AN ALTERNATING MINIMIZATION METHOD FOR ROBUST PRINCIPAL COMPONENT ANALYSIS

YUAN SHEN*, HONGYU XU†, AND XIN LIU‡

Abstract. We focus on solving robust principal component analysis (RPCA) arising from various applications such as information theory, statistics, engineering, and etc. We adopt a model to minimize the sum of observation error and sparsity measurement subject to the rank constraint. To solve this problem, we propose a two-step alternating minimization method. In one step, a symmetric low rank product minimization, which essentially is partial singular value decomposition, can be efficiently solved to moderate accuracy. Meanwhile the second step has closed-form solution. The new proposed approach is almost parameter-free, and its global convergence to a strict local minimizer can be derived under almost no assumption. We compare the proposed approach with some existing solvers and the numerical experiments demonstrate the outstanding performance of our new approach in solving a bunch of synthetic and real RPCA test problems. Especially, we discover the great potential of the proposed approach in solving problem with large size to moderate accuracy.

Key words. robust principal component analysis, symmetric low rank product minimization, singular value decomposition, alternating minimization.

MSC Classifications: 15A18, 65F15, 65K05, 90C06

1. Introduction. Robust principal component analysis (RPCA) is an important multivariate analyzing tool to exploit the special structures of the data. It is taking on increasing importance in the applications frequently arising from diverse fields in statistics, computer vision, signal processing, data mining or compression, and other domains [35, 28, 15, 27]. In the scenario where the data involve both low-rank and sparse structures, that is low-rank structure possibly being contaminated by impulsive noise with high magnitude, the RPCA is able to restore the original data which preserves most of its information. For instance, the frames of surveillance video includes static scenic background and moving objects such as pedestrians, automobiles, and etc., thus the moving ones represent the sparse component and the scene relates to the low-rank part. For a collection of text documents, the low-rank component could capture common words used in all the documents while the sparse component may capture the few key words that best distinguish each document from others [6]. In the application of face recognition, the face images are taken under varying illumination. The underlying face can be represented by low-dimensional linear subspace [2], the errors caused by shadows and specularities are large in magnitude, but sparse in the spatial domain. In the aforementioned applications, the underlying problem can be characterized as: recover a low-rank matrix from large but sparse errors.

Mathematically, the RPCA problem can be described as: recover a low-rank matrix Z and a sparse matrix S from the observation $D = Z + S + N$ which contains a noise matrix N. By assuming the magnitude of additive noise N is small and i.i.d. Gaussian, the original RPCA model can then be formulated as follows:

$$\min_{S,Z\in\mathbb{R}^{m\times n}} \quad \|D - Z - S\|_{\mathrm{F}}^2 + \frac{\mu}{\tau}\,\mathrm{rank}(Z) + \mu\|S\|_0, \tag{1.1}$$

where μ and τ are weight parameters and the ℓ_0 pseudo-norm term $\|S\|_0$ stands for the number of nonzero entries of S. Unfortunately, the original RPCA model combines rank and ℓ_0 pseudo-norm minimization in its objective, and both of them are not only nondifferentiable, but also discontinuous. Computing a global optimal solution of (1.1) has been proven to be NP-hard and intractable [1].

*School of Applied Mathematics, Nanjing University Of Finance & Economics, CHINA (ocsiban@126.com)Research supported by National Natural Science Foundation of China grants 11401295 and by Provincial Natural Science Foundation of Jiangsu grants BK20141007 and by Major Program of the National Social Science Foundation of China under Grant 12&ZD114 and by National Social Science Foundation of China under Grant 15BGL58 and by Social Science Foundation of Jiangsu Province under Grant 14EUA001 and by Qinglan Project of Jiangsu Province.

†School of Applied Mathematics, Nanjing University Of Finance & Economics, CHINA(2513509447@qq.com)

‡State Key Laboratory of Scientific and Engineering Computing, Academy of Mathematics and Systems Science, Chinese Academy of Sciences, and University of Chinese Academy of Sciences, China (liuxin@lsec.cc.ac.cn). Research supported in part by NSFC grants 11471325, 91530204 and 11622112, the National Center for Mathematics and Interdisciplinary Sciences, CAS, and Key Research Program of Frontier Sciences, CAS.

