This article appeared in a journal published by Elsevier. The attached copy is furnished to the author for internal non-commercial research and education use, including for instruction at the authors institution and sharing with colleagues.

Other uses, including reproduction and distribution, or selling or licensing copies, or posting to personal, institutional or third party websites are prohibited.

In most cases authors are permitted to post their version of the article (e.g. in Word or Tex form) to their personal website or institutional repository. Authors requiring further information regarding Elsevier’s archiving and manuscript policies are encouraged to visit:

http://www.elsevier.com/authorsrights
Smooth incomplete matrix factorization and its applications in image/video denoising

Qiulei Dong, Lu Li

Abstract

Low-rank matrix factorization with missing elements has many applications in computer vision. However, the original model without taking any prior information, which is to minimize the total reconstruction error of all the observed matrix elements, sometimes provides a physically meaningless solution in some applications. In this paper, we propose a regularized low-rank factorization model for a matrix with missing elements, called Smooth Incomplete Matrix Factorization (SIMF), and exploit a novel image/video denoising algorithm with the SIMF. Since data in many applications are usually of intrinsic spatial smoothness, the SIMF uses a 2D discretized Laplacian operator as a regularizer to constrain the matrix elements to be locally smoothly distributed. It is formulated as two optimization problems under the $l_1$ norm and the Frobenius norm, and two iterative algorithms are designed for solving them respectively. Then, the SIMF is extended to the tensor case (called Smooth Incomplete Tensor Factorization, SITF) by replacing the 2D Laplacian by a high-dimensional Laplacian. Finally, an image/video denoising algorithm is presented based on the proposed SIMF/SITF. Extensive experimental results show the effectiveness of our algorithm in comparison to other six algorithms.

1. Introduction

Matrix factorization has many applications in computer vision, such as structure-from-motion (SFM) [1], shape from varying illumination [2], feature representation [3,4], data recovery from corruption [5,6], and etc. In ideal case when the data matrix is complete, the Singular Value Decomposition (SVD) is effective to provide a reliable solution to the original matrix factorization problem without any constraints under the Frobenius norm. However, in practice, the data matrix is often not complete, i.e. some elements in the matrix are missing, and the SVD is not applicable in this case. In recent years, this incomplete matrix factorization problem has been investigated extensively [7–19].

The original model without any prior information for the incomplete matrix factorization is formulated as the following minimization problem which minimizes the total reconstruction error corresponding to the observed matrix elements as:

$$\min_{U,V} \| W \odot (A - UV^T) \|$$

(1)

where $A \in \mathbb{R}^{m \times n}$ denotes the input data matrix, $\| \cdot \|$ denotes any matrix norm, $\odot$ denotes the Hadamard (or component-wise) product of two matrices, $W \in \{0,1\}^{m \times n}$ represents a binary indicator matrix where zeros correspond to missing elements and ones correspond to the observed elements, $U \in \mathbb{R}^{m \times r}$ and $V \in \mathbb{R}^{n \times r}$ are the factors, $r$ denotes the reduced dimension.

A number of algorithms have been proposed in the literature to solve this original model based on different matrix norms [7,20–23, 12,10,9,17]. Shum et al. [7] introduced this missing problem first in computer vision and proposed a numerical algorithm under the Frobenius norm. Guerreiro and Aguiar [20] used alternated least squares to calculate a low-rank approximation of the matrix with missing elements in application to the 3D structure recovery from videos. Okatani and Deguchi [12] employed the Wiberg algorithm [21] for solving the original model under the Frobenius norm. Chen [10] proposed two variants of the Levenberg–Marquardt algorithm to calculate the low-rank factorization of a large matrix. Okatani et al. presented a Damped Wiberg (DW) algorithm for low-rank matrix factorization, which incorporated a damping factor into the Wiberg method and can constrain the ambiguity of the matrix factorization. In addition, the problem of the low-rank matrix approximation with missing data was formulated as an $l_1$-norm minimization problem, and was solved by alternative convex programming [9]. Eriksson and Hengel [17] proposed an algorithm for computing the low-rank matrix approximation under the $l_1$-norm based on the differentiability of linear programs, which can be viewed as a generalization of the Wirberg algorithm.

However, due to the non-uniqueness of solutions to (1) as well as the lack of consideration of possible intrinsic constraints on the
matrix entries in real applications, the resultant optimal solution to (1) sometimes may be physically meaningless. Therefore, many modified versions of the original formulation (1) appeared with some additional constraints. For example, Buchanan and Fitzgibbon [8] introduced two penalty terms to the original problem (1), which penalize the two factors $U$ and $V$ respectively. Marques and Costeira [16] presented an incomplete matrix factorization algorithm to solve the structure from motion problem with missing data, which imposed rigid motion constraints on the factor $U$. Zhao and Zhang [11] introduced a constrained model for matrix factorization, which required the entries of $UV^T$ bounded with preset values. In addition, the nuclear norm was introduced into the incomplete matrix factorization problem as a regularized term in [18, 24–27]. Mazumder et al. [18] proposed a SOFT-IMPUTE algorithm for large-scale incomplete matrix factorization by using a nuclear norm as a regularized term. Zheng et al. [27] presented an ALM (Augmented Lagrange Multiplier)-based algorithm for low-rank matrix approximation with missing elements, where an objective function combining a $l_1$-norm reconstructed term and a nuclear-norm regularization term was constructed. Ji et al. [25] proposed an effective patch-based video denoising algorithm with the SIMF/SITF model, and followed by two iterative least-squares algorithm for solving this SITF model under the $l_1$ norm and the Frobenius norm. Finally, detailed algorithmic analysis is provided.

### 2.1. Discretized Laplacian and its penalty function

The Laplacian operator $\Delta$ is a second-order differential operator, which is defined as the divergence of the gradient of a twice-differentiable function. And the Laplacian $\Delta(f(z))$ of a twice-differentiable function $f(z)$ on a $d$-dimensional space $\mathbb{R}^d$ is defined as

$$\Delta(f(z)) = \sum_{i=1}^{d} \frac{\partial^2 f(z)}{\partial z_i^2}$$

Then, the Laplacian penalty function $P$ on a $d$-dimensional region $\Omega$ of size $[n_1, n_2, \ldots, n_d]$ is defined as [28]

$$P(f(z)) = \int_{\Omega} (\Delta(f(z))^2 \, dz$$

It is obvious that the function $P$ reflects the smoothness of $f(z)$ on the region $\Omega$.

