^T + \lambda I) = \mathcal{X}_{(2)}(F^k)^T, \quad (13)$$

where $F^k = \text{diag}(\boldsymbol{\alpha}^k)(C^k \odot A^{k+1})^T$, and

$$C(G^k(G^k)^T + \lambda I) = \mathcal{X}_{(3)}(G^k)^T, \quad (14)$$

where $G^k = \text{diag}(\boldsymbol{\alpha}^k)(B^{k+1} \odot A^{k+1})^T$.

Since the objective functions in Equations (7)-(9) are strictly convex, the first order optimality condition is sufficient for a point to be minimum. In other words, the exact solutions are given by Equations (12)-(14).

Updating $\boldsymbol{\alpha}$. To update $\boldsymbol{\alpha}$ in Equation (10), we need the proximal operator [17] as defined below.

Definition 4.1 (proximal operator): Let g: $\mathbb{R}^n \to \mathbb{R}$ be a lower semi-continuous convex function, then the proximal operator of g with step size parameter $\beta > 0$ is defined as follow:

$$\boldsymbol{prox}_{\beta g}(\boldsymbol{y}) = \text{argmin}_{\boldsymbol{x}}\{g(\boldsymbol{x}) + \frac{1}{2\beta}\|\boldsymbol{x} - \boldsymbol{y}\|_2^2\}. \quad (15)$$

Using this notation, Equation (10) is equivalent to

$$\boldsymbol{\alpha}^{k+1} = \boldsymbol{prox}_{\beta g}(\boldsymbol{\alpha}^k - \beta \nabla_{\boldsymbol{\alpha}} f(\boldsymbol{\alpha}^k)), \quad (16)$$

Note that the proximal operator in Equation (15) is well-defined because the function $g(\boldsymbol{\alpha})$ is continuous and convex.

Now let us consider $\nabla_{\boldsymbol{\alpha}} f(\boldsymbol{\alpha})$ in Equation (16). Note that

$$\text{vec}([A, B, C, \boldsymbol{\alpha}]_R) = \sum_{r=1}^{R} \alpha_r \text{vec}(\boldsymbol{a}_r \circ \boldsymbol{b}_r \circ \boldsymbol{c}_r) = M\boldsymbol{\alpha},$$

where $M \in \mathbb{R}^{IJK \times R}$ is the matrix with columns $(\boldsymbol{c}_r \otimes \boldsymbol{b}_r \otimes \boldsymbol{a}_r)$. Note that M is computed from A, B and C. Given the definition of M, we can rewrite the objective function $f(\boldsymbol{\alpha})$ defined in Equation (3) as follows:

$$f(A, B, C, \boldsymbol{\alpha}) = \frac{1}{2}\|\text{vec}(\mathcal{X}) - M\boldsymbol{\alpha}\|_2^2 \quad (17)$$

It is easy to calculate the gradient of Equation (17) with respect to $\boldsymbol{\alpha}$:

$$\nabla_{\boldsymbol{\alpha}} f(A, B, C, \boldsymbol{\alpha}) = M^T(M\boldsymbol{\alpha} - \text{vec}(\mathcal{X})) \quad (18)$$

This implies the Lipschitz continuity of the gradient of $f(.)$ with respect to $\boldsymbol{\alpha}$, and we have the Lipschitz constant $Q_{\boldsymbol{\alpha}} = \|M^T M\|_F$ (assuming A, B and C fixed in $f(.)$) such that

$$|\nabla_{\boldsymbol{\alpha}} f(\boldsymbol{\alpha}_1) - \nabla_{\boldsymbol{\alpha}} f(\boldsymbol{\alpha}_2)| \leq Q_{\boldsymbol{\alpha}}\|\boldsymbol{\alpha}_1 - \boldsymbol{\alpha}_2\|_2. \quad (19)$$

which can be easily shown as follows:

$$\|\nabla_{\boldsymbol{\alpha}} f(\boldsymbol{\alpha}_1) - \nabla_{\boldsymbol{\alpha}} f(\boldsymbol{\alpha}_2)\|_F \leq \quad \|M^T M\|_F\|\boldsymbol{\alpha}_1 - \boldsymbol{\alpha}_2\|_2.$$