Encouraged by the successful application of ℓ_1 minimization to recover sparse signals in compressive sensing (see, for example, [8, 14]), and the successful application of nuclear norm minimization $||\cdot||_*$ to approximate the rank minimization $\mathrm{rank}(\cdot)$ in matrix completion (see, for instance, [7, 9]), we can relax the ℓ_0 pseudo-norm and the rank minimization by ℓ_1-norm and nuclear norm, respectively. Hence, it leads to the following convex surrogate problem,

$$\min_{S,Z\in\mathbb{R}^{m\times n}} \quad \|D-Z-S\|_{\mathrm{F}}^2+\frac{\mu}{\tau}||Z||_*+\mu\|S\|_1.$$

Specially, when the observation matrix D is free of additive noise, i.e., $N=0$, the fidelity term $\|D-Z-S\|_{\mathrm{F}}^2$ can be replaced by equality constraints $Z+S=D$. Consequently, we arrive at the following principal component pursuit (PCP) model.

$$\begin{aligned} \min_{S,Z\in\mathbb{R}^{m\times n}} \quad & \|Z\|_*+\tau\|S\|_1, \\ \text{s.t.} \quad & Z+S=D. \end{aligned} \tag{1.2}$$

The PCP model (1.2) was demonstrated to have encouraging numerical performance in the early study of it [40, 6]. So it has been studied extensively from the aspects of both theory and application, and there have been abundant of efficient and scalable algorithms including augmented Lagrange method (ALM) [24], alternating direction method of multipliers (ADMM) [42], fast alternating minimization (FAM) [32], iteratively reweighted least squares (IRLS) [4], proximal gradient methods (e.g. see [40]), etc. In the ADMM scheme, the augmented Lagrangian function of model (1.2) was minimized with respect to one variable, either Z or S at a time, while fixing the other at its latest value, and then the Lagrangian multiplier is updated after each round of such alternating minimization. The ADMM framework for solving RPCA has been implemented by Yuan and Yang [42] in a code entitled LRSD (low rank and sparse matrix decomposition), and by Lin, Chen, Wu and Ma [24] in a code called IALM (inexact augmented Lagrangian method). Note that LRSD and IALM are essentially the same algorithm based on the same model, and both of them need to do SVD computation in each step. However, due to implementation differences, the LRSD and IALM often behave quite differently, and their major difference lies at the computation of SVD. The LRSD uses the subroutine "DGESVD" in the LAPACK library [1] to calculate full SVD through Matlab's external interface. The IALM uses a more advanced partial SVD technique implemented in the SVD package PROPACK [2]. The partial SVD scheme begins with a small rank estimate and increases it during iterations. This strategy can be significantly faster than full SVD at beginning while the estimate rank is relatively small so that only a small number of singular values and vectors are calculated at each iteration. However, with the increasing of the rank estimate, the number of eigenpairs to be calculated increases, and its computational advantage diminishes, and completely vanishes when the full SVD is required. The proximal gradient method had been applied to solve model (1.2) (e.g., see [40]), some accelerated versions can be found in, such as, [25], utilizing Nesterov's acceleration approach [30]. In a recent work [29]. the classical ideas from Frank-Wolfe and proximal gradient methods were combined. It exploits Frank-Wolfe to update the low-rank component with rank-one SVD and exploits the proximal step for the sparse term. In [34], a variant model of (1.2) with partial observation was discussed, and they proposed a variant of ADMM to solve it. In another recent work [38], the authors proposed a greedy approach called rank-one matrix pursuit to solve matrix completion. At each iteration, it pursues a rank-one matrix basis generated by the top singular vector pair of the current approximation residual and update the weights for all rank-one matrices obtained up to the current iteration. The authors claimed that it can be extended to solve RPCA model without any difficulty.