To define a discretized approximation to the Laplacian $\Delta$ and its corresponding penalty function, the region $\Omega$ is resized to $[0, 1]^d$ at first for notational convenience. Then a lattice $\Omega_d$ is defined on $\Omega$: Let $\mathcal{Q} = q_1, q_2, \ldots, q_d$ with $q_j = 1/n_j$ ($j = 1, 2, \ldots, d$). $\Omega_d$ is composed of the $d$-dimensional vectors $z = [z_1, \ldots, z_j, \ldots, z_d]$ with $z_j = \frac{\Omega}{n_j}$ ($1 \leq z_j \leq \Omega$), in total, there are $\prod_{j=1}^{d} n_j$ grid points in this lattice.

Let $D_i$ denote an $n_j \times n_j$ matrix which is a discrete approximation to $\partial^2/\partial z_i^2$. Here we use the modified Neumann discretization [29, 28]

$$D_i = \frac{1}{q_i^2} \begin{pmatrix} 1 & -1 & 0 \\ 1 & -2 & 1 \\ 0 & -1 & 0 \\ \vdots & \vdots & \vdots \\ 0 & -1 & 0 \end{pmatrix}$$

Let $g = [g(z_1), g(z_2), \ldots, g(z_d)]$ denote an $n_d$-dimensional vector which is a discretized version of a function $g(z)$, and we have

$$[D_i g]_j = \frac{\partial^2 g(z_j)}{\partial z_i^2}, \quad i = 1, 2, \ldots, n_i$$

Then given $D_i$, the discrete approximation to the $d$-dimensional Laplacian is

$$\Delta_d = \sum_{j=1}^{d} I_j \otimes 1 \otimes \cdots \otimes D_i \otimes \cdots \otimes I_d$$

where $I_j$ is the $n_j \times n_j$ identity matrix ($j = 1, 2, \ldots, d$), and $\otimes$ denotes the Kronecker product.

Therefore, let $f_i$ be a vector which is the discretized version of a function $f(x)$ on the region $\Omega$, then the discretized version of the $d$-dimensional Laplacian penalty function is

$$P(f_a) = \|\Delta_d f_a\|^2$$

### 2.2. Constrained model with the discretized Laplacian penalty

In this paper, we mainly focus on the incomplete matrix factorization in application to image/video denoising. It is noted that in general a pixel and its neighbors in an image are usually spatially correlated, which is an important piece of prior information. Based on this prior information, we introduce the 2D discretized Laplacian penalty function into (1) to constrain the elements of the reconstructed matrix $UV^T$ to be spatially smooth, resulting in a constrained model with the discretized Laplacian penalty, called Smooth Incomplete Matrix Factorization (SIMF) model. Then, two regularized incomplete matrix factorization problems are derived

\[ Q. Dong, L. Li / Neurocomputing 122 (2013) 458–469 \]
from this model under the $l_1$ norm and the Frobenius norm respectively:

$$
\begin{align*}
\min_{U,V} \phi_1 &= \|W \odot (A - UV^T)\|_1 + \lambda_1 P(\nabla U(V^T)) \\
\min_{U,V} \phi_2 &= \|W \odot (A - UV^T)\|_1 + \lambda_2 P(\nabla U(V^T))
\end{align*}
$$

where $\nabla U$ is the vectorization operator which vectorizes a matrix by stacking its rows, $\eta_1(\eta_2)$ is the penalty constant.

In the following two subsections, two iterative algorithms for solving (8a) and (8b) are described respectively in detail.

### 2.3. The iterative $l_1$-norm algorithm

In order to solve the $l_1$-norm optimization problem (8a), we first reformulate it by replacing the reconstructed low-rank matrix $UV^T$ in its first item with an auxiliary matrix $X$ as

$$
\begin{align*}
\min_{X,U,V} \phi_1 &= \|W \odot (A - X)\|_1 + \lambda_1 P(\nabla U(V^T)) \\
\text{s.t.} \quad X &= UV^T
\end{align*}
$$

(9)

Then, the Augmented Lagrange Multiplier (ALM) [30] is introduced for solving (9). The corresponding augmented Lagrangian function of (9) is

$$
\begin{align*}
\ell(U,V,X,H,\eta) &= \|W \odot (A - X)\|_1 + \lambda_1 P(\nabla U(V^T)) \\
&\quad + \langle H, UV^T - X \rangle + \eta \frac{1}{2} \|UV^T - X\|^2
\end{align*}
$$

(10)

where $H$ denotes the Lagrange multiplier, and $\eta$ is a positive penalty parameter, $\langle \cdot, \cdot \rangle$ is the inner product operator. Theoretically, (10) can be solved by the classical iterative ALM-method [31,32] as

$$
\begin{align*}
(U_t, V_t, X_t) &= \arg \min_{U,V} \ell(U,V,X,H_{t-1}, \eta) \\
H_t &= H_{t-1} + \rho\left(\nabla U_t V^T - X_t\right)
\end{align*}
$$

(11)

where $\rho$ is a tuning constant.

