Sparse Non-Negative Tensor Factorization Using Columnwise Coordinate Descent

Ji Liu, Jun Liu, Peter Wonka, and Jieping Ye

Department of Computer Science and Engineering, Arizona State University, Tempe, AZ, 85287.

Abstract

Many applications in computer vision, biomedical informatics, and graphics deal with data in the matrix or tensor form. Non-negative matrix and tensor factorization, which extract data-dependent non-negative basis functions, have been commonly applied for the analysis of such data for data compression, visualization, and detection of hidden information (factors). In this paper, we present a fast and flexible algorithm for sparse non-negative tensor factorization (SNTF) based on columnwise coordinate descent (CCD). Different from the traditional coordinate descent which updates one element at a time, CCD updates one column vector simultaneously. Our empirical results on higher-mode images, such as brain MRI images, gene expression images, and hyperspectral images show that the proposed algorithm is 1-2 orders of magnitude faster than several state-of-the-art algorithms.

Key words:
Sparse, Nonnegative, Tensor Factorization, Columnwise Coordinate Descent

1. Introduction

Non-negative matrix and tensor factorization (NMF/NTF) aim to extract data-dependent non-negative basis functions [13, 6, 20, 27], so that the target data can be expressed by the linear or multi-linear combination of these non-negative components. They have been commonly applied for the analysis of such data for data compression, visualization, and detection of hidden information (factors), e.g., in face recognition [24], psychometric [19] and image analysis [25]. Additionally, the basis can be constrained to be sparse which typically leads to an even more meaningful decomposition of the data. As a result, many researchers focused on

sparse non-negative matrix factorization (SNMF) [13, 14, 4, 9] in the past few years.

A tensor, as a more general "matrix", can be used to express more complicated intrinsic structures of higher-mode data. Thus, sparse non-negative tensor factorization (SNTF) is a natural extension of the SNMF problem. Recently, SNTF began to receive more attention. It is used in high-mode medical data analysis [18], psychometric [19], etc. The SNTF problem is not as well studied as the matrix case. In comparison with SNMF, SNTF has two additional challenges. First, a tensor usually contains a much larger data set than a matrix, thus SNMF needs to pay more attention to computing efficiency than other factors.

The other challenge lies in how to deal with the so-called "core tensor" in SNMF. Because of the special structure, tensor factorization always contains implicitly or explicitly a core tensor, which does not exist in matrix factorization. How to efficiently and effectively deal with it is one key problem in SNTF. We can either fix it as an identity [18], or incorporate it into the optimization procedure [17]. The former approach is not flexible in handling the unbalanced target tensor data, while the latter one is computationally very expensive, which makes it unsuitable for large high-mode tensor data.

In this paper, we propose a fast and flexible SNTF algorithm, which iteratively updates one basis at a time. We employ the idea of coordinate descent (CD) for the updating. CD has recently been shown to be very efficient in solving the sparse learning problem [22]. It updates one element of the basis at a time and the algorithm cycles through all elements until convergence. The key to its high efficiency lies in the closed-form solution for each update. In the proposed algorithm, we identify the independent groups of elements among different bases, and update at one time one column vector which consists of elements from all bases, rather than one element from one group. We call it "columnwise coordinate descent" (CCD). In addition, we design a flexible way to deal with the core tensor problem mentioned above. We apply the proposed algorithms to three types of higher-mode images such as brain MRI images, gene expression pattern images, and hyperspectral images. Our experiments show that the proposed algorithm is 1-2 orders of magnitude faster than several recent SNMF and SNTF algorithms while achieving the same objective value.

The rest of the paper is structured as follows: the related work is presented in section 2; section 3 introduces the proposed CCD algorithm; we present the experimental results in section 4; finally, we conclude this paper in section 5.

2. Related Work

Since Lee and Seung [13] started a flurry of research on non-negative matrix factorization (NMF), this field received broad attention. In addition to the use of different objective functions such as the least squares [4] and Kullback Leibler [13], the main difference among various algorithms lies in the update rule. The update rule directly influences the convergence speed and the quality of the factorization. The multiplicative update rule proposed by Lee and Seung [14] was considered to be the classical one, although its convergence speed was quite slow. Gonzalez and Zhang [7] used an interior-point gradient method to accelerate the multiplicative update. Recently, a quasi-Newton optimization approach was employed as the update rule by Zdunedk and Cichocki [28]. Lin [15] employed a projected gradient bound-constrained optimization method which has better convergence properties than the multiplicative update. Recently, Kim and Park [10] proposed a novel formulation of sparse non-negative matrix factorization (SNMF) and used alternating non-negativity-constrained least squares for the computation. Their results showed that it achieved better clustering performance with a lower computational cost than other existing NMF algorithms. See [1] for a more detailed review on NMF.

Tensor factorization is a natural extension of matrix factorization. It has been applied successfully in face recognition [24], psychometric [19], and image analysis [25]. Two popular models have been studied for tensor factorization including the parafac model [8, 3] and the tucker model [23]. Most work focus on the parafac model, since it is simpler and easier to understand from the matrix perspective. Welling, M. and Weber, M. [26] proposed a non-negative tensor factorization algorithm based on the multiplicative updating rule [14], a natural extension of matrix factorization; Kim et al. [11] proposed a non-negative tensor factorization algorithm based on the alternating large-scale non-negativity constrained least squares; A non-negative sparse factorization algorithm using Kullback-Leibler divergence [13] was introduced by FitzGerald et al. [20]; Mørup et al. [16] proposed a SNTF algorithm based on the parafac model, which employs the L_1 norm penalty as the sparseness constraint like ours, and applies the multiplicative updating rule like most other algorithms [6, 20]. They further employ over relaxed bound optimization strategy to accelerate the computing speed; Recently, Cichocki et al. [5] presented an algorithm using alpha and beta divergences.

It is interesting to note that the parafac model is just a specific example of the tucker model when the core tensor is fixed to be an identity. A key issue in applying the tucker model is how to construct the core tensor. Lathauwer et al. [12]

presented the well-known HOSVD (Higher-Order Singular Value Decomposition) factorization based on the SVD algorithm for matrices. Without a non-negative requirement, it forced all factors to be orthogonal so that the core tensor could be computed through a unique and explicit expression. Bro and Andersson [2] implemented a non-negative tucker model factorization, but the core tensor was not guaranteed to be non-negative. Recently, Mørup et al. [17] proposed a non-negative sparse tensor factorization algorithm, which incorporated the core tensor into the optimization. However, the optimization process for the core tensor dominates the computational cost per iteration, resulting in overloaded computing time for convergence.