[1] `http://www.netlib.org/lapack/`
[2] `http://sun.stanford.edu/~rmunk/PROPACK/`

Due to the convexity of model (1.2), the convergence of the algorithms above mentioned have been richly and thoroughly studied. The limitation of PCP model lies in the expensive computational cost required by the full SVD for nuclear-norm minimization in every step or in the last a few steps in IALM. It becomes unaffordable when the sizes of the underlying matrices grow. To completely avoid the expensive computation of full SVD, nonconvex decomposition models have been proposed to take the place of the nuclear norm term. e.g., in [33], the authors suggested to adopt such a nonconvex RPCA model:

$$\begin{aligned} \min_{S\in\mathbb{R}^{m\times n},X\in\mathbb{R}^{m\times p},Y\in\mathbb{R}^{n\times p}} \quad & \|S\|_1, \\ \text{s.t.} \quad & XY+S=D, \end{aligned} \tag{1.3}$$

where the low-rank matrix Z is replaced by a rank$-p$ factorization, and hence the variable scale is significantly reduced from $2mn$ to $mn+(m+n)p$. A premise of this model is a priorly known estimate rank p or a strategy to pursue it adaptively. The nonconvex RPCA model (1.3) is solved by the classic ADMM in [33], i.e., minimizing the augmented Lagrangian function w.r.t. X, Y, and S in a Gaussian-Seidel scheme and then updating the multipliers. Such ADMM [3] significantly reduces the CPU time needed to obtain a desirable solution, compared with the algorithms based on nuclear norm minimization, such as LRSD, and specially on some difficult instances.

For general convex multi-block problem, the ADMM does not necessarily produce a convergent sequence to the stationary point, see [12] for counterexamples, though the convergence can still be derived by imposing very restrictive assumptions or modifying the original ADMM, e.g., see [18, 13, 17, 23]. The nonconvexity is also a big challenging issue in the theoretical analysis, and it has been shown that, even for a general two-block nonconvex problem, the ADMM might not converge, see Section 3.2.1 of [5]. For special problems, see e.g. [19, 37], the convergence of ADMM can be established. As a consequence, for the problem (1.3) which is nonconvex as well as multi-block, in [33], the authors can only prove that any clustering point of LMaFit is a stationary point to (1.3), but it is still not clear whether LMaFit always provides convergent subsequence.

The main limitations of ADMM for nonconvex multi-block optimization problems contain two folds. Firstly, it is almost impossible to deliver convergence to local minimizer as the ADMM framework with respect to the dual variable is based on dual gradient ascent and hence belongs to first-order method. Secondly, ADMM for nonconvex problems often suffers from slow convergence when the dual step length is improperly set.

To reach a compromise between computation speed and convergence guarantee, we would like to construct a new algorithm enjoying the stunningly good computation speed of SVD-free algorithm as LMaFit, while better convergence results can be established without additional assumptions at the same time. For the sake of new algorithm, we consider a slightly different model with (1.3), in which the additive noise term is considered.

$$\begin{aligned} \min_{S,Z\in\mathbb{R}^{m\times n}} \quad & \tilde{f}(S,Z) := \|D-Z-S\|_{\mathrm{F}}^2+\mu\|S\|_1, \\ \text{s.t.} \quad & \mathrm{rank}(Z)\le p. \end{aligned} \tag{1.4}$$

It is not difficult to verify (1.4) has an equivalent nonconvex decomposition form as follows.

$$\min_{S\in\mathbb{R}^{m\times n},X\in\mathbb{R}^{m\times p},Y\in\mathbb{R}^{n\times p}} \quad f(S,X,Y) := \tilde{f}(S,XY^\top) = \frac{1}{2}\|D-XY^\top-S\|_{\mathrm{F}}^2+\mu\|S\|_1. \tag{1.5}$$

Despite of the nonconvexity, a natural way to solve the above model is to alternatively minimize the objective over variables S and variables (X, Y) in turn.

[3] The corresponding solver is entitled LMaFit: `http://lmafit.blogs.rice.edu/`

The above mentioned alternating minimization approach belongs to the block coordinate descent (BCD) method, e.g. see [36, 39, 31, 3], or [41] for a complete review. The convergence of BCD can be guaranteed under convex assumption, e.g., see [31, 3]. However, the theoretical justification is far from complete when the method is extended to solve a general nonconvex problem. Some researchers have made some attempts to deal with some specific nonconvex problem. Recently, the authors of [20] proved the convergence of a class of BCD for solving a special class of nonconvex problems. For solving another class of special nonconvex feasibility problems, Li and Pong proposed a modified douglas-rachford splitting method [22] which can be viewed as a special BCD. References [20, 22, 21] work on special classes of nonconvex problems where the nonconvexity only occurs on the separable part. Nevertheless, it is still unknown whether the classical alternating minimization can ensure the convergence to a stationary point in nonconvex programming such as (1.5), where the nonconvexity lies in the coupled term. It has been shown in [24, 10, 16, 11, 6] that, the solution of problem (1.2) is equivalent to that of (1.4) under some suitable conditions.