Since it is hard to minimize (11a) simultaneously, we adopt the alternating minimization scheme to minimize (11a). At each iteration, each variable is updated while the others are fixed. The detailed performance of the iterative procedure is described as follows (For notational convenience, the subscript ‘t’ is omitted temporally here):

**Optimizing for $U$:** Given $V, X, H, \eta$, the optimization problem (11a) becomes

$$
\begin{align*}
\tilde{U}_t &= \arg \min_{U} \phi_1 + \langle H_t, U V^T - X_t \rangle + \frac{\eta}{2} \|UV^T - X_t\|^2
\end{align*}
$$

(12)

where $\tilde{U}_t$ denotes the pseudo-inverse, whose solution is

$$
\tilde{U}_t = \left(F^T + \frac{2\lambda_1}{\eta} F^T \Delta_2^T G^T F \right)^T \left(x - \frac{1}{\eta} h\right)
$$

(13)

**Optimizing for $V$:** Given $U, X, H, \eta$, the optimization problem (11a) becomes

$$
\begin{align*}
\tilde{V}_t &= \arg \min_{V} \phi_2 + \langle H, U V^T - X_t \rangle + \frac{\eta}{2} \|UV^T - X_t\|^2
\end{align*}
$$

(14)

where $\tilde{V}_t$ denotes the pseudo-inverse, whose solution is

$$
\tilde{V}_t = \left(F^T + \frac{2\lambda_1}{\eta} F^T \Delta_2^T G^T F \right)^T \left(x - \frac{1}{\eta} h\right)
$$

(15)

Obviously, (14) is also a least squares problem, whose solution is

$$
\tilde{V}_t = \left(F^T + \frac{2\lambda_1}{\eta} F^T \Delta_2^T G^T F \right)^T \left(x - \frac{1}{\eta} h\right)
$$

(15)

**Optimizing for $X$:** Given $U, V, H, \eta$, the optimization problem (11a) becomes

$$
\begin{align*}
\tilde{X}_t &= \arg \min_{X} \|W \odot (A - X)\|_1 + \langle H, UV^T - X \rangle + \frac{\eta}{2} \|UV^T - X\|^2
\end{align*}
$$

(16)

And (16) can be minimized based on the soft-thresholding operator as

$$
\begin{align*}
\tilde{X}_t &= \left(S\left(\frac{UV^T - A}{\eta} + \frac{1}{\eta} H_{t-1}\right) + A\right) \odot W \\
\tilde{X}_t^{(1-W)} &= \left(UV^T + \frac{1}{\eta} H_{t-1}\right) \odot (1-W)
\end{align*}
$$

(17)

where $S(x) = \text{sgn}(x)(\max(|x| - \eta, 0))$ is the soft-thresholding (shrinkage) operator.

Summarizing the above description, it is noted that the complete procedure for minimizing (10) is a dual-iterative procedure, where the inner loop is to solve (11a), while the outer loop is to update $H$ and $\eta$. In order to speed up the performance, we only update $(U, V, X)$ once in the inner loop, so that the original dual-loop procedure becomes a single-loop procedure. The complete algorithm is shown in Algorithm 1, denoted as SIMF.

### Algorithm 1. SIMF

**Input:** initial $V_0, R_0, H_0, \eta_0$

**Output:** $U, V$

1. Set $t=1$
2. repeat
3. Compute $\hat{u}_t$ according to (13), $U_t = \hat{u}_t$;
4. Compute $\hat{v}_t$ according to (15), $V_t = \hat{v}_t$;
5. Compute $\hat{x}_t$ according to (17a) (17b);
6. Update $H_t$ according to (11b);
7. $t = t+1$;
8. until convergence;

### 2.4. The iterative $l_0$-norm algorithm

Let $a$ be the row-wise vectorization of $A$, and $w$ the row-wise vectorization of $W$. Then, (8b) can be rewritten as

$$
\begin{align*}
\min_{u} \phi_2 &= \|a - W \odot \hat{u}(Vu)\|_2^2 + 2\lambda_2 \|\Delta_2 \hat{u}(Vu)\|_2^2
\end{align*}
$$

(18)

By omitting the elements in the first term of (18) relating to the missing entries of $a$, (18) can be simplified as

$$
\begin{align*}
\min_{u} \phi_2 &= \|a_{ab} - F_{ab}(Vu)\|_2^2 + 2\lambda_2 \|\Delta_2 \hat{u}(Vu)\|_2^2
\end{align*}
$$

(19)

where $a_{ab}$ is the vector containing all the known elements in $a$, $F_{ab}(V)$ is obtained by omitting the rows relating to the missing entries of $a$ from $F(V)$.

If $V$ is given, the least-squares solution to (19) is

$$
\hat{u}_{ab} = \left(F_{ab}^T F_{ab} + \lambda_2^2 \Delta_2^T \Delta_2 F_{ab}^T F_{ab}\right)^{-1} \left(F_{ab}^T a_{ab}\right)
$$

(20)

Similarly, let $\hat{a}$ be the column-wise vectorization of $A$, and $\hat{w}$ the column-wise vectorization of $W$. Then, (8b) can also be rewritten as the following equivalent form:

$$
\begin{align*}
\min_{v} \phi_2 &= \|\hat{a} - W \odot \hat{G}(Uv)\|_2^2 + 2\lambda_2 \|\Delta_2 \hat{G}(Uv)\|_2^2
\end{align*}
$$

(21)

By omitting the elements in (21) relating to the missing entries in $\hat{a}$, (21) can be simplified as

$$
\begin{align*}
\min_{v} \phi_2 &= \|\hat{a}_{ab} - G_{ab}(Uv)\|_2^2 + 2\lambda_2 \|\Delta_2 \hat{G}(Uv)\|_2^2
\end{align*}
$$

(22)
where $\hat{a}_{ab}$ is the vector containing all the observed elements of $\hat{a}$, $G_{ob}(U)$ is obtained by omitting the rows relating to the missing elements from $G(U)$.

If $U$ is given, the least-squares solution to (22) is

$$\hat{v}_2 = (G_{ob}^T G_{ob} + \lambda_2^2 G_{ab}^T G_{ab})^{-1} G_{ob}^T \hat{a}_{ab}$$

(23)

Based on (20) and (23), given an initial $V_0$ (or $U_0$), $U$ (or $V$) can be updated by fixing $V$ (or $U$) in an alternated way. The procedure is repeated until convergence. The complete algorithm is shown in Algorithm 2, denoted as SIMF2.