3. Algorithm

We detail the proposed algorithm in this section. We first introduce the SNMF problem in section 3.1. Then, the more general SNTF problem is presented in section 3.2. We handle the core tensor issue in section 3.3.

3.1. SNMF

SNMF approximates a matrix A by a matrix $\hat{A}$ so that $\hat{A}$ is the product of two matrices, i.e.,

$$A \sim \hat{A} = MN^T.$$

We formulate the SNMF problem as the following optimization:

$$\begin{aligned} \min_{M,N} : \quad & \frac{1}{2}\|A - MN^T\|^2 + \lambda_1|M| + \lambda_2|N| \\ s.t. \quad & M \geq 0, N \geq 0 \end{aligned} \tag{1}$$

where $A \in \mathbb{R}^{p\times q}$, $M \in \mathbb{R}^{p\times r}$ and $N \in \mathbb{R}^{q\times r}$. Here, the L_1 norm penalty $\lambda_1|M| + \lambda_2|N|$ is used as the sparseness constraint. Without loss of generality, we can express the target matrix A as $A \approx MIN^T$, where I is an identity matrix with the rank r. It is worthwhile to point out that I can be considered as a core tensor for the matrix $\hat{A}$, when $\hat{A}$ is regarded as a 2-mode tensor. We will discuss the more general tensor case in the next subsection.

3.2. SNTF

Before formulating the SNTF problem, we first introduce some tensor fundamentals and notations. Tensors are multilinear mappings over a set of vector

spaces. An N-mode tensor is defined as $\mathcal{A} \in \mathbb{R}^{I_1 \times I_2 \times \ldots I_N}$. Its elements are denoted as $a_{i_1 i_2 \ldots i_N}$, where $1 \leq i_n \leq I_n, 1 \leq n \leq N$. For instance, a vector is a 1-mode tensor and a matrix is a 2-mode tensor. It is sometimes convenient to unfold a tensor into a matrix. The unfolding of a tensor $\mathcal{A}$ along the nth mode is defined as

$$A_{(n)} \in \mathbb{R}^{I_n \times (I_1 \ldots I_{n-1} I_{n+1} \ldots I_N)}.$$

The mode-n product of a tensor $\mathcal{A} \in \mathbb{R}^{I_1 \times I_2 \times \ldots I_N}$ and a matrix $U \in \mathbb{R}^{J \times I_n}$ is denoted by $\mathcal{A} \times_n U$. Its result is still a N-mode tensor $\mathcal{B} \in \mathbb{R}^{I_1 \ldots \times I_{n-1} \times J \times I_{n+1} \ldots \times I_N}$. Its elements are defined as:

$$b_{i_1 \ldots i_{n-1} j i_{n+1} \ldots i_N} = \sum_{i_n} a_{i_1 \ldots i_{n-1} i_n i_{n+1} \ldots i_N} u_{j i_n} \tag{2}$$

The mode-n product $\mathcal{B} = \mathcal{A} \times_n U$ can be computed via the matrix multiplication $B_{(n)} = U A_{(n)}$ followed by a "fold" operation along the nth mode. The "fold" operation is defined as $fold_n(B_{(n)}) = \mathcal{B}$. This paper uses

$$\|\mathcal{A}\| = (\sum_{i_1, i_2, \ldots i_N} |a_{i_1, i_2, \ldots i_N}|^2)^{\frac{1}{2}}$$

as the Frobenious norm of a tensor and $|\mathcal{A}| = \sum_{i_1, i_2, \ldots i_N} (|a_{i_1, i_2, \ldots i_N}|)$ as the L_1 norm.

Next, we formally formulate the SNTF problem. Like SNMF, the goal of SNTF is to search for the factorization of a tensor $\hat{\mathcal{A}}$ to approximate the target tensor $\mathcal{A}$. For convenience, let us first consider a simple example where the mode of $\hat{\mathcal{A}}$ is 2 , i.e. $\hat{A}$ is a matrix. According to the definition of SNMF, $\hat{\mathcal{A}}$ can be factored as $\hat{\mathcal{A}} = MN^T$ or $\hat{\mathcal{A}} = MIN^T$. Using the tensor notation, $\hat{\mathcal{A}}$ is denoted as $\hat{\mathcal{A}} = I \times_1 M \times_2 N$. Here, the identity I plays the role of the core tensor. Through a simple generalization, when the mode of the target tensor is greater than 2, the approximate tensor $\hat{\mathcal{A}}$ can be factored as

$$\hat{\mathcal{A}} = \mathcal{C} \times_1 U_1 \times_2 \ldots \times_N U_N,$$

where $\mathcal{C}$ is an identity tensor whose elements are 1 in the diagonal and 0 otherwise. Similar to equation (1), the SNTF problem can be described as the following optimization problem:

$$\begin{aligned} \min_{U_1, \ldots, U_N} \quad & \frac{1}{2} \|\mathcal{A} - \mathcal{C} \times_1 U_1 \times_2 \ldots \times_N U_N\|^2 + \sum_{1 \leq n \leq N} \lambda_n |U_n| \\ s.t. \quad & U_n \geq 0, 1 \leq n \leq N \end{aligned} \tag{3}$$

where $\hat{\mathcal{A}} \in \mathbb{R}^{I_1 \times I_2 \times \ldots I_N}$ and $U_n \in \mathbb{R}^{I_n \times J_n}$ for all $1 \leq n \leq N$. Since the core tensor $\mathcal{C}$ is assumed to be an identity tensor, $J_1 = J_2 = \ldots = J_N$. The L_1 norm penalty in the objective function forces U_n to be sparse.