In this paper, we propose a new alternating minimization algorithm for solving (1.5). There are two main contributions of this paper. Firstly, we can prove the global convergence of the iterative sequence generated by our new algorithm converges to a local minimizer providing that the subproblems are solved exactly. Considering that one of the subproblem is a low-rank approximation, which is essentially a partial SVD, it can be solved exactly by low-rank approximation solver [26], or any existing partial SVD solver which is little bit more costly. The other subproblem enjoys closed form solution. Thus, the assumption under which the global convergence can be guaranteed is naturally satisfied. This result is much stronger than the existing results for BCD in solving nonconvex model such as [20, 22, 33, 21]. In addition, some existing approaches (e.g., the method in [22] by Li and Pong) can only handle the case where the nonconvexity lies in the separable part, in contrast, the nonconvexity in our model occurs on the coupled term. The second main contribution is the promising numerical behavior of the proposed algorithm. Our preliminary experiments show that the new algorithm, with efficient low-rank approximation solver SLRP [26] embedded, can be much faster than the algorithms based on nuclear norm, and even outperforms LMaFit in solving many test problems. It is also worthy of mentioning that if the (X,Y)-related subproblem is only solved to stationary point, as the efficient low-rank approximation or partial SVD solvers can only guarantee global convergence to stationary point, we can still establish the global convergence of the resulting variant of our approach to stationary point. Last but not least, there is only model parameter μ to be tuned in our approach, but no algorithmic parameter. Therefore, our approach is more robust and easy to be implemented.

The rest of this paper is organized as follows. We will present our algorithm framework in Section 2. The theoretical analysis will be established in Section 3. In Section 4, we will display the numerical experiments about the comparison between our proposed algorithm and some state-of-the-art algorithms. Conclusions will be drawn in the last section.

2. New algorithm. First we present a concrete description of our new algorithm framework to solve (1.4). The main steps are to update S, Z in turn with the other fixed.

$$S^k = \underset{S\in\mathbb{R}^{m\times n}}{\operatorname{argmin}} \frac{1}{2}\|D - Z^{k-1} - S\|_{\mathrm{F}}^2 + \mu\|S\|_1, \tag{2.1}$$

$$Z^k = \underset{Z\in\mathbb{R}^{m\times n}}{\operatorname{argmin}} \frac{1}{2}\|D - Z - S^k\|_{\mathrm{F}}^2, \quad \text{s.t.} \quad \operatorname{rank}(Z) \le p. \tag{2.2}$$

The solution of S-subproblem (2.1) has the following closed-form

$$S^k = \mathcal{S}_\mu\left(D - Z^{k-1}\right)$$

where $\mathcal{S}_\mu(\cdot)$ is the element-wise soft shrinkage operator defined by

$$(S^k)_{i,j} = \text{sign}((D - Z^{k-1})_{i,j}) \cdot \max(|(D - Z^{k-1})_{i,j}| - \mu, 0).$$

The solution of Z-subproblem (2.2) is actually a low-rank approximation of $D - S^k$, which can be solved efficiently by SLRPGN (see [26]), which is a recent developed fast low-rank approximation solver. According to [26], we have

PROPOSITION 2.1. *[26] Let X^k be a global solution of the following symmetric low-rank product (SLRP) problem*

$$\underset{X \in \mathbb{R}^{m \times p}}{\text{argmin}} \quad \frac{1}{2}\|(D - S^k)(D - S^k)^\top - XX^\top\|_{\text{F}}^2, \tag{2.3}$$

Y^k is defined by

$$Y^k = (D - S^k)^\top X^k \left((X^k)^\top X^k\right)^{-1}. \tag{2.4}$$

Then $Z^k := X^k(Y^k)^\top$ solves (2.2) *globally.*

Now, we can describe the framework of our new algorithm as follows.

