AIMS Mathematics, 8(6): 14047–14063.
DOI:10.3934/math.2023718
Received: 31 January 2023
Revised: 26 March 2023
Accepted: 29 March 2023
Published: 14 April 2023

http://www.aimspress.com/journal/Math

Research article

An accelerated alternating directional method with non-monotone technique for matrix recovery

Ruiping Wen[1,*] **and Wenwei Li**[2]

[1] Key Laboratory for Engineering and Computing Science, Shanxi Provincial Department of Education, Taiyuan Normal University, Jinzhong 030619, Shanxi, China

[2] School of Mathematics and Statistics, Taiyuan Normal University, Jinzhong 030619, Shanxi, China

* **Correspondence:** Email: wenrp@163.com; Tel: 03512886656; Fax: 03512886653.

Abstract: Low-rank and sparse structures have been frequently exploited in matrix recovery and robust PCA problems. In this paper, we develop an alternating directional method and its variant equipped with the non-monotone search procedure for solving a non-convex optimization model of low-rank and sparse matrix recovery problems, where the concerned matrix with incomplete data is separable into a low-rank part and a sparse part. The main idea is to use the alternating minimization method for the low-rank matrix part, and to use the non-monotone line search technique for the sparse matrix part to iteratively update, respectively. To some extent, the non-monotone strategy relaxes the single-step descent into a multi-step descent and then greatly improves the performance of the alternating directional method. Theoretically, we prove the global convergence of the two proposed algorithms under some mild conditions. Finally, the comparison of numerical experiments shows that the alternate directional method with non-monotone technique is more effective than the original monotone method and the previous method. The efficiency and effectiveness of the proposed algorithms are demonstrated by solving some instances of random incomplete matrix recovery problems and some problems of the background modeling in video processing.

Keywords: matrix recovery; alternating directional methods; non-monotone search; low-rank sparse structure
Mathematics Subject Classification: 90C06, 90C25

1. Introduction

The problem of recovering low-rank and sparse matrices or tensors from small sets of linear measurements occurs in many areas, such as the foreground in video surveillance [1], hyperspectral compressive sensing [2] and image denoising [3]. Mathematically, the optimization model of a matrix

recovery is described as follows:

$$\begin{aligned} \min_{A,E} \quad & \|A\|_* + \lambda\|E\|_1 \\ \text{s.t.} \quad & D = P_Q(A+E), \end{aligned} \tag{1.1}$$

where $\|A\|_* = \Sigma_{k=1}^{r}\sigma_k(A)$, $\sigma_k(A)$ denotes the kth largest singular value of $A \in \mathbb{R}^{n_1\times n_2}$ of rank r. $\|E\|_1$ denotes the sum of the absolute values of the matrix entries and λ is a positive weighting parameter. $Q \subseteq \mathbb{R}^{n_1\times n_2}$ is a linear subspace, and P_Q denotes the projection onto Q. Since the matrix recovery problem is closely connected to the robust principal component analysis (RPCA) problem, then it can be formulated in the same way as RPCA. Further, many theoretical results and algorithmic methods for recovering a low-rank and sparse matrix or tensor have been obtained; for example, see [1, 2, 4–12], and the references therein.

Wright et al. [6], Li [13], Chen and Xu [14] studied the robust matrix completion problem that we call matrix compressive recovery namely, $P_Q = P_\Omega$, where Ω is a random subset of indices for the known entries. Further, (1.1) is formulated as follows:

$$\begin{aligned} \min_{A,E} \quad & \|A\|_* + \lambda\|E\|_1 \\ \text{s.t.} \quad & D = P_\Omega(A+E). \end{aligned} \tag{1.2}$$

In [1], Candès et al. have proved that both A and E can be recovered by solving (1.2) with high probability. Meanwhile, an algorithm and some applications in the area of video surveillance were discussed in detail. Li [13] also gave some new theorems and models for solving (1.2). Then Chen and Xu [14] discussed (1.2) via the augmented Lagrange multiplier (ALM) method and studied its application in image restoration. Meng et al. proposed the following problem in [3]:

$$\begin{aligned} \min_{A,E} \quad & \|A\|_* + \lambda\|P_\Omega(E)\|_1 \\ \text{s.t.} \quad & A + E = D. \end{aligned} \tag{1.3}$$

They also used the ALM algorithm for solving (1.3), but they did not discuss the convergence. Li et al. [15] proved that the ALM algorithm was convergent for the optimization (1.3). Further, they stated that (1.3) was equivalent to (1.2). Chen et al. [16] provided a new unified performance guarantee that an exact recovery can be obtained when minimizing the nuclear norm plus l_1 norm.

Then the robust formulation has been improved to deal with Gaussian noise [17], leading to the following convex optimization problem (convex robust matrix completion):

$$\begin{aligned} \min_{A,E_1,E_2} \quad & \lambda\|A\|_* + \gamma\|E_1\|_1 + \tfrac{1}{2}\|E_2\|_2^2 \\ \text{s.t.} \quad & D = P_\Omega(A + E_1 + E_2). \end{aligned} \tag{1.4}$$

He et al. [18, 19] developed a robust version of the Grassmannian rank-one update subspace estimation (GROUSE) [20] algorithm named the Grassmannian robust adaptive subspace tracking algorithm (GRASTA), which aims at solving the problem of robust subspace tracking. Their algorithm can be cast to solve problems formulated as

$$\begin{aligned} \min \quad & \|P_\Omega(E)\|_1 \\ \text{s.t.} \quad & P_\Omega(UV + E) = P_\Omega(M) \\ & U \in Gr(m, r) \\ & V \in \mathbb{R}^{r\times n}, \end{aligned} \tag{1.5}$$

where $Gr(m, r)$ is the Grassman manifold. The advantage of their algorithm is that it is designed to tackle the problem of online subspace estimation from incomplete data; hence it can also be cast to solve online low-rank matrix completion where we observe one column of the matrix M at a time.

In 2018, Chiang et al. [21] proposed a general model

$$\begin{aligned} \min_{A,E} \quad & \|A\|_* + \lambda\|E\|_1, \\ \text{s.t.} \quad & x_i^T A y_j + E_{ij} = M_{ij}, \quad \forall (i,j) \in \Omega, \end{aligned} \tag{1.6}$$

which exploits side information to better learn low-rank matrices from missing and corrupted observations, and show that the proposed model can be further applied to several popular scenarios such as matrix completion and RPCA.

On the other hand, for a given low-rank r and sparse level $|S|_0$, some non-convex models and algorithms were proposed.