We propose to solve the optimization problem in Eq. (3) iteratively by updating one part at a time with all other parts fixed. For example, if we fix $U_1, ...U_{n-1}$, $U_{n+1}, ..., U_N$ and search for the optimal U_n^*, we obtain the following optimization sub-problem:

$$\begin{aligned} \min_{U_n} : \quad & \frac{1}{2}\|\mathcal{A} - \mathcal{C} \times_1 U_1 ... \times_N U_N\|^2 + \lambda_n |U_n| \\ s.t. \quad & U_n \geq 0. \end{aligned} \tag{4}$$

Since $\|\mathcal{A} - \mathcal{C} \times_1 U_1 \ldots \times_N U_N\| = \|A_{(n)} - U_n(\mathcal{C} \times_1 U_1 \ldots \times_{n-1} U_{n-1} \times_{n+1} \ldots \times_N U_N)_{(n)}\|$, the problem is equal to the following one:

$$\begin{aligned} \min_{U_n} : \quad & \frac{1}{2}\|A_{(n)} - U_n B_{(n)}\|^2 + \lambda_n |U_n| \\ s.t. \quad & U_n \geq 0 \end{aligned} \tag{5}$$

where

$$B_{(n)} = (\mathcal{C} \times_1 U_1 \ldots \times_{n-1} U_{n-1} \times_{n+1} \ldots \times_N U_N)_{(n)}.$$

We can further simply the optimization problem in Eq. (5), by taking the transpose and separating the equations into the In columns of the matrix U_n^T. resulting in I_n independent optimization problems:

$$\begin{aligned} \min_{u_i} : \quad & \frac{1}{2}\|B_{(n)}^T u_i - a_i\|^2 + \lambda_n |u_i| \\ s.t. \quad & u_i \geq 0, \end{aligned} \tag{6}$$

where

$$\begin{aligned} A_{(n)} &= [a_1, a_2, ..., a_{I_n}]^T, \\ U_n &= [u_1, u_2, \ldots, u_{I_n}]^T. \end{aligned}$$

For this convex but not differentiable optimization problem, a coordinate descent (CD) method can be applied to find the global minimum [22]. The basic idea of CD is to optimize one element at a time while fixing other elements by decomposing the problem in the Eq. (6) into several sub-problems as:

$$\begin{aligned} \min_{u_{ij}} : \quad & \frac{1}{2}\|B_{(n)}^T u_i - a_i\|^2 + \lambda_n |u_{ij}| \\ s.t. \quad & u_{ij} \geq 0 \end{aligned} \tag{7}$$

where $u_i = [u_{i1}, \ldots, u_{ij}, \ldots, u_{iJ_n}]^T$. Since the objective function is not differentiable, we need to compute its subdifferential to search for its optimum. The subdifferential of a function $f(x)$ at a point x is defined as

$$\partial f(x) = \{d | f(y) \geq f(x) + < d, y - x >, \forall y\}.$$

In general, the subdifferential at a particular point is a set rather than a single value. If the function $f(x)$ is differentiable at the point $\hat{x}$, then the differential value $f'(\hat{x})$ is the unique element in its subdifferential. If x^* is a global optimum, then $0 \in \partial f(x^*)$ must be satisfied. For the problem in Eq. (7), we compute the subdifferential of the objective function as follows:

$$\partial f(u_{ik}) = \begin{cases} \{b_k(B_{(n)}^T u_i - b_k^T a_i) + \lambda_n\}, & u_{ik} > 0 \\ \{b_k(B_{(n)}^T u_i - b_k^T a_i) - \lambda_n\}, & u_{ik} < 0 \\ [b_k(B_{(n)}^T u_i - b_k^T a_i) - \lambda_n, & \\ b_k(B_{(n)}^T u_i - b_k^T a_i) + \lambda_n], & u_{ik} = 0 \end{cases} \tag{8}$$

where

$$B_{(n)} = [b_1^T, \ldots, b_j^T, \ldots, b_{J_N}^T]^T.$$

Eq. (8) distinguishes between three cases. In the first two cases, the function is differentiable at point u_{ik} and the subdifferential corresponds to the gradient. In the third case the subdifferential is a closed interval. We then look for the optimal u_{ij}^* which satisfies $0 \in \partial f(u_{ij}^*)$. We can obtain the solution in a closed-form as follows:

$$u_{ij}^* = \begin{cases} \frac{t-\lambda_n}{b_k b_j^T}, & t > \lambda_n \\ \frac{t+\lambda_n}{b_k b_j^T}, & t < \lambda_n \\ 0, & \text{otherwise} \end{cases} \tag{9}$$

where

$$t = b_j(B_{(n)}^T u_i - b_j^T a_i) - b_j u_{ij}. \tag{10}$$

If we further force the non-negative constraint $u_{ij} \geq 0$, the optimum solution can be computed as follows:

$$u_{ij}^* = \begin{cases} \frac{t-\lambda_n}{b_j b_j^T}, & t > \lambda_n \\ 0, & \text{otherwise} \end{cases} \tag{11}$$

The coordinate descent method discussed above optimizes the objective function in Eq. (5) by one element at a time where the formulation for each individual

element is given by Eq. (7). This is illustrated in the first row of Fig. (1). A big advantage of our derivation is that several variables are independent and can be updated simultaneously. The optimization formulation in Eq. (6) considers a row of U_n at the time, but all elements in a row are dependent. Therefore, it seems that no more simplification is possible. However, the trick is to realize that all of the n^{th} coordinates in each of the I_n rows of U_n are independent. Therefore, we can update one column vector at the time. That means we simultaneously work on I_n individual elements, one from each of the row equations shown in Eq. (6). We call the algorithm "columnwise coordinate descent" (CCD), which updates all elements in one column together as shown in the second row of Fig.(1). Our empirical results show that CCD is 1-2 orders of magnitude faster than CD. The detailed CCD updating procedure is shown in **Algorithm** 1.

Algorithm 1 Columnwise Coordinate Descent Updating

Input: $A_{(n)}, B_{(n)}, \lambda_n$

Output: U_n

1: Set $M = B_{(n)}B_{(n)}^T$;
2: Set $N = A_{(n)}B_{(n)}^T - \lambda_n$;
3: Set $D = [M(1,1)\ M(2,2)\ \ldots\ M(J_n, J_n)]$;
4: Set $M(j,j) = 0$ for all $1 \leq j \leq J_n$;
5: **while** not convergent **do**
6: **for** $j = 1$ to J_n **do**
7: Updating $U_n(:,j) = \max\left(0,\ \frac{N(:,j) - U_n M}{D(j)}\right)$
8: **end for**
9: **end while**

The computational complexity is presented in Tab. 1 (considering only the "multiplication" and "addition" operations). We defined K_I as the maximal iteration number for the iteration in step 5 to 9 in **Algorithm** 1 and K_O as the running number of Algorithm 1.