**Algorithm 2. SIMF2.**

**Input:** initial $V_0$ for $V$

**Output:** $U$, $V$

1. Set $t = 1$;
2. repeat
3. Construct $F_{ob}$ from $V_{t-1}$, and compute $u_2$ according to (20);
4. $U_{t-1}$, $u_2$;
5. Construct $G_{ob}$ from $U_{t}$, and compute $v_2$ according to (23), $V_{t-1}$, $v_2$;
6. Until convergence;

2.5. Algorithmic analysis

**Convergence analysis:**

SIMF1: Lin et al. [33] have given a convergence proof for an iterative ALM-based algorithm on a convex optimization problem recently. However, it has to be pointed out that since the introduced optimization problem (8a) in this work is non-convex and the SIMF1 algorithm minimizes the augmented Lagrange function (10) with multiple variables in an alternated way, we are not able to give a theoretical proof yet for its convergence currently. In addition, so far as we know, there is also no strict theoretical proof for the convergence of such kind of iterative ALM-based algorithms (as such as [32, 22], etc.) with multiple blocks of variables on non-convex optimization problems as the SIMF1 algorithm. However, extensive experimental results in Section 5 show that the proposed SIMF1 algorithm is prone to achieve convergence. We will investigate the strict theoretical proof for the convergence of such iterative ALM-based algorithms including SIMF1 in the future.

SIMF2: It is obvious that the value of $\phi_2$ in (8b) is always non-negative, hence it is bounded below by zero. And it can be found from (20) and (23) that the $\phi_2$ value is monotonically non-increasing, therefore, the algorithm converges in the limit. The stopping criterion in the SIMF2 algorithm is that the difference of the computed $\phi_2$ in the two consecutive iterations is less than a preset threshold $\xi$.

**Computational complexity:** Here, solving a linear equation with an $L \times R$ coefficient matrix takes $O(LR^2)$ time. Let $T$ denote the number of iterations.

SIMF1: In Line 3 and Line 4 of the Algorithm 1, since the matrices $F$ and $G$ are of block diagonal form, the computational costs for updating $U$ and $V$ are $O(m^2r^3)$ and $O(n^2r^3)$ according to (13) and (15) respectively. In addition, it takes $O(mnr)$ time to calculate $X$ in Line 5 of the Algorithm 1. Because both $O(m^2r^3)$ and $O(n^2r^3)$ are much larger than $O(mnr)$, the total computational complexity of SIMF1 with $T$ iterations can also be considered as $O(m^2r^3 + n^2r^3)T$.

SIMF2: In Line 3 of the Algorithm 2, since the matrices $F$ and $F_{ob}$ are of block diagonal form, the computational cost for updating $U$ is $O(m^2r^3) + O(n^2r^3)$ according to (20). Similarly, it takes $O(n^2r^3)$ for computing $V$ according to (23). Therefore, the total computational complexity with $T$ iterations is $O(m^2r^3 + n^2r^3)T$.

Parameter selection: The penalty parameter $\lambda_1(\xi_2)$ controls the smoothness of the reconstructed low-rank matrix in our proposed constrained model under the $l_1(\xi_2)$ norm, and it influences the corresponding reconstruction error between the recovered observed elements and the true observed elements in an indirect way. When $\lambda_1(\xi_2) \to \infty$, the spatial smoothness of the reconstructed matrix is enhanced, and the reconstruction error becomes larger accordingly. On the contrary, when $\lambda_1(\xi_2) \to 0$, the spatial smoothness of the reconstructed matrix is weakened, and the reconstruction error becomes smaller. In the extreme case when $\lambda_1(\xi_2) = 0$, the smooth term in the SIMF model is discarded and the remaining degenerated form of the SIMF model is the same as the model (1) without taking any prior information.

Therefore, when SIMF is applied in image denoising, in order to obtain good denoising quality, an appropriate penalty parameter $\lambda_1(\xi_2)$ is needed to achieve a trade-off between the smoothness of the reconstructed image and the reconstruction accuracy of the observed pixels. Our experimental results in Section 5 show that our algorithm is not too sensitive to the choice of $\lambda_1(\xi_2)$.

3. Tensor generalization for SIMF

In this section, we extend the matrix case with the 2D Laplacian to the tensor case with the high-dimensional Laplacian.

Similar to SIMF, the incomplete tensor factorization model with high-dimensional discretized Laplacian, called Smooth Incomplete Tensor Factorization (SITF), can also be formulated as the following two optimization problems under the $l_1$ norm and the Frobenius norm:

$$\left\{ \begin{array}{l} \min_{C,\tilde{u}_{i}}, \phi_1 = \|W \otimes (A - C_{x1} U_1 \cdots U_d)\|_1 + \lambda_1 P(A_{d}) \\ \min_{C,\tilde{u}_{i}}, \phi_2 = \|W \otimes (A - C_{x1} U_1 \cdots U_d)\|_2^2 + \lambda_2 P(A_{d}) \end{array} \right. \quad (24)$$

where $A$ is a $d$-mode tensor of dimension $n_1 \times n_2 \times \cdots \times n_d$. $W$ is a $d$-mode indicator tensor of dimension $n_1 \times n_2 \times \cdots \times n_d$, $\otimes$ ($i = 1, 2, \ldots, d$) is the $i$-mode product symbol of a tensor with a matrix, $C$ is a $d$-mode tensor of dimension $r_1 \times r_2 \times \cdots \times r_d$. $U_i$ is an $n_i \times r_i$ matrix, $A_d = \sum_{i=1}^{d} (C_{x1} \otimes U_i \cdots U_d)$, $\lambda_1(\xi_2)$ is the penalty constant.

Here, two iterative least-squares algorithm are proposed for solving (24a) and (24b) respectively by unfolding the tensor into the vector case along each mode of the tensor, and the detailed algorithms are as follows.

3.1. Tensor generalization for SIMF1

In order to solve the $l_1$-norm optimization problem (24a), we first reformulate it by replacing the reconstructed tensor $C_{x1} U_1 \times U_2 \cdots \times U_d$ in its first item with an auxiliary tensor $X$ as

$$\min_{C,\tilde{u}_{i}, X_{\cdot} \cdot \cdot d} \phi_1 = \|W \otimes (A - X)\|_1 + \lambda_1 P(A_{d})$$

s.t. $X = C_{x1} U_1 \cdots U_d$.