In 2014, a non-convex algorithm based on alternating projections, namely AltProj, was presented in [22] which solves the following problem:

$$\begin{aligned} \min_{L,S} \quad & \|D - L - S\|_F \\ \text{s.t.} \quad & \text{rank}(L) \leq r^*, \|S\|_0 \leq |\Omega|, \end{aligned} \tag{1.7}$$

where r^* is the rank of the underlying low rank matrix, Ω denotes the support set of the underlying sparse matrix and $|\Omega|$ is the cardinality of Ω. AltProj iteratively updates L by projecting the matrix $D - S$ onto the space of rank-r matrices (denoted by $\mathcal{L}_r$,) which can be done via the singular value decomposition, followed by truncating out small singular values, and then updating S by projecting the matrix $D - L$ onto the space of sparse matrices (denoted by $\mathcal{S}$,) which can be done by the hard thresholding operator. In 2016, Gu et al. [23] factorized L into the product of two matrices, that is $L = UV$, and performd alternating minimization over both matrices. Yi et al. [24] applied a similar factorization and an alternating descent algorithm. Then, a method based on manifold optimization which reduces the dependence on the condition number of the underlying low-rank matrix theoretically was proposed by Zhang and Yang [25]. In 2019, an accelerated AltProj was proposed by Cai et al. [26] and empirical performance evaluations showed the advantage of the accelerated AltProj over other state-of-the-art algorithms for RPCA.

In 2015, Wang et al. [27] proposed a proximal alternating robust subspace minimization method for solving the practical matrix completion problem under the prevalent case where the rank of A and the cardinality of Ω are upper bounded by some known paremeters r and k via the following non-convex, non-smooth optimization model:

$$\begin{aligned} \min_{A,E} \quad & \tfrac{1}{2}\|P_\Omega(D - L - S)\|_F^2 + \tfrac{\epsilon}{2}\|P_{\bar{\Omega}}(D)\|_1^2 \\ \text{s.t.} \quad & \text{rank}(L) \leq r, L \in \mathbb{R}^{m\times n} \\ & \|S\|_0 \leq k, \|S\|_F \leq K_S, S \in \mathbb{R}_\Omega^{m\times n}, \end{aligned} \tag{1.8}$$

where $\mathbb{R}_\Omega^{m\times n}$ denotes the set of $m\times n$ matrices whose supports are subsets of Ω and K_S is a finite constant introduced to facilitate the convergence proof.

In this paper, we develop an alternating directional method and its variant equipped with the non-monotone search procedure for solving low-rank and sparse structure matrix completion problems

from incomplete data. By introducing a parameter α, we consider the following problem:

$$\begin{aligned} \min \quad & \|P_\Omega(D-L-S)\|_F^2, \\ \text{s.t.} \quad & \operatorname{rank}(L) \le r, \\ & \|S\|_0 \le \alpha|\Omega|, \end{aligned} \tag{1.9}$$

where α represents the sparsity ratio of the sparse part. Based on the factorization $L = UV$ with $U \in \mathbb{R}^{m\times r}$ and $V \in \mathbb{R}^{r\times n}$, (1.9) can be represented as

$$\begin{aligned} \min \quad & f(U,V,S) \\ \text{s.t.} \quad & \|S\|_0 \le \alpha|\Omega|, \end{aligned} \tag{1.10}$$

where $f(U,V,S) = \frac{1}{2}\|P_\Omega(D-UV-S)\|_F^2$.

The rest of this paper is organized as follows. In Section 2, we give the proposed algorithms in details. Convergence analysis is discussed under mild condition in Section 3. In Section 4, we compare Algorithm 1 with Algorithm 2 through numerical experiments to illustrate the efficiency of the non-monotone technique, also compare the proposed algorithms with the previous algorithm to show the effectiveness of the new algorithms. Finally, we conclude the paper in Section 5.

2. Proposed algorithms

In this section, we develop two alternating directional methods for solving (1.10), where one of U, V and S is solved in the Gauss-Seidel manner while the other two variables are fixed until convergence. In both algorithms, we apply a single step of the steepest gradient descent method with exact step-size to solve the least square subproblems with respect to variable U or V. If $f(U,V,S)$ is denoted by $f_{V,S}(U)$ when V and S are held constant while $f_{U,S}(V)$ when U and S are held constant. The steepest descents about U and V are

$$\triangledown f_{V,S}(U) = -\big(P_\Omega(D) - P_\Omega(UV+S)\big)V^T$$

and

$$\nabla f_{U,S}(V) = -U^T\big(P_\Omega(D) - P_\Omega(UV+S)\big),$$

respectively. The associated exact step-sizes are denoted by t_U and t_V, and can be computed explicitly by

$$t_U = \frac{\|\nabla f_{V,S}(U)\|_F^2}{\|P_\Omega(\nabla f_{V,S}(U)V)\|_F^2} \text{ and } t_V = \frac{\|\nabla f_{U,S}(V)\|_F^2}{\|P_\Omega(U\nabla f_{U,S}(V))\|_F^2}.$$

Based on the above discussion, the concrete algorithms can be formally described as Algorithm 1 and Algorithm 2, where Algorithm 1 updates S by minimizing $f(U,V,S)$ with the sparsity level constraint while Algorithm 2 updates it by a non-monotone search approach.

Algorithm 1. Input: $P_\Omega(D)$, $U_0 \in \mathbb{R}^{m\times r}$, $V_0 \in \mathbb{R}^{r\times n}$, sparsity parameter α, $S_0 \in \mathbb{R}^{m\times n}$ with $|S_0| \le \alpha|\Omega|$; Repeat

1) $\nabla f_{V_k,S_k}(U_k) = -\big(P_\Omega(D) - P_\Omega(U_kV_k + S_k)\big)V_k^T$
2) $t_{U_k} = \frac{\|\nabla f_{V_k,S_k}(U_k)\|_F^2}{\|P_\Omega\left(\nabla f_{V_k,S_k}(U_k)V_k\right)\|_F^2}$

3) $U_{k+1} = U_k - t_{U_k}\nabla f_{V_k,S_k}(U_k)$
4) $\nabla f_{U_{k+1},S_k}(V_k) = -U_{k+1}^T(P_\Omega(D) - P_\Omega(U_{k+1}V_k + S_k))$
5) $t_{V_k} = \frac{\|\nabla f_{U_{k+1},S_k}(V_k)\|_F^2}{\|P_\Omega\left(U_{k+1}\nabla f_{U_{k+1},S_k}(V_k)\right)\|_F^2}$
6) $V_{k+1} = V_k - t_{V_k}\nabla f_{U_{k+1},S_k}(V_k)$
7) $S_{k+1} = \arg\min\limits_{|S|\leq\alpha|\Omega|} \|P_\Omega(D - U_{k+1}V_{k+1} - S)\|_F^2$.

Until termination criteria is reached.