Now that we have defined M and $Q_{\boldsymbol{\alpha}}$, using proximal algorithm, we can update $\boldsymbol{\alpha}$ as in Equation (16) using the soft thresholding operation as depicted in Section 6.5.2 of [17] for ℓ_1-norm. Specifically, let us define:

$$\boldsymbol{y}(\boldsymbol{\alpha}) = \boldsymbol{\alpha} - \beta \nabla_{\boldsymbol{\alpha}} f(\boldsymbol{\alpha}), \quad (20)$$

then Equation (16) becomes:

$$\boldsymbol{\alpha}^{k+1} = \boldsymbol{prox}_{\beta g}(y(\boldsymbol{\alpha}^k)), \quad (21)$$

and since $g(\boldsymbol{\alpha}) = \gamma\|\boldsymbol{\alpha}\|_1 = \gamma L_1(\boldsymbol{\alpha})$, we have $\boldsymbol{\alpha}^{k+1} = \boldsymbol{prox}_{\beta\gamma L_1}(y(\boldsymbol{\alpha}^k))$ by Equation (21). Using soft thresholding for $\|.\|_1$ we obtain the following update equation:

$$\boldsymbol{\alpha}^{k+1} = \begin{cases} \boldsymbol{y}(\boldsymbol{\alpha}^k) - \beta\gamma & \boldsymbol{y}(\boldsymbol{\alpha}^k) > \beta\gamma, \\ 0 & |\boldsymbol{y}(\boldsymbol{\alpha}^k)| \leq \beta\gamma, \\ \boldsymbol{y}(\boldsymbol{\alpha}^k) + \beta\gamma & \boldsymbol{y}(\boldsymbol{\alpha}^k) < -\beta\gamma. \end{cases}$$

Note that here $y(\boldsymbol{\alpha}^k)$ can be computed as follows using Equations (20) and (18):

$$\boldsymbol{y}(\boldsymbol{\alpha}) = \boldsymbol{\alpha} - \beta(M^T(M\boldsymbol{\alpha} - \text{vec}(\mathcal{X}))). \quad (22)$$

The Overall Algorithm. Putting things together, we obtain a block coordinate algorithm that iteratively updates A, B, C and $\boldsymbol{\alpha}$ as shown in Algorithm 1. The algorithm only requires users to provide a rank upper bound R and will automatically reduce it to $\hat{R}$ when it returns.

One problem remains: while the objective functions in Equations (7)-(10) are convex and so each subproblem is guaranteed to converge to optimum, this does not directly imply that the block coordinate algorithm of Algorithm 1 converges to a stationary point (or local optimum) for any initial point, which is called "global" convergence [18].