In our experience, $K_I \leq 100$ and $K_O \leq 300$ in general. One can see that if the mode number N is large, $\prod_{i \neq n} I_i \gg K_I I_n$ such that the main computational load lies in Step 1 and Step 2. The computational complexity in Step 1 and 2 has the same asymptotic complexity as an iteration in the state-of-the-art tensor/matrix factorization algorithms [18, 14]. However, our algorithm converges much faster than the state-of-the-art tensor/matrix factorization algorithms (see the empirical comparison in section 4). That is why our algorithm is more efficient especially

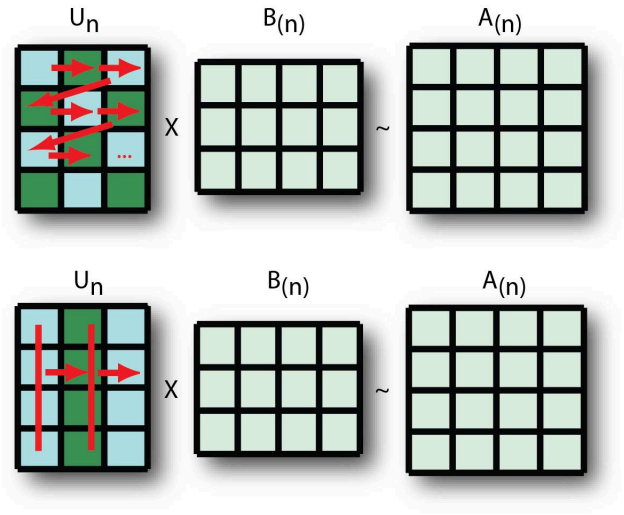

Figure 1: Comparison of CD Updating (top row) and CCD Updating (bottom row).

Method	Complexity
Step 1	$O(J_n^2 \prod_{i \neq n} I_i)$
Step 2	$O(J_n \prod_{i=1}^{N} I_i)$
Step 3	0
Step 4	0
Step 5-9	$O(K_I I_n J_n^2)$
Algorithm 1	$O(K_I I_n J_n^2) + O(J_n \prod_{i=1}^{N} I_i)$
Total	$O(K_O K_I \sum_{i=1}^{N} I_i J_i^2) + O(K_O \sum_{i=1}^{N} I_i \prod_{i=1}^{N} I_i)$

Table 1: Complexity analysis

in the large scale data.

3.3. Non-Identity Core Tensor

In the exposition above we assume that the core tensor is an identity following the SNMF structure. However, forcing the core tensor to be an identity may not work in some situations. For instance, when the target tensor is very unbalanced, say, its mode is $1000 \times 1000 \times 15 \times 3$ in size, then choosing an identity core tensor will create a dilemma. A high-mode core tensor (say, $20 \times 20 \times 20 \times 20$) obviously leads to redundant computing; a low-mode core tensor (say, $3 \times 3 \times 3 \times 3$) may result in a high error. Thus, it is essential to construct a non-identity core tensor to deal with unbalanced tensors. In the following, we describe how to establish a non-identity core tensor.

We set out to create a core tensor using the requirements described below. This will lead to a core tensor that is similar to an identity matrix.

1. It consists of "0" and "1" elements.
2. Any two slices of the core tensor along any mode must be orthogonal. In other word, all rows of $C_{(n)}$ must be orthogonal.
3. It does not decrease the rank of $\hat{A}_(n)$. In other word, for any n, $C_{(n)}$ is of full row rank: $rank(C_{(n)}) = J_n$. Here, we assume that $J_n \leq \prod_{k \neq n} J_k$ for any n.

It is clear that a tensor designed using the guidelines above is not unique. We propose to automatically generate a core tensor fulfilling all three conditions above.

Since the core tensor contains two types of elements: "1" and "0" only, we use a matrix to store all locations of 1's. The matrix can be expressed as $S = [\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_N]$ or $[\mathbf{r}_1^T, \mathbf{r}_2^T, \ldots, \mathbf{r}_M^T]^T$.

Each row of the matrix S corresponds to an element in the core tensor which has a value of "1". For example, if there is a row of the matrix given by $\mathbf{r}_m = [j_1, j_2, \ldots, j_N]$, then the $(j_1, j_2, \ldots, j_N)$-th element of the core tensor is 1. The condition 2 requires that any two rows of the matrix S have at least two different elements. To fulfill the condition 3, the n-th column $\mathbf{c}_n$ of the matrix S contains all integers from 1 to J_n. The pseudo-code for the core tensor estimation is given in **Algorithm** 2.

Under the C-filling rule, all combinations fill the matrix in a circular fashion, while under the P-filling rule, all combinations fill the matrix in a piecewise fashion. To illustrate the algorithm, We show an example in Fig.2. The mode size is given as $2 \times 3 \times 4 \times 8$. Since $2 \times 3 \leq 8 \leq 2 \times 3 \times 4$, the first two columns should

Algorithm 2 Core Tensor Estimation

Input: $J_1, J_2, \ldots, J_N$

Output: S

1: Sort $J_1, J_2, \ldots, J_N$ in increasing order. Without loss of generality, we assume $J_1 \leq J_2 \leq \ldots \leq J_N$;
2: Set $S = [0]_{J_N \times N}$;
3: Set $S(:, j) = 1 : J_n$;
4: Find n satisfying $M = \prod_{i=1}^{n-1} J_i \leq J_N \leq \prod_{i=1}^{n} J_i$;
5: **if** $J_n \geq M$ **then**
6: Fill $S(:, n)$ using the C-filling rule
7: Fill $S(:, 1 : n-1)$ using the P-filling rule
8: **else**
9: Fill $S(:, n)$ using the P-filling rule
10: Fill $S(:, 1 : n-1)$ using the C-filling rule
11: **end if**
12: Fill $S(:, n+1 : N)$ using the C-filling rule

be packed together. Since $2 \times 3 \geq 4$, the first two columns are filled using the C-filling rule and the third column follows the P-filling rule.

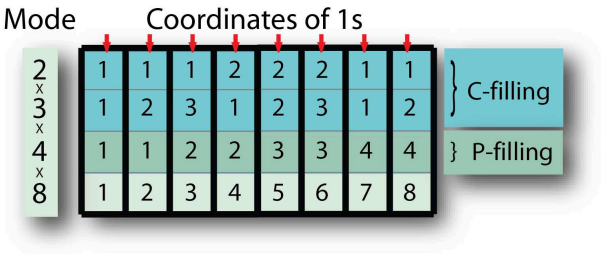

Figure 2: Illustration of C-filling and P-filling.

4. Empirical Evaluation

In this section, we evaluate the proposed algorithms using three different types of higher-mode images including brain MRI images, gene expression images, and hyperspectral images. All algorithms were implemented in Matlab version 7.6.0 and all tests were performed on an Intel Core 2 2.0Hz and 3GB RAM computer.