Algorithm 2. Input: $P_\Omega(D)$, $U_0 \in \mathbb{R}^{m\times r}$, $V_0 \in \mathbb{R}^{r\times n}$, integer $l \geq 0$, sparsity parameter α, $S_0 \in \mathbb{R}^{m\times n}$ with $|S_0| \leq \alpha|\Omega|$;

Repeat

1) $\nabla f_{V_k,S_k}(U_k) = -(P_\Omega(D) - P_\Omega(U_kV_k + S_k))V_k^T$
2) $t_{U_k} = \frac{\|\nabla f_{V_k,S_k}(U_k)\|_F^2}{\|P_\Omega\left(\nabla f_{V_k,S_k}(U_k)V_k\right)\|_F^2}$
3) $U_{k+1} = U_k - t_{U_k}\nabla f_{V_k,S_k}(U_k)$
4) $\nabla f_{U_{k+1},S_k}(V_k) = -U_{k+1}^T(P_\Omega(D) - P_\Omega(U_{k+1}V_k + S_k))$
5) $t_{V_k} = \frac{\|\nabla f_{U_{k+1},S_k}(V_k)\|_F^2}{\|P_\Omega\left(U_{k+1}\nabla f_{U_{k+1},S_k}(V_k)\right)\|_F^2}$
6) $V_{k+1} = V_k - t_{V_k}\nabla f_{U_{k+1},S_k}(V_k)$
7) Set $\bar{S}_k = \arg\min\limits_{|S|\leq\alpha|\Omega|} \|P_\Omega(D - U_{k+1}V_{k+1} - S)\|_F^2$ and update

$$(S_{k+1})_{ij} = \begin{cases} (\bar{S}_k)_{ij} + \tau_k, & \text{if } (\bar{S}_k)_{ij} < 0, \\ (\bar{S}_k)_{ij} - \tau_k, & \text{if } (\bar{S}_k)_{ij} > 0, \\ 0, & \text{otherwise,} \end{cases}$$

where τ_k satisfies the following non-monotone condition:

$$\|P_\Omega(D - U_{k+1}V_{k+1} - S_{k+1})\|_F^2 \leq \max\{\|R_{k+\frac{1}{2}}\|_F^2, \cdots, \|R_{k-l+\frac{1}{2}}\|_F^2\}.$$

Until termination criteria is reached.

Remark 1. In the above algorithms, $|S|$ is the number of non-zero entries of S and the termination criteria is $\frac{\|P_\Omega(D-U_{k+1}V_{k+1}-S_{k+1})\|_F^2}{\|P_\Omega(D)\|_F^2} < \epsilon$.

Remark 2. By preserving the entries of $P_\Omega(D - U_{k+1}V_{k+1})$ with top $\alpha|\Omega|$ large magnitudes and setting the rest to zero, we obtain the sparse matrix S_{k+1} in Algorithm 1 and $\bar{S}_k$ in Algorithm 2.

Remark 3. Algorithm 2 is different from Algorithm 1 in that it employs the non-monotone search procedure for updating S. The non-monotone line search technique relaxes the single-step descent into a multi-step descent, which greatly improves the computational efficiency. The next section also gives the convergence analysis of the proposed algorithms. In Section 4, many work verified numerically that the non-monotone technique outperforms the traditional monotone strategies. Hence, we here utilize the non-monotone search technique to improve the performance of the proposed algorithms.

3. Convergence analysis

In this section, we discuss the global convergence monotone Algorithm 1 and non-monotone Algorithm 2. Theoretically, that is an equivalent verification of the robust PCA can reasonably express the mutual encouragement between the low-rank and structured sparsity.

Before that, we first list some notations and preliminaries for the coming main result. For the ease of exposition, let $\{U_k, V_k, S_k\}$ be the sequence generated by Algorithm 1 or Algorithm 2, and

$$\begin{aligned} R_k &= P_\Omega(D - U_k V_k - S_k), \\ R_{k+\frac{1}{2}} &= P_\Omega(D - U_{k+1} V_{k+1} - S_k), \\ R_{k+1} &= P_\Omega(D - U_{k+1} V_{k+1} - S_{k+1}), \\ \bar{R}_k &= P_\Omega(D - U_{k+1} V_k - S_k). \end{aligned} \tag{3.1}$$

Two propositions are provided first before analysing the convergence, they are the conditions of Lemma 3.6 in [26]:

- $|H| \geq \frac{8}{3}\beta\mu_0\gamma(m+n)\log n$;
- the matrix L is μ_0-relevant.

Lemma 1. The sequence $\{U_k, V_k, S_k\}$ generated by Algorithm 2 satisfies:

$$f(U_{k+1}, V_k, S_k) = f(U_k, V_k, S_k) - \frac{1}{2}\frac{\|\nabla_{V_k,S_k}(U_k)\|_F^4}{\|P_\Omega(\nabla_{V_k,S_k}(U_k))V_k\|_F^2}, \tag{3.2}$$

$$f(U_{k+1}, V_{k+1}, S_k) = f(U_{k+1}, V_k, S_k) - \frac{1}{2}\frac{\|\nabla_{U_{k+1},S_k}(V_k)\|_F^4}{\|P_\Omega(U_{k+1}\nabla_{U_{k+1},S_k}(V_k))V_k\|_F^2}. \tag{3.3}$$

Proof. By simple deduction

$$\begin{aligned} f(U_{k+1}, V_{k+1}, S_k) &= \frac{1}{2}\|P_\Omega(D - S_k - U_{k+1}V_k)\|_F^2 \\ &= \frac{1}{2}\|P_\Omega(D - S_k - (U_k + t_{U_k}\nabla V_k, S_k(U_k))V_k\|_F^2 \\ &= \frac{1}{2}\|P_\Omega(D - S_k - U_k V_k) - t_{U_k}P_\Omega(\nabla V_k, S_k(U_k))V_k\|_F^2 \\ &= \frac{1}{2}\|R_k\|_F^2 - t_{U_k}\langle P_\Omega(\nabla V_k, S_k(U_k))V_k, R_k\rangle + \frac{1}{2}\|P_\Omega(\nabla V_k, S_k(U_k))V_k\|_F^2 \\ &= f(U_k, V_k, S_k) - \frac{1}{2}\frac{\|\nabla_{V_k,S_k}(U_k)\|_F^4}{\|P_\Omega(\nabla_{V_k,S_k}(U_k))V_k\|_F^2}. \end{aligned}$$

Similarly, we have

$$f(U_{k+1}, V_{k+1}, S_k) = f(U_{k+1}, V_k, S_k) - \frac{1}{2}\frac{\|\nabla_{U_{k+1},S_k}(V_k)\|_F^4}{\|P_\Omega(U_{k+1}\nabla_{U_{k+1},S_k}(V_k))V_k\|_F^2}.$$