Algorithm 1: Alternating Minimization for RPCA

1 Input $D \in \mathbb{R}^{m \times n}$.
2 Initialize an $m \times n$ matrix $Z^0 := 0$ and set $k := 1$.
3 **while** *not "converged"* **do**
4 Compute S^k by (2.1).
5 Compute Z^k by:
6 i) Solve (2.3) to global optimality (stationarity), and the global minimizer (stationary point) is denote by X^k;
7 ii) Calculate Y^k by (2.4).
8 iii) Update $Z^k = X^k(Y^k)^\top$.
9 Increment k and continue.

As to be shown in the next section, which extent Step 6 of Algorithm 1, i.e. the X-subproblem, is solved to determines the guaranteed optimality of the clustering points. It is worthy of mentioning that the global solution of Step 6 is polynomial time solvable, as it is essentially partial SVD which can be fulfilled by simultaneous subspace iteration (see [26] for details). However, to pursue fast convergence, we would suggest to use fast partial SVD solvers such as LANSVD[4], LOBPCG[5], LMSVD [6] or a recent developed low-rank approximation solver SLRPGN [26]. Here, we particularly recommend [26], due to its efficiency, robustness and better fitness to subproblem (2.3). Moreover, although only the convergence to stationary point can be guaranteed in theory, in practice, SLRPGN can always provide a global minimizer.

3. Convergence. Before establishing the convergence results for Algorithm 1, we first list the optimality condition of (1.5) which is equivalent to (1.4).

[4] http://sun.stanford.edu/~rmunk/PROPACK/
[5] https://cn.mathworks.com/matlabcentral/fileexchange/48-lobpcg-m
[6] http://https://cn.mathworks.com/matlabcentral/fileexchange/46875-lmsvd-m

PROPOSITION 3.1. *$(\hat{S}, \hat{X}, \hat{Y})$ is a stationary point of* (1.5)*, if and only if the following first-order optimality conditions hold*

$$\begin{array}{rcl} \hat{S} - D + \hat{X}\hat{Y}^\top + \mu\partial\|\hat{S}\|_1 & \ni & 0; \\ (D - \hat{X}\hat{Y}^\top - \hat{S})\hat{Y} & = & 0; \\ \hat{X}^\top(D - \hat{X}\hat{Y}^\top - \hat{S}) & = & 0. \end{array} \tag{3.1}$$

REMARK 3.2. *It is easy to verify that the first optimality condition in* (3.1) *is equivalent to*

$$\mathcal{S}_\mu(D - \hat{X}\hat{Y}^\top) = \hat{S}. \tag{3.2}$$

LEMMA 3.3. *Let $\{(S^k, X^k, Y^k)\}$ be the iterative sequence generated by Algorithm 1 with X-subproblem* (2.3) *being solved to stationarity, then $\{(S^k, X^k, Y^k)\}$ is bounded and consequently has accumulation point.*

Proof. It is clear that the iterates generated by Algorithm 1 have monotonically non-increasing function values. Namely,

$$\tilde{f}(S^{k+1}, Z^k) \le \tilde{f}(S^k, Z^k) \le \tilde{f}(S^k, Z^{k-1}) \le \cdots \le \tilde{f}(S^1, Z^1) \le \tilde{f}(S^1, Z^0). \tag{3.3}$$

On the other hand, it holds that

$$\tilde{f}(S^{k+1}, Z^k) = \frac{1}{2}\|D - Z^k - S^{k+1}\|_{\mathrm{F}}^2 + \mu\|S^{k+1}\|_1 \ge \mu\|S^{k+1}\|_1. \tag{3.4}$$

Consequently, $\{S^k\}$ is bounded.

By using the triangle inequality and the fact that,

$$Z^k = \underset{Z\in\mathbb{R}^{m\times n}}{\operatorname{argmin}} \frac{1}{2}\|D - Z - S^k\|_{\mathrm{F}}^2, \text{ s.t. } \operatorname{rank}(Z) \le p,$$

we obtain

$$||Z^k||_{\mathrm{F}}^2 - \frac{1}{2}||D - S^k||_{\mathrm{F}}^2 \le \frac{1}{2}||D - Z^k - S^k|| \le \frac{1}{2}||D - 0 - S^k||_{\mathrm{F}}^2,$$

which implies

$$\|Z^k\|_{\mathrm{F}}^2 \le \|D - S^k\|_{\mathrm{F}}^2 \le \|D\|_{\mathrm{F}}^2 + \|S^k\|_{\mathrm{F}}^2.$$

Together with the boundedness of $\{S^k\}$, $\{Z^k\}$ is bounded.