Mode size = $20 \times 20 \times 20$ $\lambda_1 = \lambda_2 = \lambda_3 = 0.5$

Method	Iter.	Comp.	Spar.	Time	Error
Parafac	211	**0.0016**	0.18532	192.875	0.1451
Tucker	721	0.0027	0.4632	4327.75	**0.1034**
T-CCD	**49**	**0.0016**	**0.5080**	**94.0803**	0.1402

Mode size = $40 \times 40 \times 40$ $\lambda_1 = \lambda_2 = \lambda_3 = 0.5$

Method	Iter.	Comp.	Spar.	Time	Error
Parafac	227	**0.0033**	0.1856	334.145	0.0877
Tucker	-	0.0123	-	>6000	-
T-CCD	**54**	**0.0033**	**0.5365**	**136.876**	**0.0809**

Mode size = $60 \times 60 \times 60$ $\lambda_1 = \lambda_2 = \lambda_3 = 0.5$

Method	Iter.	Comp.	Spar.	Time	Error
Parafac	233	**0.0049**	0.2539	440.370	0.0748
Tucker	-	0.0353	-	>6000	-
T-CCD	**61**	**0.0049**	**0.5844**	**198.394**	**0.0651**

Table 2: Comparison of Parafac, Tucker, and T-CCD on the brain MRI data: Iter. = Iteration number; Comp. = Compression ratio; Spar. = Sparseness ratio, indicating the percentage of zero values in the factors; Time = Running Time; Error = Error ratio defined as $\|\mathcal{A} - \hat{\mathcal{A}}\|/\|\mathcal{A}\|$.

Core tensor size = $5 \times 5 \times 5$ $\lambda_1 = \lambda_2 = \lambda_3 = 0.1$

Method	Iter.	Comp.	Spar.	Time	Error
Parafac	330	**0.00141**	**0.0802**	29.3842	0.3955
Tucker	46	0.00145	0.0081	29.7751	0.4152
T-CCD	143	**0.00141**	0.0791	29.3618	**0.3935**

Core tensor size = $10 \times 10 \times 10$ $\lambda_1 = \lambda_2 = \lambda_3 = 0.1$

Method	Iter.	Comp.	Spar.	Time	Error
Parafac	330	**0.0027**	0.2392	30.8170	0.3471
Tucker	45	0.0032	0.0567	29.4274	0.3868
T-CCD	130	**0.0027**	**0.2423**	29.6574	**0.3425**

Core tensor size = $15 \times 15 \times 15$ $\lambda_1 = \lambda_2 = \lambda_3 = 0.1$

Method	Iter.	Comp.	Spar.	Time	Error
Parafac	235	**0.0041**	0.1702	29.8741	0.2997
Tucker	42	0.0056	0.0097	30.6174	0.3672
T-CCD	113	**0.0041**	**0.1822**	29.4654	**0.2943**

Table 3: Comparison of Parafac, Tucker, and T-CCD on the EEG data: Iter. = Iteration number; Comp. = Compression ratio; Spar. = Sparseness ratio, indicating the percentage of zero values in the factors; Time = Running Time; Error = Error ratio defined as $\|\mathcal{A} - \hat{\mathcal{A}}\|/\|\mathcal{A}\|$.

4.1. Comparison with Several Recent Algorithms

We compare our proposed sparse non-negative tensor/matrix factorization using columnwise coordinate descent (called "T-CCD"/"M-CCD") against three existing methods, including Parafac [18], Tucker [17], and ALS (alternating least squares) [10]. Note that we name the first two algorithms after the well established tensor structure they use, but all three optimization algorithms stem from recent papers.

Our first experiment compares the proposed tensor factorization algorithm T-CCD against the Parfac and Tucker algorithms. We use a Brain MRI image of size $181 \times 217 \times 181$. We use a core tensor of size $20 \times 20 \times 20$, $40 \times 40 \times 40$, and $60 \times 60 \times 60$, respectively. We apply the same initialization and the same sparseness coefficient ("λ" value in Tab. 2) for all three methods. The tolerance value is fixed at $10^{-5} \times \|A\|^2$. To make a fair comparison, we report the average results over five runs with different initializations. We compare the algorithms in the following aspects:

- running time,
- compression ratio: the memory size of storing the factorization results over the original size of this data,
- sparseness ratio: the percentage of nonzero entries in the factorization result, and
- error ratio: the reconstruction error defined by $\|\mathcal{A} - \hat{\mathcal{A}}/\|\mathcal{A}\|$.

We also compare three algorithms on an open EEG (electroencephalography) data set [1] used in the Parfac and Tucker algorithms [17, 18]. This data of size $64 \times 512 \times 72$ is linearly normalized into the range $[0, 1]$. We report the performance comparison in Tab. 3. Note that different from Tab. 2, we fix the computation time to about 30 seconds for three algorithms in terms of "Error ratio" and "Sparseness ratio".

We can observe from Tab. 2 and 3 that the proposed method outperforms the other two competing algorithms in terms of the convergence speed and the sparseness ratio. A visual comparison of the convergence speed is shown in Fig. 3. The proposed algorithm is 1-2 orders of magnitude faster than other methods to reach

[1] `http://www.erpwavelab.org/`

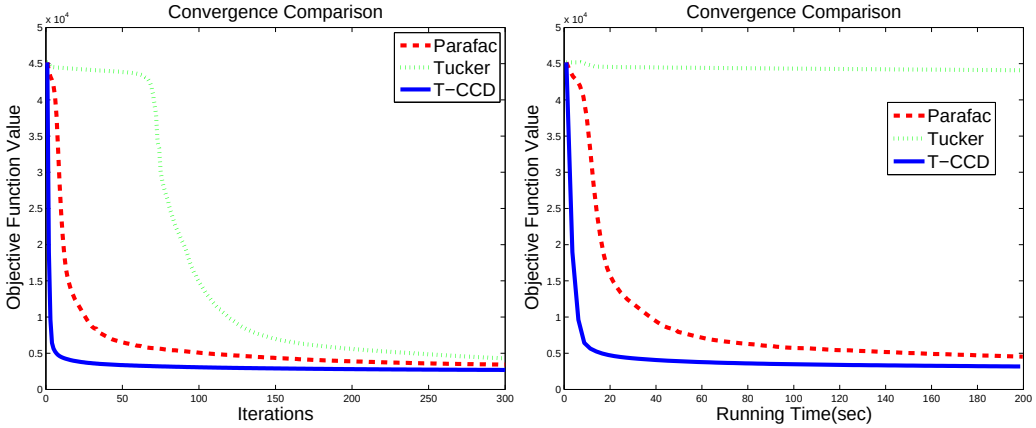

Figure 3: Comparison of all three methods in terms of the decrease of the objective function over iterations (left figure) and time (right figure). The size of the core tensor is set to be $40 \times 40 \times 40$, and $\lambda_1 = \lambda_2 = \lambda_3 = 0.5$. We can observe from the figures that the Tucker algorithm will take orders of magnitude longer to achieve the same value of the objective function.

a certain objective value. Although the Tucker algorithm can achieve a lower error ratio, its computing time and additional storage requirement for the core tensor makes it not competitive for larger data sets. Note that our algorithm can outperform the Tucker algorithm in terms of the error ratio if both algorithms employ the same compression ratio, especially when the the core tensor is unbalanced as verified in the experiment shown in next subsection.