(25)

Then the ALM [30] is used for solving (25). The corresponding augmented Lagrangian function of (25) is

$$L(C, (U_{i})_{i=1}^{d}, X, \Lambda) = \|W \otimes (A - X)\|_1 + \lambda_1 P(A_{d})$$

$$+ \langle \Lambda, \sum_{i=1}^{d} (C_{x1} \otimes U_i \cdots U_d) - X \rangle$$

$$+ \frac{\nu}{2} \|\sum_{i=1}^{d} (C_{x1} \otimes U_i \cdots U_d) - X\|^2$$

(26)

Similar to Algorithm 1, we design a single-loop procedure to minimize (26) in an alternated way. At each iteration, each variable...
is updated while the others are fixed

\[
\begin{align*}
\mathcal{C}_t &= \arg \min_{\mathcal{C}_t} L(C_t, (U_{t-1})^d_{i=1, X_t-1, \eta_{t-1}, \eta}) \quad (a) \\
U_{tj} &= \arg \min_{U_{tj}} L(C_t, U_{tj}, (U_{t-1})^d_{j=2, X_t-1, \eta_{t-1}, \eta}) \quad (b) \\
U_{tj} &= \arg \min_{U_{tj}} L(C_t, (U_{tj})^d_{j=1, X_t-1, \eta_{t-1}, \eta}) \quad (c) \\
U_{dt} &= \arg \min_{U_{dt}} L(C_t, (U_{dt})^d_{t=1, X_t, \eta_{t-1}, \eta}) \quad (d) \\
X_t &= \arg \min_{X_t} L(C_t, (U_{t+1})^d_{i=1, X_t, \eta_{t-1}, \eta}) \quad (e) \\
\eta_t &= \mathcal{H}_{t-1} + \rho \mathcal{F}_t(C_t \times U_{t+2} \cdots \times U_{t+2}) \quad (f)
\end{align*}
\]

where \( \rho \) is a tuning constant.

The detailed performance of the above procedure is described as follows (For notational convenience, the subscript ‘t’ is omitted temporarily here):

Optimizing for \( \mathcal{C} \): Given \( (U_{i,t})^d_{i=1} \), \( X_t \), \( \eta_t \), the optimization problem \((27a)\) becomes

\[
\hat{\mathcal{C}} = \arg \min_{\mathcal{C}} L\left(\mathcal{C}, (U_{t-1})^d_{i=1, X_t-1, \eta_{t-1}, \eta}\right)
\]

Optimizing for \( U_t \): Given \( C_t \), \( U_{t-1}, U_{t+1}, \ldots, U_d \), \( X_t, \eta_t, \) the optimization problem \((27b)\) becomes

\[
\hat{U}_t = \arg \min_{U_t} L\left(\mathcal{C}, U_t, (U_{t-1})^d_{j=2, X_t-1, \eta_{t-1}, \eta}\right)
\]

Optimizing for \( X_t \): Given \( C_t \), \( (U_{t+1})^d_{i=1} \), \( \mathcal{H}, \eta_t \), the optimization problem \((27c)\) becomes

\[
\hat{X}_t = \mathcal{H}_{t-1} + \rho \mathcal{F}_t(C_t \times U_{t+2} \cdots \times U_{t+2})
\]

Optimizing for \( \eta_t \): Given \( C_t \), \( (U_{t+1})^d_{i=1} \), \( \mathcal{H}, \) \( \eta_t \), the optimization problem \((27d)\) becomes

\[
\hat{\eta}_t = \mathcal{H}_{t-1} + \rho \mathcal{F}_t(C_t \times U_{t+2} \cdots \times U_{t+2})
\]

3.2. Tensor generalization for SIMF

Let \( a_i \) be the unfolded vector of the tensor \( \mathcal{A} \) along its i-mode, \( w_i \) be the vector corresponding to \( \mathcal{W} \). Then \((24b)\) can be rewritten as

\[
\min_{\mathcal{F}_i} \|a_i - w_i \mathcal{G}(F_i u_i)\|^2_2 + \lambda_2 \|\mathcal{A}_d(F_i u_i)\|^2_2
\]

Algorithm 4. SITF

Input: \( (U_{i,t})^d_{i=1} \), \( C \)
Output: \( (U_{i,t})^d_{i=1} \), \( \mathcal{C} \)
1 Set \( t = 1 \);
2 repeat
3 Construct \( \mathcal{G} \) from \( (U_{i,t})^d_{i=1} \), and compute \( \hat{\mathcal{C}} \) according to \((29)\); \( \hat{\mathcal{C}}_t \leftarrow \hat{\mathcal{C}} \);
4 for \( i = 1 \) to \( d \) do
5 Construct \( F_i \) from \( \hat{\mathcal{C}}_t, U_{t+1}, \ldots, U_{t+1}, U_{t+2}, \ldots, U_{d,t} \), and compute \( \hat{u}_{i,t} \) according to \((31)\);
6 \( \hat{u}_{i,t} \leftarrow \hat{u}_{i,t} \);
7 end
8 Compute \( \hat{X}_t \) according to \((33a)\); compute \( \hat{\eta}_t \) according to \((27f)\); \( t = t + 1 \);
9 until convergence;

By omitting the elements in \((34)\) relating to the missing entries of \( a_i \), \((34)\) can be simplified as

\[
\min_{\mathcal{F}} \|a_{i,ob} - F_i \mathcal{G}(F_i u_i)\|^2_2 + \lambda_2 \|\mathcal{A}_d(F_i u_i)\|^2_2
\]

where \( a_{i,ob} \) is the vector containing all the known elements in \( a_i \), \( F_i \mathcal{G}(F_i u_i) \) is obtained by omitting the rows relating to the missing entries of \( a_i \) from \( F_i \).