Theorem 1. Suppose that there exist $(U_i^T U_i)^{-1}$ and $(V_i^T V_i)^{-1}$ and they are bounded. Then, the sequence $\{U_k, V_k, S_k\}$ generated by Algorithm 2 satisfies:

$$\lim_{k\to\infty} \nabla_{V_k,S_k}(U_k) = 0,$$

and

$$\lim_{k\to\infty} \nabla_{U_{k+1},S_k}(V_k) = 0.$$

Proof. According to S_{k+1} given by Algorithm 2, we obtain

$$f(U_{k+1}, V_{k+1}, S_{k+1}) \leq f(U_k, V_k, S_k) + |\Omega|\tau_k^2,$$

together with Lemma 1, which implies that

$$\begin{aligned} f(U_{k+1}, V_{k+1}, S_{k+1}) \quad \leq \quad & f(U_0, V_0, S_0) - \frac{1}{2}\sum_{i=1}^{k} \frac{\|\nabla_{V_i,S_i}(U_i)\|_F^4}{\|P_\Omega(\nabla_{V_i,S_i}(U_i))V_i\|_F^2} \\ & -\frac{1}{2}\sum_{i=1}^{k} \frac{\|\nabla_{U_{i+1},S_i}(V_i)\|_F^4}{\|P_\Omega(U_{i+1}\nabla_{U_{i+1},S_i}(V_i))\|_F^2} + |\Omega|\sum_{i=1}^{k}\tau_i^2. \end{aligned}$$

Since

$$\begin{aligned} \|P_\Omega(\nabla_{V_k,S_k}(U_k))V_k\|_F^2 \quad &\leq \quad \|\nabla_{V_k,S_k}(U_k)V_k\|_F^2 \\ &= \quad \|R_k V_k^T V_k\|_F^2 \\ &= \quad \mathrm{tr}(R_k^T R_k (V_k^T V_k)^2), \end{aligned}$$

and

$$\|\nabla_{U_{i+1},S_i}(V_i)\|_F^2 = \|R_k V_k^T\|_F^2 = \mathrm{tr}(R_k^T R_k (V_k^T V_k)),$$

then

$$\begin{aligned} \frac{1}{2}\sum_{i=1}^{k} \frac{\|\nabla_{V_i,S_i}(U_i)\|_F^4}{\|P_\Omega(\nabla_{V_i,S_i}(U_i))V_i\|_F^2} \quad &\geq \quad \frac{\mathrm{tr}^2(R_i^T R_i (V_i^T V_i)^2)}{\mathrm{tr}(R_i^T R_i (V_i^T V_i))} \\ &\geq \quad \frac{1}{2}\sum_{i=1}^{k} \sigma_{ir}^2 \|\nabla_{V_i,S_i}(U_i)\|_F^2, \end{aligned}$$

where σ_{ir} is the ith largest singular value of the matrix V_i.
Similarly, we have

$$\frac{1}{2}\sum_{i=1}^{k} \frac{\|\nabla_{U_{i+1},S_i}(V_i)\|_F^4}{\|P_\Omega(U_{i+1}\nabla_{U_{i+1},S_i}(V_i))\|_F^2} \quad \geq \quad \frac{1}{2}\sum_{i=1}^{k} \tilde{\sigma}_{ir}^2 \|\nabla_{U_{i+1},S_i}(V_i)\|_F^2,$$

where $\tilde{\sigma}_{ir}$ is the ith largest singular value of the matrix U_i.

Consequently, $\sum\limits_{i=1}^{\infty}(\sigma_{ir}^2\|\nabla_{V_i,S_i}(U_i)\|_F^2 + \tilde{\sigma}_{ir}^2\|\nabla_{U_{i+1},S_i}(V_i)\|_F^2 + |\Omega|\tau_i^2)$ is convergent. Thus,

$$\lim_{k\to\infty} \sigma_{ir}^2 \|\nabla_{V_i,S_i}(U_i)\|_F = 0,$$

and

$$\lim_{k\to\infty} \tilde{\sigma}_{ir}^2 \|\nabla_{U_{i+1},S_i}(V_i)\|_F = 0.$$

The theorem is true from the assumptions.

In addition, from the Lemma 3.6 of [28], there exists a scalar c $(0 < c < 1)$, such that

$$\|P_\Omega(\bar{V}_{k,S_k}(V_k)V_k\|_\infty \geq (1-c)\|\bar{V}_{k,S_k}(V_k)V_k\|_\infty.$$

Hence,

$$t_{U_k} \leq \frac{\sigma_{k_1}^2}{1-c_1},$$

where $0 < c_1 < 1, \sigma_{k_1}$ is the top singular value of V_k. Therefore,

$$\lim_{k\to\infty}(U_{k+1} - U_k) = 0.$$

Similarly,

$$\lim_{k\to\infty}(V_{k+1} - V_k) = 0.$$

The proof is completed.

Theorem 2. Assume that the sequence $\{U_k, V_k\}$ is bounded and there exist $(U_k^T U_k)^{-1}$ and $(V_k^T V_k)^{-1}$. Then

$$\lim_{k\to\infty} L_k = L_*,$$

and

$$\lim_{k\to\infty} S_k = S_*.$$

Proof. The $\{L_k\}$ is bounded from the sequence $\{U_k, V_k\}$ is bounded. Then there exists a sub-sequence $\{L_{k_i}\}$ is closed to L_*^α. It is noted that

$$\lim_{k\to\infty} \|U_{k+1} - U_k\| = 0 \quad \text{and} \quad \lim_{k\to\infty} \|V_{k+1} - V_k\| = 0.$$

Then

$$\begin{aligned}
\|L_{k+1} - L_k\|_F &= \|U_{k+l}V_{k+l} - U_kV_k\|_F \\
&= \|(U_k + t_{U_k}\nabla_{V_k,S_k}(U_k))(V_k + t_{V_k}\nabla_{V_{k+1},S_k}(V_k)) - U_kV_k\|_F \\
&= \|(U_{k+1} - U_k)V_k + U_k(V_{k+1} - V_k) + (U_{k+1} - U_k)(V_{k+1} - V_k)\|_F \\
&\leq \|U_{k+1} - U_k\|_F\|V_k\|_F + \|U_k\|_F\|V_{k+1} - V_k\|_F + \|U_{k+1} - U_k\|_F\|V_{k+1} - V_k\|_F.
\end{aligned}$$

Thus,

$$\lim_{k\to\infty} \|L_{k+1} - L_k\|_F = 0.$$

That is to say,

$$\lim_{k\to\infty} L_k = L_*.$$

Let $D - L_* = S_*$. Then

$$\begin{aligned}
\|(S_k)_{ij} - (S_*)_{ij}\|_\infty &= \|(D - L_k)_{ij} - (D - L_*)_{ij} + \tau_k\|_\infty \\
&\leq \|L_k - L_*\|_\infty + \tau_k, \qquad i, j \in \alpha\Omega,
\end{aligned}$$

which shows that

$$\lim_{k\to\infty} S_k = S_*.$$

The proof is completed.