Next, we evaluate the performance of the proposed algorithm for matrix factorization. We compare the algorithms on a *Drosophila* gene expression pattern image data set from the BDGP database[2]. The data matrix is of size 10240×1000. The M-CCD algorithm is compared with the ALS algorithm [10]. Since ALS employs both L_1 norm and L_2 norm in their objective function, it is difficult to compare the objective function directly. Our comparison will focus on the main aspects of the SNMF algorithm including the sparseness ratio, the running time, and the error ratio. The results are summarized in Tab. 4. We can observe from Tab. 4 that (1) the performance of both methods are similar, if the mode size is low; (2) when the mode size become larger, our algorithm is significantly faster

[2] `http://www.fruitfly.org/cgi-bin/ex/insitu.pl`

than ALS. We present in Fig. 4 a visual comparison between these two algorithms.

Core tensor size = 20×20

Method	Iter.	Spar.	Time	Error
ALS	**42**	0.2345	**76.4423**	0.1641
M-CCD	47	**0.3541**	81.3976	**0.1603**

Core tensor size = 40×40

Method	Iter.	Spar.	Time	Error
ALS	**53**	0.3281	312.395	0.1349
M-CCD	59	**0.4219**	**126.812**	**0.1323**

Core tensor size = 80×80

Method	Iter.	Spar.	Time	Error
ALS	**73**	0.4530	1475.39	0.1074
M-CCD	75	**0.4913**	**306.123**	**0.1072**

Table 4: Comparison between ALS and M-CCD using *Drosophila* gene expression pattern images.

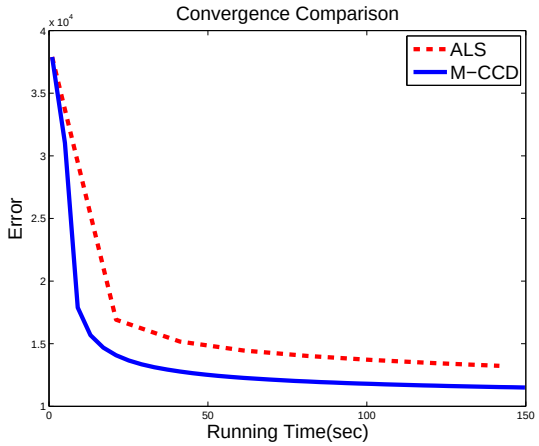

Figure 4: Comparison between M-CCD and ALS algorithm for matrix factorization. The mode size is set to be 80×80.

Data: Hyperspectral image Size = $256 \times 307 \times 11$

Mode	Iter.	Comp.	Spar.	Time	Error
29×29×29	64	0.0193	0.4968	87.3137	0.0892
30×30×10	53	0.0197	0.4897	**56.9032**	0.0884

Data: Brain image Size = $181 \times 217 \times 20$

Mode	Iter.	Comp.	Spar.	Time	Error
48×48×48	77	0.0246	0.5301	95.4573	0.0765
50×50×20	65	0.0244	0.5398	**69.3374**	0.0772

Table 5: Comparison between Balanced Mode and Unbalanced Mode.

4.2. Unbalanced Core Tensor

When the modes of the target tensor are very different, e.g.,

$$\max(I_1, I_2, .., I_N)/\min(I_1, I_2, .., I_N) \gg 1,$$

the proposed algorithm may not perform well if we force all modes of the tensor to a common size. We evaluate the effectiveness of the proposed unbalanced core tensor using the MRI brain image and the "lunar lake" hyperspectral image data[3]. We sub-sample them into images of size $256 \times 207 \times 11$ and $181 \times 217 \times 21$, respectively as the target tensor.

For the first data set, we test two different mode sizes: a balanced size $29 \times 29 \times 29$ and an unbalanced size $30 \times 30 \times 10$. For the second data set, two different mode sizes: $48 \times 48 \times 48$ and $50 \times 50 \times 20$ are used. The results in Tab. 5 show that using an unbalanced mode size requires less running time and iteration number than using a balanced one when achieving a similar compression ratio, sparseness ratio, and error ratio.

Next, we compare the proposed unbalanced core tensor with the one automatically learnt from the Tucker model. Note that the learning of the core tensor can further decrease the error ratio, however, additional time and space are required for the core tensor optimization. In this experiment, we compare T-CCD with the core tensor derived from **Algorithm** 2 with the Tucker model in terms of the error ratio when using the same compression ratio. The results are summarized in Tab. 6. We can observe from the table that the proposed algorithm can achieve a lower error ratio than the Tucker model when using the same compression ratio.

[3] `http://aviris.jpl.nasa.gov/html/aviris.freedata.html`

Data: Hyperspectral image Size = $256 \times 307 \times 11$

Method	Mode	Comp.	Spar.	Time	Error
Tucker	30×30×10	0.030	0.5049	1250	0.0852
T-CCD	46×46×10	0.030	0.4917	**67.32**	**0.0813**

Data: Brain MRI image Size = $181 \times 217 \times 61$

Method	Mode	Comp.	Spar.	Time	Error
Tucker	30×30×10	0.009	0.5187	1834	0.0972
T-CCD	50×50×30	0.009	0.5297	**105.3**	**0.0779**

Table 6: Comparison between Tucker and T-CCD when using the same compression ratio.