Finally, it follows from the boundedness of $\{Z^k\}$, (2.3) and (2.4) that $\{X^k\}$ and $\{Y^k\}$ are bounded, which complete the proof.

□

THEOREM 3.4. *Let $\{(S^k, X^k, Y^k)\}$ be the iterative sequence generated by Algorithm 1 with X-subproblem* (2.2) *being solved to stationarity, and suppose $(\hat{S}, \hat{X}, \hat{Y})$ is an accumulation point of $\{(S^k, X^k, Y^k)\}$. Then $(\hat{S}, \hat{X}, \hat{Y})$ satisfies the first-order optimality conditions* (3.1)*.*

Proof. First, we note that $f(S^k, X^k, Y^k)$ is monotonically decreasing and bounded below. Therefore, there exists f^* so that

$$\lim_{k\to\infty} f(S^k, X^k, Y^k) = f^*. \tag{3.5}$$

Without loss of generality, we assume that $(S^{k_j}, X^{k_j}, Y^{k_j})$ converges to $(\hat{S}, \hat{X}, \hat{Y})$ with $j \to \infty$. According to Step 4 of Algorithm 1, we have

$$S^{k_j+1} = \mathcal{S}_\mu\left(D - X^{k_j}(Y^{k_j})^\top\right),$$

i.e., S^{k_j+1} solves

$$\min_{S\in\mathbb{R}^{m\times n}} \frac{1}{2}\|D - X^{k_j}(Y^{k_j})^\top - S\|_{\mathrm{F}}^2 + \mu\|S\|_1.$$

Namely, we have

$$\begin{aligned}
&f(S^{k_j}, X^{k_j}, Y^{k_j}) - f(S^{k_j+1}, X^{k_j}, Y^{k_j}) \\
&= \left\{\frac{1}{2}||D - X^{k_j}(Y^{k_j})^\top - S^{k_j}||_{\mathrm{F}}^2 + \mu\|S^{k_j}\|_1\right\} - \left\{\frac{1}{2}||D - X^{k_j}(Y^{k_j})^\top - S^{k_j+1}||_{\mathrm{F}}^2 + \mu\|S^{k_j+1}\|_1\right\} \\
&= \langle S^{k_j} - S^{k_j+1}, S^{k_j+1} + X^{k_j}(Y^{k_j})^\top - D\rangle + \mu(\|S^{k_j}\|_1 - \|S^{k_j+1}\|_1) + \frac{1}{2}\|S^{k_j} - S^{k_j+1}\|_{\mathrm{F}}^2. \qquad (3.6)
\end{aligned}$$

Since S^{k_j+1} solves

$$\min_{S\in\mathbb{R}^{m\times n}} \frac{1}{2}\|D - X^{k_j}(Y^{k_j})^\top - S\|_{\mathrm{F}}^2 + \mu\|S\|_1, \tag{3.7}$$

hence S^{k_j+1} satisfies the optimality condition of (3.7), i.e.,

$$\mu(\|S'\|_1 - \|S^{k_j+1}\|_1) + (S' - S^{k_j+1}, S^{k_j+1} + X^{k_j}(Y^{k_j})^\top - D) \geq 0, \ \ \forall \ \ S' \in \mathbb{R}^{m\times n}. \tag{3.8}$$

On one hand, the sum of first two terms in the right hand side of (3.6) is nonnegative by letting $S' = S^{k_j}$ in (3.8); on the other hand, invoking (3.3), we obtain

$$f(S^{k_j+1}, X^{k_j}, Y^{k_j}) \ \geq \ f(S^{k_j+1}, X^{k_j+1}, Y^{k_j+1}) \ \geq \ \cdots \ \geq \ f(S^{k_{j+1}}, X^{k_{j+1}}, Y^{k_{j+1}}). \tag{3.9}$$