Then given \( C, U_1, \ldots, U_{t+1}, U_{t+2}, \ldots, U_d \), the least-squares solution to \((35)\) is

\[
\hat{u}_{i,t} = (F_i \mathcal{G}(F_i u_i) + \lambda_2 F_i \mathcal{G}(F_i u_i) F_i^T)^{-1} F_i \mathcal{G}(F_i u_i) a_i
\]

Similarly, let \( a \) be the unfolded vector of the tensor \( \mathcal{A} \), and \( \mathcal{W} \) the unfolded vector of the indicator tensor \( \mathcal{W} \). Then \((24b)\) can be rewritten as

\[
\min_{\mathcal{F}} \|a - w \mathcal{G}(F_i u_i)\|^2_2 + \lambda_2 \|\mathcal{A}_d(F_i u_i)\|^2_2
\]

By omitting the elements in \((37)\) relating to the missing entries of \( a \), \((37)\) can be simplified as

\[
\min_{\mathcal{F}} \|a_{ob} - \mathcal{G}(F_i u_i)\|^2_2 + \lambda_2 \|\mathcal{A}_d(F_i u_i)\|^2_2
\]

where \( a_{ob} \) is the vector containing all the known elements in \( a \), \( \mathcal{G}(F_i u_i) \) is obtained by omitting the rows relating to the missing entries of \( a \) from \( \mathcal{G} \).
Then given \(|U|d=1\), the least-squares solution to (38) is

\[
\hat{c} = (G^TG \lambda_2 + \lambda_2 G^T \Delta \theta G) \Gamma \Delta \theta^T \Gamma \hat{c}_0
\]  

(39)

Given the initials \(|U|d=1\), each of these variables can be updated by fixing the others alternatively according to (36) and (39). The procedure is repeated until convergence, and the updated by

fi

as SITF2.

4. Image/video denoising with the SIMF/SITF

Here, the detailed image/video denoising algorithm is proposed with the proposed SIMF/SITF:

At first, same as the way to identify the missing pixels in an input corrupted image by VDLRMC [25], two kinds of pixels are identified as missing elements according to the following two criterions: (i) the pixels detected by the adaptive median filter based impulsive noise detector; (ii) the pixels whose values differ from the mean of the corresponding row by the amount larger than a given threshold.

Then, we partition the noisy image (or video) into small-size image (or video) blocks, each of which overlaps its neighbor by \(b\) pixels, and apply the proposed SIMF (SITF) to recover each block with missing pixels independently. The reasons for partitioning the noisy image/video into small-size blocks are: (i) In many cases, a local block contains limited texture information and its image matrix can be considered to be rank deficient, so it can be reconstructed by low-rank matrices without losing too much image quality. (ii) As discussed in Section 2.5, the computational complexity of SIMF increases with the increase of the image size, the image partition can reduce the total computational cost of our denoising algorithm significantly.

Finally, the denoised image is synthesized from these recovered image (video) blocks where the pixel value is determined by using the mean value of the corresponding pixel values in these blocks.

5. Experiments

The proposed SIMF1/SIMF2 algorithms as well as the SITF1/SITF2 algorithms, which are all implemented on a Core 2 Duo 2.53 GHz PC, are tested in application to image denoising and video denoising.

5.1. Image denoising

The performance of the proposed algorithms are tested on the images corrupted by different types of mixed noises from the Berkeley database [35], which contains 300 images of size 321 \(\times\) 481 (For the convenience of implementation, we resize each original image to 320 \(\times\) 480 pixels). The image noise used here is synthesized by mixing three types of noises: independent and identical Gaussian noise with mean 0 and standard variance \(\delta\), Poisson noise (shot noise) with mean 0 and variance \(\eta\), and impulsive noise (either 0 or 255) with probability \(r\). The corrupted images are synthesized by different configurations of these three noise parameters \((\delta, \eta, r)\). In our experiments, the image block is set to be 20 \(\times\) 30 pixels, and the overlap \(b\) is set to be 5 pixels.

5.1.1. Parameter analysis

At first, ten randomly selected images from the Berkeley database are corrupted by the mixed noise level \((\delta = 8, \eta = 5, r = 20\%)\), and used to test the influences of these parameters \(\lambda_1\) in (8a), \(\eta\) in (10), \(\lambda_2\) in (8b), and the reduced dimensionality \(r\) on the visual quality of the denoised images. The SIMF, with \(\lambda_1 = (10^{-21}, 10^{-20}, 10^{-19}, 10^{-18})\) and \(\eta = (5^{-1} \times 10^{-16}, 10^{-16}, 5 \times 10^{-16}, 5^2 \times 10^{-16}, 5^3 \times 10^{-16}, 5^4 \times 10^{-16})\), as well as SIMF2, with \(\lambda_2 = (10^{-11}, 10^{-10}, 10^{-9}, 10^{-8}, 10^{-7})\), are performed for these images with \(r = (5, 10, 15)\). Figs. 1–3 show the denoised results of a test image by SIMF1 with \(r = (5, 10, 15)\) under different configurations of \((\lambda_1, \eta)\). Fig. 4 shows the denoised results of a test image by SIMF2 with \(r = (5, 10, 15)\) under different configurations of \(\lambda_2\). The above results are evaluated by using the quality measure PSNR (Peak Signal to Noise Ratio) defined as

\[
\text{PSNR} = 10 \log_{10} \frac{255^2}{\text{MSE}}
\]  

(40)