4.3. An Application Example on Biological Images

In this experiment, we employ the proposed algorithm (M-CCD) on 5 groups of biological images. The target data in each group consists of 1000 *Drosophila* gene expression pattern images from the BDGP database. The *Drosophila* gene expression pattern images [21] document the spatial and temporal dynamics of gene expression and provide valuable resources for explicating the gene functions, interactions, and networks during *Drosophila* embryogenesis. The images of 5 groups are from stage ranges 4-6, 7-8, 9-10, 11-12, 13-16 of embryo development, respectively. Each image is of size 64×160 and is unfolded into a column vector. The resulting target tensor (matrix) for each group is of size 10240×1000. We apply M-CCD to extract a set of basis images by setting the mode size to be 100×100. We show some sample basis images of stage range 11-12 in Fig. 5. The complete 100 basis images are not shown due to the space constraint. In Fig. 6, we show sample images on the decomposition of an image as a linear combination of several basis images. We are currently working with developmental biologists to analyze the biological significance of the learnt basis images.

5. Conclusion

In this paper, we propose a fast and flexible algorithm for sparse non-negative tensor factorization (SNTF) based on columnwise coordinate descent (CCD). Different from the traditional coordinate descent, CCD updates one column vector simultaneously, resulting in a significant reduction in the computation time. Our empirical results on brain MRI images, gene expression images, and hyperspectral images show that the proposed algorithm is 1-2 orders of magnitude faster

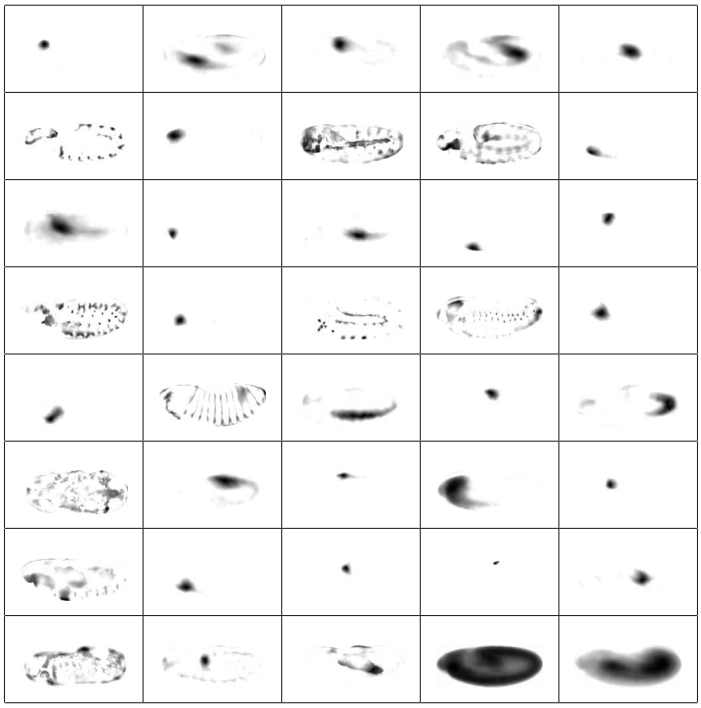

Figure 5: Sample basis images at stage range 11-12 learnt by M-CCD.

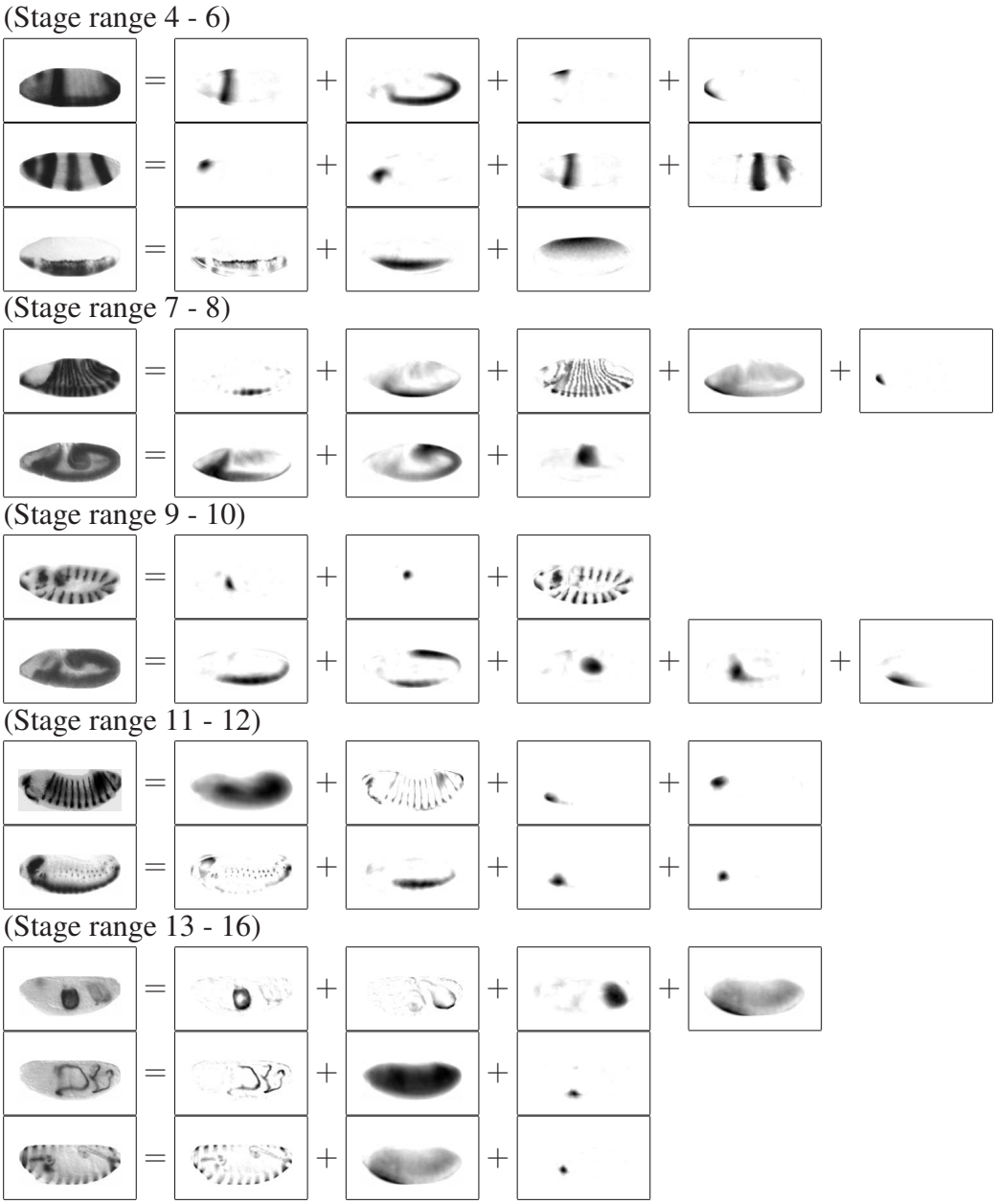

Figure 6: Decomposition of an image as the summation of a set of the learnt basis images.

than several state-of-the-art algorithms. In addition, we propose to construct non-identity core tensors. Our experiments show the effectiveness of the proposed unbalanced core tensor especially when the target tensor is very unbalanced.