Algorithm 1 The Block Coordinate Descent Algorithm

```
Input:
    A third-order tensor $\mathcal{X}$, an upper bound $R$ of rank($\mathcal{X}$),
    Tikhonov regularization parameter $\lambda > 0$,
    $\ell_1$-regularization parameter $\gamma > 0$,
    and the fixed step size $\beta$.
Output:
    An approximated tensor $\hat{\mathcal{X}}$ with an estimated rank $\hat{R}$.
1:  Given initial guess $\mathcal{X}^0 = [A^0, B^0, C^0, \boldsymbol{\alpha}^0]_R$
2:  while convergence criterion is not met do
3:      {Update $A$:}
4:      $E \leftarrow \text{diag}(\boldsymbol{\alpha})(C \odot B)^T$
5:      $A \leftarrow (\mathcal{X}_{(1)}E)/(EE^T + \lambda I)$
6:      {Update $B$:}
7:      $F \leftarrow \text{diag}(\boldsymbol{\alpha})(C \odot A)^T$
8:      $B \leftarrow (\mathcal{X}_{(2)}F)/(FF^T + \lambda I)$
9:      {Update $C$:}
10:     $G \leftarrow \text{diag}(\boldsymbol{\alpha})(B \odot A)^T$
11:     $C \leftarrow (\mathcal{X}_{(3)}G)/(GG^T + \lambda I)$
12:     {Update $\boldsymbol{\alpha}$:}
13:     for $r = 1$ to $R$ do
14:         $M[:, r] \leftarrow \text{vec}(\boldsymbol{a}_r \circ \boldsymbol{b}_r \circ \boldsymbol{c}_r)$
15:     end for
16:     $\boldsymbol{y} \leftarrow \boldsymbol{\alpha} - \beta(M^T(M\boldsymbol{\alpha} - \text{vec}(\mathcal{X})))$
17:     for $r = 1$ to $R$ do
18:         $\boldsymbol{\alpha}[r] \leftarrow \begin{cases} \boldsymbol{y}[r] - \beta\gamma & \boldsymbol{y}[r] > \beta\gamma \\ 0 & |\boldsymbol{y}[r]| \leq \beta\gamma \\ \boldsymbol{y}[r] + \beta\gamma & \boldsymbol{y}[r] < -\beta\gamma \end{cases}$
19:     end for
20: end while
21: $\hat{R} \leftarrow$ the number of non-zero elements of $\boldsymbol{\alpha}$
22: Make $A$, $B$, $C$ and $\boldsymbol{\alpha}$ compact by removing from them
    those $\boldsymbol{a}_r$, $\boldsymbol{b}_r$, $\boldsymbol{c}_r$ and $\alpha_r$ where $\alpha_r = 0$
23: Tensor $\hat{\mathcal{X}}$ can be constructed with factor matrices $A, B, C$
    and coefficients $\boldsymbol{\alpha}$.
```

We set $\beta^k = 0.99/Q^k_{\boldsymbol{\alpha}}$ in each iteration k in order to ensure global convergence. Note that we set step size β^k to be close to its upper bound to allow faster convergence, and this is step size is also larger than $1/Q_{\boldsymbol{\alpha}}$.

In Algorithm 1, we should actually compute β from M right after Line 15. We were treating β as fixed in order to simplify the notation in our presentation.

V. Experiments

In this section, we test our algorithm on tensors with different rank and dimensions. We find that setting the ℓ_1-regularization parameter γ to be large promotes sparsity of $\boldsymbol{\alpha}$ but does not compromise residual error $\|\mathcal{X} - \hat{\mathcal{X}}\|_F$ for a moderate-sized tensor (i.e., I, J, K does not have to be large), and thus we set $\gamma = 50$ by default. Among other parameters, the Tikhonov regularization parameter $\lambda = 10^{-2}$ and the step size $\beta^k = 0.99/Q^k_{\boldsymbol{\alpha}}$ in each iteration k. We use these hyperparameter values since they work well in all our cases, though they can be further tuned to achieve better decomposition quality and rank approximation.

TABLE I
Rank Approximation

	Size of Tensor		
	$I, J, K = 5$	$I, J, K = 7$	$I, J, K = 10$
Actual Rank	5	8	10
Upper bound	10	15	20
Mean of Estimated Ranks	5	8.8	14.25

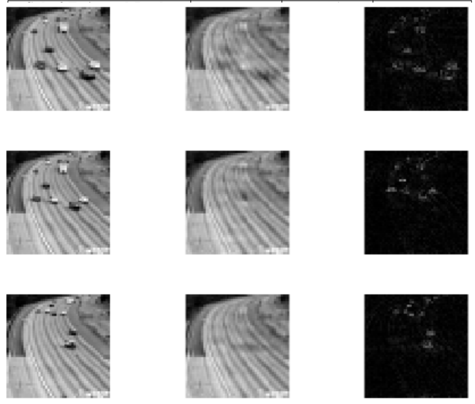

Fig. 1. Separating Vehicle from the Highway Background

A. Quality of Tensor Rank Estimate

We randomly generate synthetic cubic tensors of a specific rank R by summing R random rank-one tensors.

Table I shows the results for three experimental settings with different tensor dimensions and ranks:

- A $5 \times 5 \times 5$ tensor generated with 5 rank-one components;
- A $7 \times 7 \times 7$ tensor generated with 8 rank-one components;
- A $10 \times 10 \times 10$ tensor generated with 10 rank-one components.