Fig. 1. Denoised results by SIMF1 with \(r=5\) under different configurations of \((\lambda_1, \eta)\).
where MSE is the mean squared error between the original image and the reconstructed denoised image. Tables 1–4 list the corresponding PSNR values of the denoised images in Figs. 1–4 respectively. From Tables 1–4 and Figs. 1–4, it can be noted that: (1) For a given $r$, over a large range of $(\lambda_1 \in [10^{-21}, 10^{-18}], \eta \in [10^{-10}, 5 \times 10^{-16}])/(\lambda_2 \in [10^{-10}, 10^{-8}])$, the PSNR value changes slightly and SIMF1/SIMF2 achieve comparably good denoising quality, which means SIMF1/SIMF2 are not quite sensitive to the choice of these parameters $(\lambda_1, \eta)/3$. (2) For a given $(\lambda_1, \eta)/2$, when $r$ ranges from 5 to 15, both the PSNR value and the visual quality of the recovered image by SIMF1/SIMF2 changes slightly in most cases. (3) In SIMF1, for a given $r$, when $\lambda_1$ is set to a relatively small value and $\eta$ is set to a relatively large value (e.g. $\lambda_1 = 10^{-21}, \eta = 5 \times 10^{-16}, r = 15$), the corresponding denoised image is of low visual quality and the corresponding PSNR is small, probably because at this situation the spatial smoothness of the reconstructed image is weakened so seriously that the estimations on the missing pixels by SIMF1 are of lower accuracy. And when $\lambda_1$ is set to a relatively large value and $\eta$ is set to a relatively small value (e.g. $\lambda_1 = 10^{-18}, \eta = 5 \times 10^{-16}, r = 15$), the corresponding denoised image is also of low visual quality and the corresponding PSNR is small, probably because at this situation the spatial smoothness of the reconstructed image becomes blurred. Similarly in SIMF2, for a given $r$, when $\lambda_2$ is set to an excessively small value or an excessively large value, the corresponding denoised image is of low visual quality and the corresponding PSNR is small due to the influence of the spatial smoothness term.

In addition, in order to demonstrate the SIMF1's convergence, we randomly selected 4 image blocks partitioned from the above image sample, and plot the corresponding variation curves of the augmented Lagrangian function $L(U, V, X, H, \eta)$ with $(\lambda_1 = 10^{-20}, \eta = 5 \times 10^{-15})$ and $r = 5, 10, 15$ during 200 iterations in Fig. 5.
Table 1
PSNRs by SIMF1 with $r=5$ under different configurations of $(\lambda_1, \eta)$.

<table>
<thead>
<tr>
<th>$\lambda_1$</th>
<th>$\eta$</th>
<th>$5 \times 10^{-16}$</th>
<th>$10^{-16}$</th>
<th>$5 \times 10^{-16}$</th>
<th>$5^2 \times 10^{-16}$</th>
<th>$5^4 \times 10^{-16}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$10^{-21}$</td>
<td>27.13</td>
<td>27.68</td>
<td>27.36</td>
<td>26.71</td>
<td>24.80</td>
<td></td>
</tr>
<tr>
<td>$10^{-20}$</td>
<td>26.57</td>
<td>26.82</td>
<td>27.86</td>
<td>27.77</td>
<td>27.28</td>
<td></td>
</tr>
<tr>
<td>$10^{-19}$</td>
<td>25.07</td>
<td>26.37</td>
<td>27.41</td>
<td>27.25</td>
<td>27.76</td>
<td></td>
</tr>
<tr>
<td>$10^{-18}$</td>
<td>22.88</td>
<td>24.32</td>
<td>25.97</td>
<td>26.53</td>
<td>26.93</td>
<td></td>
</tr>
</tbody>
</table>

Table 2
PSNRs by SIMF1 with $r=10$ under different configurations of $(\lambda_1, \eta)$.

<table>
<thead>
<tr>
<th>$\lambda_1$</th>
<th>$\eta$</th>
<th>$5 \times 10^{-16}$</th>
<th>$10^{-16}$</th>
<th>$5 \times 10^{-16}$</th>
<th>$5^2 \times 10^{-16}$</th>
<th>$5^4 \times 10^{-16}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$10^{-21}$</td>
<td>28.03</td>
<td>28.56</td>
<td>27.73</td>
<td>27.63</td>
<td>23.17</td>
<td></td>
</tr>
<tr>
<td>$10^{-20}$</td>
<td>27.17</td>
<td>27.85</td>
<td>28.63</td>
<td>28.61</td>
<td>28.17</td>
<td></td>
</tr>
<tr>
<td>$10^{-19}$</td>
<td>25.36</td>
<td>26.56</td>
<td>27.79</td>
<td>28.16</td>
<td>28.57</td>
<td></td>
</tr>
<tr>
<td>$10^{-18}$</td>
<td>23.15</td>
<td>24.62</td>
<td>26.19</td>
<td>27.22</td>
<td>27.97</td>
<td></td>
</tr>
</tbody>
</table>

Table 3
PSNRs by SIMF1 with $r=15$ under different configurations of $(\lambda_1, \eta)$.

<table>
<thead>
<tr>
<th>$\lambda_1$</th>
<th>$\eta$</th>
<th>$5 \times 10^{-16}$</th>
<th>$10^{-16}$</th>
<th>$5 \times 10^{-16}$</th>
<th>$5^2 \times 10^{-16}$</th>
<th>$5^4 \times 10^{-16}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$10^{-21}$</td>
<td>28.18</td>
<td>28.53</td>
<td>28.13</td>
<td>27.55</td>
<td>22.09</td>
<td></td>
</tr>
<tr>
<td>$10^{-20}$</td>
<td>27.01</td>
<td>27.88</td>
<td>28.62</td>
<td>28.39</td>
<td>27.97</td>
<td></td>
</tr>
<tr>
<td>$10^{-19}$</td>
<td>25.48</td>
<td>26.42</td>
<td>27.82</td>
<td>28.20</td>
<td>28.52</td>
<td></td>
</tr>
<tr>
<td>$10^{-18}$</td>
<td>23.50</td>
<td>24.86</td>
<td>26.07</td>
<td>27.17</td>
<td>27.96</td>
<td></td>
</tr>
</tbody>
</table>

5.1.2. Comparative experiments
To further test the effectiveness of our proposed SIMF1 and SIMF2, six existing algorithms, the SALS algorithm [11], the SOFT-IMPUTE algorithm [18], the VDLRMC algorithm [25], the DW algorithm [23], the RegL1-ALM algorithm [27], and the AS algorithm [6] are also tested for a further comparison. It has to be pointed out that (1) Similar to SIMF1 and SIMF2, the SALS algorithm, the SOFT-IMPUTE algorithm, the DW algorithm, and the RegL1-ALM algorithm are designed for reconstructing matrices with missing elements. Therefore, the procedures for image denoising by these algorithms in our experiments are the same as the ones by SIMF1 and SIMF2; (2) The AS algorithm is designed for reconstructing corrupted matrices, so its procedures for image denoising omits the pre-process of identifying missing pixels, and it is applied directly to reconstruct the partitioned image blocks in our experiments; (3) VDLRMC is designed primarily for video denoising, however, in this subsection we test its degenerated version regardless of temporal information for image denoising, and in the next subsection the complete VDLRMC algorithm is tested for video denoising. Fig. 6 shows an image sample corrupted by the mixed noise level ($\delta = 8$, $m = 5$, $r = 20\%$), and the corresponding denoised images by all the above algorithms. Fig. 7 shows the denoised results on a image sample corrupted by the mixed noise level ($\delta = 10$, $m = 10$, $r = 50\%$). In addition, these algorithms are also evaluated under the extreme condition that 80% of the pixels in the test images are corrupted by impulsive noise, and the corresponding denoised images by the above algorithms are shown in Fig. 8.