4. Numerical experiments

In this section, we apply the proposed algorithms to solve two problems: some matrix recovery tasks with incomplete samples and some background modeling in video processing. We compare our algorithms and the AM algorithm in [23] in the senses of the iteration step (denoted as IT) and the total CPU time (denoted as CPU) in second. Moreover, R.error and Error represent the relative deviation of the deserved matrices (or images) from the given matrices (or images), which are computed by the following formulas:

$$\text{R.error} = \frac{\|P_\Omega(X_k) - P_\Omega(D)\|_F}{\|P_\Omega(D)\|_F},$$

and

$$\text{Error} = \frac{\|A + E - D\|_F}{\|D\|_F}.$$

In our implementations, all the codes were written by Matlab R2019b and run on a PC with Intel Xeon E5 processor 2.5GHz and 20GB memory.

4.1. Numerical experiments for recovering a random matrix

In the experiments, the dimension n of a square matrix $X \in \mathbb{R}^{n\times n}$ is denoted as Size(X) in the list. For each concerned X, the sampling density is denoted as Den(X) may be 60%, 70%, 75% and 80%, the rank of the low-rank component is denoted by Rank(L) may be 50, 60, 90 and 100, and the sparsity ratio of sparse part is denoted by Spa(S) may be 5% and 10%. In total, we obtain some instances for each dimension. In each instance, the test matrix is randomly generated. The numerical results are provided in Tables 1–3. Here, the iteration is terminated once the current iterations obey R.error $< 10^{-4}$ or the criterion is not satisfied after 10000 iteration steps. The symbol "-" indicates that the iteration is failing.

For Algorithm 2, we set $l = 2$ and the initial thresholding value in each iteration to be the $p + 1$-th largest number of the absolute values of all elements of S_k, where p is the required number of the non-zero entries of S, i.e., $p = \alpha|\Omega|$. By the way, Alg 1 and Alg 2 represent the abbreviations of Algorithm 1 and Algorithm 2, respectively.

The results recorded in Tables 1–3 show that the proposed algorithms are feasible and efficient for solving some matrix recovery tasks with incomplete samples. It is worth mentioning that our proposed algorithms are more effective than AM algorithm (see [23]) for large-size problems since both of the algorithms can obtain a solution within reasonable time for the problems with size 10000×10000.

Moreover, the results recorded in Figures 1–2 illustrate the superiority of non-monotone decrease in the iteration procedure of Algorithm 2. We observe Algorithm 2 outperforms Algorithm 1 since the CPU times of Algorithm 2 are much shorter than those of Algorithm 1 and the relative error of Algorithm 2 is smaller than those of Algorithm 1. The acceleration ratio of Algorithm 2 with 60% samples is 13.3 (the left in Figure 1) and 26.8 (the right in Figure 1) respectively, that with 80% samples is 8.0 (the left in Figure 1) and 18.1 (the right in Figure 2) respectively.

Table 1. Computational results for small-size problems.

Size(X)	Rank(L)	Den(X)(%)	Spar(S)(%)	Algorithm	R.error	CPU(S)	IT
				AM	$9.23e-05$	27.11	142
1000	50	70	5	Alg 1	$9.96e-05$	104.79	308
				Alg 2	$8.74e-05$	4.06	14
				AM	$9.90e-05$	29.01	143
1000	50	80	10	Alg 1	$9.99e-05$	119.50	319
				Alg 2	$9.74e-05$	4.23	13
				AM	$8.03e-05$	39.65	75
1000	100	60	5	Alg 1	$9.94e-05$	23.01	68
				Alg 2	$9.98e-05$	7.75	24
				AM	$8.61e-05$	56.03	80
1000	100	60	10	Alg 1	$9.49e-05$	22.21	67
				Alg 2	$9.09e-05$	8.21	26
				AM	$9.66e-05$	61.97	79
1000	100	70	5	Alg 1	$9.99e-05$	27.19	76
				Alg 2	$9.92e-05$	7.17	21
				AM	$8.89e-05$	71.03	80
1000	100	70	10	Alg 1	$9.92e-05$	39.27	105
				Alg 2	$9.72e-05$	7.97	23
				AM	$7.81e-05$	73.11	91
1000	100	75	10	Alg 1	$9.02e-05$	44.25	111
				Alg 2	$9.42e-05$	6.99	27
				AM	$9.21e-05$	90.79	144
2000	50	70	5	Alg 1	$9.99e-05$	226.39	154
				Alg 2	$8.70e-05$	12.50	10
				AM	$7.51e-05$	98.70	142
2000	50	70	10	Alg 1	$9.99e-05$	514.65	339
				Alg 2	$6.64e-05$	13.97	11
				AM	$9.88e-05$	109.23	70
2000	50	80	5	Alg 1	$9.78e-05$	164.68	106
				Alg 2	$9.91e-05$	11.50	9
				AM	$6.72e-05$	86.11	90
2000	50	80	10	Alg 1	$9.99e-05$	323.54	205
				Alg 2	$7.85e-05$	13.88	10
				AM	$9.05e-05$	101.23	98
2000	100	60	5	Alg 1	$9.99e-05$	278.48	190
				Alg 2	$8.90e-05$	17.65	14
				AM	$7.03e-05$	98.27	108
2000	100	75	10	Alg 1	$9.01e-05$	511.54	329
				Alg 2	$8.21e-05$	16.89	12

Table 2. Computational results for middle-size problems.