B. Experiments on Real Videos and Images

We test the effectiveness of our algorithm in two real applications. The first one is video foreground/background seperation, where we stack grayscale video frames into a 3D tensor, and perform CP decomposition over it. Since objects moving in the foreground are different frame by frame, the CP components should mainly capture the shared background signal. Once we reconstruct the background tensor $\hat{\mathcal{X}}$ from the components, the moving objects in the foreground can be obtained by subtracting each original video frame by the corresponding background frame. This allows us to use CP decomposition to implement video foreground/background separation (into 2 separate videos).

We use a video about moving vehicles on a highway with 51 frames of size 48×48, giving a tensor of size $48 \times 48 \times 51$. After running our algorithm, the CP decomposition gets an estimated rank of 23, i.e., with 23 components which is much smaller than the dimensions (i.e., 48 and 51), demonstrating that the tensor is compactly approximated. Figure 1 illustrates the quality of foreground/background separation: Columns 1, 2 and 3 show the original frames, the reconstructed background, and the foreground (moving objects), respectively; and Rows 1, 2 and 3 refer to 3 sampled frames to display: Frames 11, 16 and 49. Note that moving vehicles are properly separated from background.

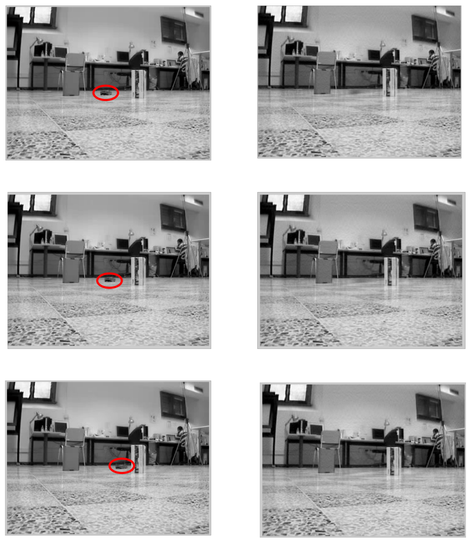

Fig. 2. Separating a Car from the Room Background

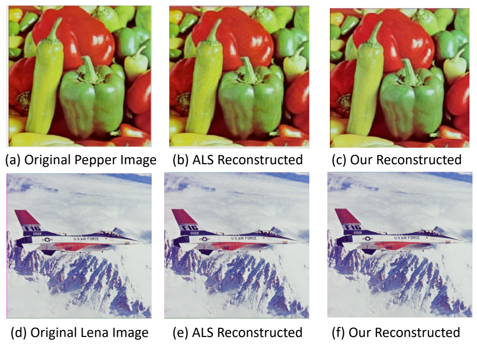

Fig. 3. Image Compression and Reconstruction

As another video example, Figure 2 is about a toy car moving in a room, and the goal is to separate the car from the background. The video-frame tensor is of size $240 \times 320 \times 500$ (i.e., 500 frames each of size 240×320). After running our algorithm, the CP decomposition gets an estimated rank of 90. In Figure 2, Columns 1 and 2 show the original frames and the reconstructed background, respectively; and Rows 1, 2 and 3 refer to 3 sampled frames to display: Frames 25, 30 and 40. We can see that the moving car is properly removed from the background.

Finally, we consider image compression. Obviously, with rank approximation, our algorithm obtains a more compact representation of an image than the conventional CP decomposition since components with α_i being almost 0 are eliminated. It remains to see if this benefit is a result of trading off image quality after decomposition.

To see this, Figure 3 shows two images as well as the reconstructed ones compressed by the conventional ALS and by our algorithm. The pepper image is $200 \times 200 \times 3$ and the estimated rank is 90, while the airplane one is $512 \times 512 \times 3$ and the estimated rank is 200. We can see that our reconstructed image is similar to that from the conventional ALS.