As seen from Figs. 6–8, it is noted that with the increase of the mixed noise level, the visual quality of the denoised images by each algorithm declines. And the denoised images by SIMF1 and SIMF2 always exhibit better visual quality compared with those by SALS, SOFT-IMPUTE, VDLRMC, DW, RegL1-ALM, and AS. In particular, when the impulsive noise level is large (i.e. the number of the identified missing pixels is large), SIMF1 and SIMF2 can still effectively recover the images, but other referred algorithms in our experiments all fail doing it. The possible reason is that local smoothness is taken into account by the 2D discretized Laplacian, and consequently SIMF1 and SIMF2 become more tolerant to the missing data.
Fig. 5. Variation curves of $L(U, V, X, H, \eta)$ during 200 iterations.

Fig. 6. Image denoising comparison with the mixed noise level ($\delta = 8$, $\sigma = 5$, $\tau = 20\%$): the images and their zoomed regions (red rectangle in (a)) are shown respectively in the first row and the second row of each subfigure: (a) original image; (b) noisy image; (c) denoised image by SALS; (d) denoised image by SOFT-IMPUTE; (e) denoised image by VDLRM; (f) denoised image by DW; (g) denoised image by RegL-ALM; (h) denoised image by AS; (i) denoised image by SIMF$_1$; (j) denoised image by SIMF$_2$. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)
Fig. 7. Image denoising comparison with the mixed noise level \((\delta = 8; \varpi = 10; \tau = 50\%)\): the images and their zoomed regions (red rectangle in (a)) are shown respectively in the first row and the second row of each subfigure. (a) original image; (b) noisy image; (c) denoised image by SALS; (d) denoised image by SOFT-IMPUTE; (e) denoised image by VDLRMC; (f) denoised image by DW; (g) denoised image by \text{RegL}_1\text{-ALM}; (h) denoised image by AS; (i) denoised image by SIMF_1; (j) denoised image by SIMF_2. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

Fig. 8. Image denoising comparison with the mixed noise level \((\delta = 0; \varpi = 0; \tau = 80\%)\): (a) original image; (b) noisy image; (c) denoised image by SALS; (d) denoised image by SOFT-IMPUTE; (e) denoised image by VDLRMC; (f) denoised image by DW; (g) denoised image by \text{RegL}_1\text{-ALM}; (h) denoised image by AS; (i) denoised image by SIMF_1; (j) denoised image by SIMF_2.
5.2. Video denoising

Here, the performances of the proposed denoising algorithm with the SITF1/SITF2 as well as the VDLRMC algorithm [25] are tested on several video samples (downloaded from the website [36]) corrupted by the mixed noise level ($\delta = 8$, $\omega = 5$, $\tau = 80\%$). Here, the reduced dimensions $r_1, r_2, r_3$ are all set to 10.

Fig. 9 shows three frames from a video sample and the corresponding denoised frames by these algorithms. As seen from it, the denoised frames by SITF1 and SITF2 have better visual quality compared with the VDLRMC algorithm. For example, the eyes of the mother in the denoised frame by SITF1 and SITF2 are clearer than those by the VDLRMC algorithm. The possible reason is that the performance of the VDLRMC algorithm is dependent on the patch matching, although a pre-processing to remove impulsive noise from the noisy data is implemented for patch matching in the VDLRMC algorithm, when the video is corrupted seriously, it is not able to obtain sufficiently accurate matching results yet. On the contrary, both SITF1 and SITF2 deal with each video block directly, and make use of the temporal constraint between consecutive frames in addition to the spatial constraint within each frame.

6. Conclusions

In this paper, a regularized incomplete matrix factorization model, called SIMF, is constructed in application to image denoising at first. Since in general each image pixel does not change abruptly with respect to its neighboring pixels, a 2D discretized Laplacian is introduced to this model as a penalty term to constrain the matrix elements to be spatially smooth. Two iterative algorithms, SIMF1 and SIMF2, are proposed to provide an optimal solution to this constrained model under the $l_1$ norm and the Frobenius norm. Then, SIMF is extended to the tensor case with the high-dimensional discretized Laplacian, and two corresponding algorithms are present for solving it under the $l_1$ norm and the Frobenius norm respectively. Finally, an image/video denoising algorithm is proposed with the proposed SIMF/SITF. Experimental results show the effectiveness of our proposed algorithm.

Currently, our proposed SIMF model is solved based on the least squares algorithm. In the future, other optimal algorithms, which could obtain better solutions than the least squares algorithm, will be investigated for this model. In addition, the strict theoretical proof for the convergence of SIMF1 will also be investigated.

Acknowledgment

This work was supported by National Natural Science Foundation of China under Grant nos. (60905020, 61173098).

References


Qiulei Dong received the B.S. degree in automation from Northeastern University, Shenyang, China, in 2003, and the Ph.D. degree from the Institute of Automation, Chinese Academy of Sciences, Beijing, in 2008.

He is currently an associate professor with the Institute of Automation, Chinese Academy of Sciences. His research interest covers pattern classification and 3D computer vision.

Lu Li received her B.S. degree in Automation from Harbin Institute of Technology in 2012. She is now pursuing her M.S. degree in Beijing Jiaotong University. Her major research interests include Compressed Sensing and Image Classification.