Size(X)	Rank(L)	Den(X)(%)	Spar(S)(%)	Algorithm	R.error	CPU(S)	IT
				AM	$9.61e-05$	98.81	83
3000	50	70	5	Alg 1	$9.94e-05$	429.42	141
				Alg 2	$9.72e-05$	22.58	9
				AM	$7.08e-05$	122.00	98
3000	50	70	10	Alg 1	$9.97e-05$	891.26	288
				Alg 2	$6.04e-05$	25.56	10
				AM	$7.08e-05$	101.03	111
3000	60	75	5	Alg 1	$9.33e-05$	668.20	269
				Alg 2	$9.41e-05$	24.44	8
				AM	$7.23e-05$	100.31	40
4000	50	70	10	Alg 1	$9.97e-05$	1158.50	235
				Alg 2	$5.88e-05$	40.06	10
				AM	$6.28e-05$	85.02	35
4000	50	75	5	Alg 1	$9.15e-05$	356.20	70
				Alg 2	$9.22e-05$	28.44	7
				AM	$8.28e-05$	88.22	36
4000	50	80	5	Alg 1	$9.95e-05$	374.40	73
				Alg 2	$9.39e-05$	30.74	8
				AM	$7.23e-05$	85.33	38
4000	50	80	10	Alg 1	$9.99e-05$	841.22	161
				Alg 2	$6.54e-05$	38.48	9
				AM	$9.23e-05$	90.21	41
4000	100	60	5	Alg 1	$9.99e-05$	874.89	183
				Alg 2	$8.48e-05$	739.76	10
				AM	$6.44e-05$	104.27	39
5000	50	70	5	Alg 1	$9.98e-05$	735.44	98
				Alg 2	$5.60e-05$	53.51	9
				AM	$7.05e-05$	99.45	38
5000	50	70	10	Alg 1	$9.97e-05$	1585.46	206
				Alg 2	$9.08e-05$	56.45	9
				AM	$5.92e-05$	196.04	47
5000	50	80	5	Alg 1	$9.98\ e-05$	468.41	59
				Alg 2	$7.20e-05$	48.16	8
				AM	$7.73e-05$	395.47	69
5000	100	75	10	Alg 1	$9.07e-05$	1011.22	124
				Alg 2	$5.88e-05$	57.94	9
				AM	$6.72e-05$	256.11	98
5000	50	80	10	Alg 1	$9.99e-05$	322.04	201
				Alg 2	$8.05e-05$	13.11	9

Table 3. Computational results for large-size problems.

Size(X)	Rank(L)	Den(X)(%)	Spar(S)(%)	Algorithm	R.error	CPU(S)	IT
				AM	$7.02e-05$	950.03	70
10000	50	60	5	Alg 1	$9.97e-05$	2527.91	98
				Alg 2	$5.54e-05$	191.48	9
				AM	$9.01e-05$	1417.34	151
10000	50	60	10	Alg 1	$9.99e-05$	5125.45	194
				Alg 2	$8.67e-05$	191.29	9
				AM	$8.81e-05$	1022.03	41
10000	50	70	5	Alg 1	$9.98e-05$	1683.68	59
				Alg 2	$7.83e-05$	173.10	8
				AM	$7.70e-05$	445.11	50
10000	50	75	10	Alg 1	$9.95e-05$	3863.30	135
				Alg 2	$6.29e-05$	208.24	9
				AM	$6.27e-05$	566.03	48
10000	50	80	5	Alg 1	$9.93e-05$	1249.34	41
				Alg 2	$5.46e-05$	189.02	8
				AM	$9.08e-05$	575.14	72
10000	50	80	10	Alg 1	$9.96e-05$	2528.33	80
				Alg 2	$8.50e-05$	190.36	8
				AM	-	-	-
10000	100	60	5	Alg 1	$9.99e-05$	2907.35	109
				Alg 2	$4.93e-05$	197.94	9
				AM	-	-	-
10000	100	60	10	Alg 1	$9.99e-05$	6455.45	235
				Alg 2	$8.36e-05$	199.31	9
				AM	$6.72e-05$	551.20	61
10000	100	70	5	Alg 1	$9.90e-05$	1820.84	63
				Alg 2	$6.97e-05$	178.44	8
				AM	$7.27e-05$	748.86	65
10000	100	70	10	Alg 1	$9.99e-05$	4139.06	140
				Alg 2	$5.42e-05$	210.46	9
				AM	$8.21e-05$	605.16	76
10000	90	75	10	Alg 1	$9.02e-05$	4009.77	128
				Alg 2	$6.72e-05$	200.01	8
				AM	$8.02e-05$	761.11	62
10000	100	80	5	Alg 1	$9.88e-05$	1461.60	45
				Alg 2	$4.56e-05$	197.07	8
				AM	-	-	-
10000	100	80	10	Alg 1	$9.93e-05$	2892.42	90
				Alg 2	$7.55\ e-05$	197.00	8

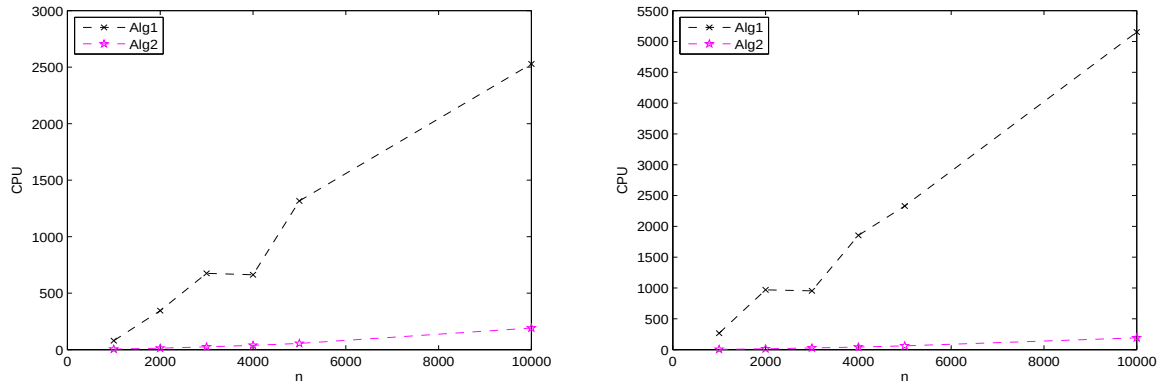

Figure 1. Compare the difference in CPU time between different parameters. **Left**: R=50, Den(X)=60, Spa=0.05. **Right**: R=50, Den(X)=60, Spa=0.1.

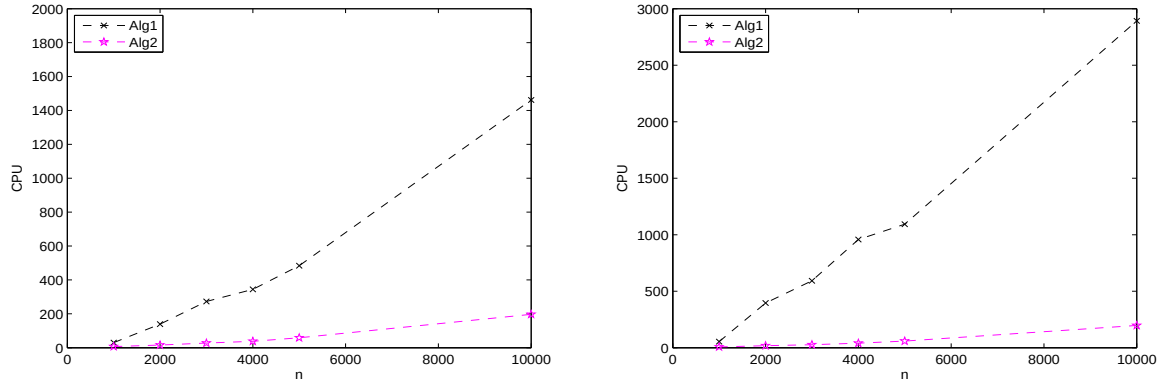

Figure 2. Compare the difference in CPU time between different parameters. **Left**: R=100, Den(X)=80, Spa=0.05. **Right**: R=100, Den(X)=80, Spa=0.1.