We have constructed a collection of basis images for *Drosophila* gene expression pattern images from stages 11-12. We plan to analyze the biological significance of the learnt basis images in the future. In addition, we plan to construct and compare the basis images for all stages to study the dynamics of the embryo development. We will explore the automatic estimation of the parameters involved in T-CCD including the size of the core tensor and λ in the future.

Acknowledgments

This work was supported by NSF IIS-0612069, IIS-0812551, CCF-0811790, NGA HM1582-08-1-0016, NSFC 60905035 and NSFC 61035003.

References

[1] M. W. Berry, M. Browne, A. N. Langville, V. Paul Pauca, and R. J. Plemmons. Algorithms and applications for approximate nonnegative matrix factorization. *Computational Statistics and Data Analysis*, 52(1):155–173, 2007.

[2] R. Bro and C. A. Andersson. The n-way toolbox for matlab. *Chemometrics and Intelligent Laboratory Systems*, 52:1–4, 2000.

[3] J. D. Carroll and J. J. Chang. Analysis of individual differences in multidimensional scaling via an n-way generalization of eckart-young decomposition. *Psychometrika*, 35:283–319, 1970.

[4] A. Cichocki, R.Zdunek, and S. Amari. In *Csiszar's divergences for non-negative matrix factorization Family of new algorithms*, pages 32–39, 2006.

[5] A. Cichocki, R. Zdunek, S. Choi, R. Plemmons, and S. Amari. Nonnegative tensor factorization using alpha and beta divergencies. In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 1393–1396, 2007.

[6] D. FitzGerald, M. Cranitch, and E. Coyle. Non-negative tensor factorisation for sound source separation. In *Irish Signals and Systems Conference*, volume 5, pages 8–12, 2006.

[7] E. Gonzalez and Y. Zhang. Accelerating the lee-seung algorithm for non-negative matrix factorization. *Technical Report, Rice University*, 5(2), 2005.

[8] R. A. Harshman. Foundations of the parafac procedure: models and conditions for an "explanatory" multi-modal factor analysis. *UCLA Working Papers in Phonetics*, 16:1–84, 1970.

[9] P. O. Hoyer. Nonnegative matrix factorization with sparseness constraints. *Journal of Machine Learning Research*, 5:1457–1469, 2004.

[10] H. Kim and H. Park. Sparse non-negative matrix factorizations via alternating non-negativity-constrained least squares for microarray data analysis. *Bioinformatics*, 23(12):1495–1502, 2007.

[11] H. Kim, H. Park, and L. Elden. Non-negative tensor factorization based on alternating large-scale non-negativity-constrained least squares. In *The 7th IEEE International Conferencehe on Bioinformatics and Bioengineering*, pages 1147–1151, 2007.

[12] L. D. Lathauwer, B. D. Moor, and J. Vandewalle. Multilinear singular value decomposition. *SIAM J. MATRIX ANAL. APPL. c*, 21(4):1253–1278, 2000.

[13] D. D. Lee and H. S. Seung. Learning the parts of objects by nonnegative matrix factorization. *Nature*, 401(6755):788–791, 1999.

[14] D. D. Lee and H. S. Seung. Algorithms for non-negative matrix factorization. *Advances in Neural Information Processing Systems*, 13:556–562, 2001.

[15] C.J. Lin. Projected gradient methods for nonnegative matrix factorization. *Neural Computation*, 19(10):2756–2779, 2007.

[16] M. Mørup, L. K. Hansen, and S. M. Arnfred. Erpwavelab a toolbox for multi-channel analysis of time-frequency transformed event related potentials. *Neurosci Methods*, 2:361–368, 2007.

[17] M. Mørup, L. K. Hansen, and S. M. Arnfred. Algorithms for sparse nonnegative tucker decomposition. *Neural Computation*, 20(8):2112–2131, 2008.

[18] M. Mørup, L. K. Hansen, J. Parnas, and S. M. Arnfred. Decomposing the time-frequency representation of eeg using non-negative matrix and multi-way factorization. *Technical University of Denmark Technical Report*, 2006.

[19] T. Murakami and P. M. Kroonenberg. Three-mode models and individual differences in semantic differential data. *Multivariate Behavioral Research*, 38(2):247–283, 2003.

[20] M. R. Parry and I. Essa. Estimating the spatial position of spectral components in audio. *Lecture notes in computer science*, 3889:666–673, 2006.

[21] P. Tomancak, A. Beaton, R. Weiszmann, E. Kwan, S. Q. Shu, S. E. Lewis, S. Richards, M. Ashburner, V. Hartenstein, S. E. Celniker, and G. Rubin. Systematic determination of patterns of gene expression during *Drosophila* embryogenesis. *Genome Biology*, 3(12), 2002.

[22] P. Tseng. Convergence of block coordinate descent method for nondifferentiable minimization. *Journal of Optimization Theory and Applications*, 109:474–494, 2001.

[23] L. R. Tucker. Some mathematical notes on three-mode factor analysis. *Psychometrika*, 31:279–311, 1966.

[24] M.A.O. Vasilescu and D. Terzopoulos. Multilinear independent components analysis. In *The 2005 IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 547–553, 2005.

[25] H. Wang and N. Ahuja. Facial expression decomposition. In *The 9th IEEE International Conference on Computer Vision*, volume 2, pages 958–965, 2003.

[26] M. Welling and M. Weber. Positive tensor factorization. *Pattern Recognition Letters*, 22(12):1255–1261, 2001.

[27] S. Wild, J. Curry, and A. Dougherty. Improving non-negative matrix factorizations through structured initialization. *Pattern Recognition*, 37(11):2217 – 2232, 2004.

[28] R. Zdunek and A. Cichocki. Non negative matrix factorization with quasi newton optimization. In *The 8th International Conference on Artificial Intelligence and Soft Computing*, volume 4029, pages 870–879, 2006.