VI. Conclusion

This paper proposed a new block coordinate descent algorithm that sparsifies and thus reduces the number of components in CP decomposition. This algorithm alleviates users burden to guess a proper tensor rank before the decomposition, and obtains an accurate estimate of the tensor rank. We tested our algorithm over real video and image data; the results verify that our approach leads to both an accurate rank estimate and high-quality components for tensor reconstruction.

References

[1] E. Acar, S. A. Çamtepe, M. S. Krishnamoorthy, and B. Yener. Modeling and multiway analysis of chatroom tensors. In *International Conference on Intelligence and Security Informatics*, pages 256–268. Springer, 2005.
[2] M. Astrid and S. Lee. Cp-decomposition with tensor power method for convolutional neural networks compression. In *2017 IEEE International Conference on Big Data and Smart Computing, BigComp 2017, Jeju Island, South Korea, February 13-16, 2017*, pages 115–118, 2017.
[3] J. Bolte, S. Sabach, and M. Teboulle. Proximal alternating linearized minimization for nonconvex and nonsmooth problems. *Math. Program.*, 146(1-2):459–494, 2014.
[4] E. J. Candès and T. Tao. Decoding by linear programming. *IEEE Trans. Information Theory*, 51(12):4203–4215, 2005.
[5] E. J. Candes, M. B. Wakin, and S. P. Boyd. Enhancing sparsity by reweighted l1 minimization. *Journal of Fourier analysis and applications*, 14(5-6):877–905, 2008.
[6] J. D. Carroll and J.-J. Chang. Analysis of individual differences in multidimensional scaling via an n-way generalization of "eckart-young" decomposition. *Psychometrika*, 35(3):283–319, 1970.
[7] V. de Silva and L. Lim. Tensor rank and the ill-posedness of the best low-rank approximation problem. *SIAM J. Matrix Analysis Applications*, 30(3):1084–1127, 2008.
[8] M. De Vos, L. De Lathauwer, B. Vanrumste, S. Van Huffel, and W. Van Paesschen. Canonical decomposition of ictal scalp eeg and accurate source localisation: Principles and simulation study. *Computational intelligence and neuroscience*, 2007, 2007.
[9] S. Foucart and H. Rauhut. A mathematical introduction to compressive sensing. *Bull. Am. Math*, 54:151–165, 2017.
[10] R. A. Harshman et al. Foundations of the parafac procedure: Models and conditions for an" explanatory" multimodal factor analysis. 1970.
[11] C. J. Hillar and L. Lim. Most tensor problems are np-hard. *J. ACM*, 60(6):45:1–45:39, 2013.
[12] F. L. Hitchcock. The expression of a tensor or a polyadic as a sum of products. *Journal of Mathematics and Physics*, 6(1-4):164–189, 1927.
[13] F. L. Hitchcock. Multiple invariants and generalized rank of a p-way matrix or tensor. *Journal of Mathematics and Physics*, 7(1-4):39–79, 1928.
[14] T. G. Kolda and B. W. Bader. Tensor decompositions and applications. *SIAM Review*, 51(3):455–500, 2009.
[15] D. Kressner, M. Steinlechner, and B. Vandereycken. Low-rank tensor completion by riemannian optimization. *BIT Numerical Mathematics*, 54:447–468, 2014.
[16] N. Li, S. Kindermann, and C. Navasca. Some convergence results on the regularized alternating least-squares method for tensor decomposition. *Linear Algebra and its Applications*, 438(2):796–812, 2013.
[17] N. Parikh and S. P. Boyd. Proximal algorithms. *Foundations and Trends in Optimization*, 1(3):127–239, 2014.
[18] B. K. Sriperumbudur and G. R. G. Lanckriet. On the convergence of the concave-convex procedure. In *NIPS*, pages 1759–1767, 2009.
[19] R. Tomioka and T. Suzuki. Convex tensor decomposition via structured schatten norm regularization. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 1331–1339. Curran Associates, Inc., 2013.
[20] Q. Yao, J. T.-Y. Kwok, and B. Han. Efficient nonconvex regularized tensor completion with structure-aware proximal iterations. In *PMLR*, volume 97, pages 7035–7044, 2019.