4.2. Numerical experiments for recovering the real data

In order to verify the performance of the new algorithms in solving practical problems, we compare Algorithm 2 and the AM algorithm for solving the background modeling in video processing. The problem of the background modeling is to separate the foreground and the background in the video. We use the AM algorithm and our Algorithm 2 to separate the foreground and the background for four real videos from [29]. All videos meet the requirement of the low-rank sparse structure since the background of all frames are relevant and the moving objects are sparse and independent. In our tests, each data matrix consists of the first 200 frames of each video. For example, the first video consists of the first 200 frames with a resolution of 144×176, the size of the matrix should be 25344×200 by converting each frame into a vector. Here, the iteration is terminated once the current iterations obey Error $< 10^{-7}$ or the criterion is not satisfied after 3000 iteration steps.

The results recorded in Figures 3–6 are the separation of one frame of each video sequence under $D = A + E$ model. D is the original image, A denotes its background (the low-rank part) and E its foreground (the sparse part).

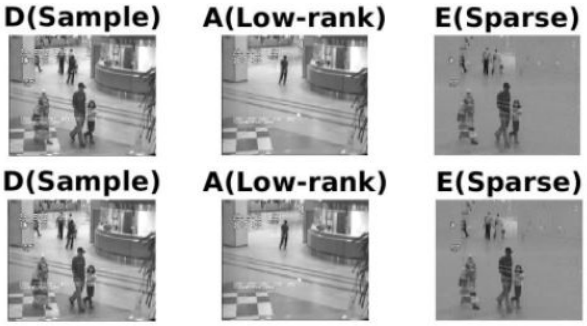

Figure 3. Separation results for the hall: the upper is the result of AM and the lower is one of Alg 2.

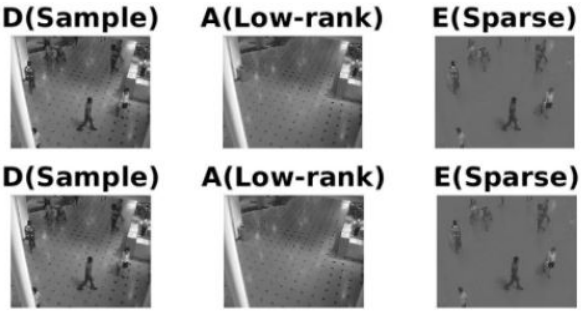

Figure 4. Separation results for the shopping-mall: the upper is the result of AM and the lower is one of Alg 2.

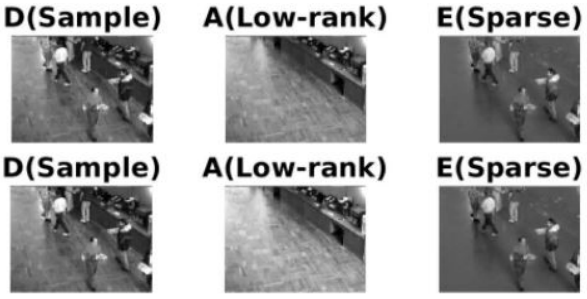

Figure 5. Separation results for the bootstrap: the upper is the result of AM and the lower is one of Alg 2.

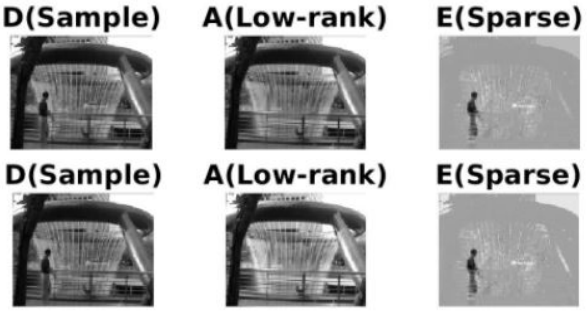

Figure 6. Separation results for the fountain: the upper is the result of AM and the lower is one of Alg 2.

Table 4 lists the Error and the running time CPU(S) of the experiments. From Table 4, we can see that the running time of the AM is about 2.8–5.4 times that of Algorithm 2, and the accuracy of Algorithm 2 is also always higher than that of AM algorithm in Figures 3–6 and Table 4, so the advantage of Algorithm 2 in the recovery of high-dimensional array is obvious.

Table 4. Experiment results of background-foreground separation.

Image	**Resolution**	**AM**		**Alg 2**	
		Error	CPU(S)	Error	CPU(S)
Fig 3	144×176	8.83e-08	334.35	6.42e-08	108.85
Fig 4	256×320	8.52e-08	1038.13	8.07e-08	191.09
Fig 5	120×160	9.17e-08	275.68	7.31e-08	98.57
Fig 6	128×160	8.58e-08	304.14	6.45e-08	99.30

5. Conclusions

In this paper, we focus on the problem of recovering a matrix that is the sum of a low-rank matrix and a sparse matrix from a subset of its entries. This model can characterize many problems arising from

the areas of signal and image processing, statistical inference, and machine learning. We propose an alternating directional method for solving the low-rank matrix sparse structure model. The key idea of the method is that each block variables are solved in the Gauss-Seidel manner while the others are fixed until convergence. We further develop a version of our algorithm by introducing non-monotone search technique to improve the performance of the new algorithm. Both versions are theoretically proved to be globally convergent under some requirements. Based on computational records, we observe that both algorithms are computationally inexpensive to find satisfactory results and the non-monotone strategy performs much better than the monotone one from these instances.

Acknowledgments

The authors are very much indebted to the anonymous referees for their helpful comments and suggestions which greatly improved the original manuscript of this paper. The authors are so thankful for the support from the special fund for science and technology innovation teams of Shanxi province (202204051002018) and the scientific research project of Returned Overseas Chinese in Shanxi Province, China (2022-169).

Conflict of interest

The authors declare that they have no conflict of interests.

References

1. E. J. Candès, X. Li, Y. Ma, J. Wright, Robust principal component analysis, *J. ACM*, **58** (2011), 1–37. https://doi.org/10.1145/1970392.1970395
2. A. E. Water, A. C. Sankaranarayanan, R. G. Baraniuk, SpaRCS: recovering low-rank and sparse matrices from compressive measurements, *24th International Conference on Neural Information Processing Systems*, 2011, 1089–1097.
3. F. Meng, X. M. Yang, C. H. Zhou, The augmented Lagrange multipliers method for matrix completion from corrupted samplings with application to mixed Gaussian-impulse noise removal, *PLoS ONE*, **9** (2014), e108125. https://doi.org/10.1371/journal.pone.0108125
4. J. Xue, Y. Zhao, S. Huang, W. Liao, S. G. Kong, Multilayer sparsity-based tensor decomposition for low-rank tensor completion, *IEEE T. Neur. Net. Lear.*, **33** (2022), 6916–6930. https://doi.org/10.1109/TNNLS.2021.3083931
5. J. Xue, Y. Zhao, Y. Bu, J. C. W. Chan, S. G. Kong, When Laplacian scale mixture meets three layer transform: a parametric tensor sparsity for tensor completion, *IEEE T. Cybernetics*, **52** (2022), 13887–13901. https://doi.org/10.1109/TCYB.2021.3140148
6. J. Wright, A. Ganesh, K. Min, Y. Ma, Compressive principal component pursuit, *2012 IEEE International Symposium on Information Theory Proceedings*, Cambridge, MA, USA, 2012, 1276 –1280 https://doi.org/10.1109/ISIT.2012.6283062
7. J. Wright, A. Ganesh, S. Rao, Y. Peng, Y. Ma, Robust principal component analysis: exact recovery of corrupted low-rank matrices by convex optimization, arXiv: 0905.0233.

8. H. Xu, C. Caramanis, S. Sanghavi, Robust PCA via outlier pursuit, *IEEE T. Inform. Theory*, **58** (2012), 3047–3064. https://doi.org/10.1109/TIT.2011.2173156
9. Z. Aidene, J. Huang, J. Wang, The effect of perturbation and noise folding on the recovery performance of low-rank matrix via the nuclear norm minimization, *Intelligent Systems with Applications*, **13** (2022), 200058. https://doi.org/10.1016/J.ISWA.2021.200058
10. Z. Zhou, X. Li, J. Wright, E. J. Candés, Y. Ma, Stable principal component pursuit, *2010 IEEE International Symposium on Information Theory*, Austin, TX, USA, 2010, 1518–1522. http://doi.org/10.1109/ISIT.2010.5513535
11. S. Zhang, D. Ding, C. Zhao, L. Zhao, Three-dimensional sar imaging with sparse linear array using tensor completion in embedded space, *EURASIP J. Adv. Sigal Process.*, **2022** (2022), 1–18. http://doi.org/10.1186/s13634-022-00896-x
12. J. Yi, W. Xu, Necessary and sufficient null space condition for nuclear norm minimization in low-rank matrix recovery, *IEEE T. Inform. Theory*, **66** (2020), 6597–6604. http://doi.org/10.1109/TIT.2020.2990948
13. X. Li, Compressed sensing and matrix completion with constant proportions of corruptions, *Constr. Approx.*, **37** (2013), 73–99. http://doi.org/10.1007/s00365-012-9176-9
14. Y. D. Chen, H. Xu, C. Caramanis, S. Sanghavi, Robust matrix completion with corrupted columns, *IEEE T. Inform. Theory*, **62** (2016), 503–526. http://doi.org/10.1109/TIT.2015.2499247
15. C. Li, C. L. Wang, J. Wang, Convergence analysis of the augmented Lagrange multiplier algorithm for a class of matrix compressive recovery, *Appl. Math. Lett.*, **59** (2016), 12–17. http://doi.org/10.1016/j.aml.2016.02.022
16. Y. D. Chen, A. Jalali, S. Sanghavi, C. Caramanis, Low-rank matrix recovery from errors and erasures, *IEEE T. Inform. Theory*, **59** (2013), 4324–4337. http://doi.org/10.1109/TIT.2013.2249572
17. T. Hastie, R. Mazumder, R. Tibshirani, *Matrix completion and large-scale SVD computations*, 2012. Available from: `http://web.stanford.edu/hastie/TALKS/SVDhastie.pdf`.
18. J. He, L. Balzano, J. Lui, Online robust subspace tracking from partial information, arXiv: 1109.3827.
19. J. He, L. Balzano, A. Szlam, Incremental gradient on the Grassmannian for online foreground and background separation in subsampled video, *2012 IEEE Conference on Computer Vision and Pattern Recognition*, Providence, RI, USA, 2012, 1568–1575. http://doi.org/10.1109/CVPR.2012.6247848
20. L. Balzano, R. Nowak, B. Recht, Online identification and tracking of subspaces from highly incomplete information, *2010 48th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, Monticello, IL, USA, 2010, 704–711. http://doi.org/10.1109/ALLERTON.2010.5706976
21. K. Y. Chiang, I. S. Dhillon, C. J. Hsieh, Using side information to reliably learn low-rank matrices from missing and corrupted observations, *J. Mach. Learn. Res.*, **19** (2018), 3005–3039.
22. P. Netrapalli, U. N. Niranjan, S. Sanghavi, A. Anandkumar, P. Jain, Non-convex robust PCA, arXiv: 1410.7660.

23. Q. Q. Gu, Z. R. Wang, H. Liu, Low-rank and sparse structure pursuit via alternating minimization, *The 19th International Conference on Artificial Intelligence and Statistics*, **51** (2016), 600–609.

24. X. Y. Yi, D. Park, Y. D. Chen, C. Caramanis, Fast algorithms for robust PCA via gradient descent, arXiv: 1605.07784.

25. T. Zhang, Y. Yang, Robust PCA by manifold optimization, *J. Mach. Learn. Res.*, **19** (2018), 3101–3139.

26. H. Q. Cai, J. F. Cai, K. Wei, Accelerated alternating projections for robust principal component analysis, *J. Mach. Learn. Res.*, **20** (2019), 685–717.

27. Y. X. Wang, C. M. Lee, L. F. Cheong, K. C. Toh, Practical matrix completion and corruption recovery using proximal alternating robust subspace minimiaztion, *Int. J. Comput. Vis.*, **111** (2015), 315–344. http://doi.org/10.1007/s11263-014-0746-0

28. B. Recht, A simpler approach to matrix completion, *J. Mach. Learn. Res.*, **12** (2011), 3413–3430.

29. L. Li, W. Huang, I. Gu, Q. Tian, Statistical modeling of complex backgrounds for foreground object detection, *IEEE T. Image Process.*, **13** (2004), 1459–1472. http://doi.org/10.1109/TIP.2004.836169

AIMS Press