Using the above two facts, (3.6) can be formulated into:

$$f(S^{k_j}, X^{k_j}, Y^{k_j}) - f(S^{k_{j+1}}, X^{k_{j+1}}, Y^{k_{j+1}}) \geq \frac{1}{2}\|S^{k_j} - S^{k_j+1}\|_{\mathrm{F}}^2. \tag{3.10}$$

Summing both sides of (3.10) from $j = 1$ to ∞, and using the lower boundedness of $f(S, X, Y)$, we obtain that the series $\sigma_{j=1}^{\infty}\|S^{k_j} - S^{k_j+1}\|_{\mathrm{F}}^2$ is convergent, hence

$$\|S^{k_j} - S^{k_j+1}\|_{\mathrm{F}}^2 \to 0,$$

or equivalently,

$$\|S^{k_j} - \mathcal{S}_\mu\left(D - X^{k_j}(Y^{k_j})^\top\right)\| \to 0.$$

Take the limit $j \to \infty$ on both sides, we arrive at the first condition of (3.1), i.e.,

$$\hat{S} = \mathcal{S}_\mu \left(D - \hat{X}\hat{Y}^\top \right).$$

On the other hand, pre-multiplying $\left((X^{k_j})^\top X^{k_j}\right)$ to the both sides of the Y updating formula (2.4), we obtain

$$\left((X^{k_j})^\top X^{k_j}\right)(Y^{k_j})^\top = (X^{k_j})^\top (D - S^{k_j}), \tag{3.11}$$

which gives the third condition of (3.1) after taking the limit $j \to \infty$ from both sides.

Finally, since X^{k_j} is a stationary point of the SLRP problem (2.3), we have

$$(X^{k_j})^\top (D - S^{k_j})(D - S^{k_j})^\top X^{k_j} = (X^{k_j})^\top X^{k_j} (X^{k_j})^\top X^{k_j}, \tag{3.12}$$

which implies

$$((X^{k_j})^\top X^{k_j})^{-1} (X^{k_j})^\top (D - S^{k_j})(D - S^{k_j})^\top X^{k_j} = (X^{k_j})^\top X^{k_j}. \tag{3.13}$$

Substituting the Y updating formula (2.4) into (3.13), we obtain

$$(D - S^{k_j}) Y^{k_j} = X^{k_j} ((Y^{k_j})^\top Y^{k_j}), \tag{3.14}$$

which finally implies the second condition of (3.1) after taking the limit $j \to \infty$ from both sides. Now, we recall Proposition 3.1 and complete the proof. □

Next, we establish a stronger convergence result in the case that the SLRP subproblem (2.3) is solved to global optimality.

THEOREM 3.5. *Let $\{(S^k, X^k, Y^k)\}$ be the iterative sequence generated by Algorithm 1 with X-subproblem* (2.3) *being solved to global optimality, and suppose $(\hat{S}, \hat{X}, \hat{Y})$ is an accumulation point of $\{(S^k, X^k, Y^k)\}$. Then $(\hat{S}, \hat{X}, \hat{Y})$ is a local optimizer of the decomposition model* (1.3)*. Specially, when $\hat{S}$ has at least one zero element, $(\hat{S}, \hat{X}, \hat{Y})$ is strictly local optimizer.*

Proof. According to Lemma 3.4, $(\hat{S}, \hat{X}, \hat{Y})$ shall satisfy the first-order optimality conditions (3.1). Moreover, it is not difficult to verify that $(\hat{X}, \hat{Y})$ is the global solution of

$$\min_{X \in \mathbb{R}^{m\times p}, Y \in \mathbb{R}^{n\times p}} f(\hat{S}, X, Y).$$

Suppose that $(\hat{S} + \Delta_S, \hat{X} + \Delta_X, \hat{Y} + \Delta_Y)$ is in a neighborhood of $(\hat{S}, \hat{X}, \hat{Y})$. Let

$$\tilde{\Delta}_S = -\hat{S} + \mathcal{S}_\mu \left(D - (\hat{X} + \Delta_X)(\hat{Y} + \Delta_Y)^\top \right)$$

Then, we have

$$\begin{aligned} \tilde{\Delta}_S &= -\mathcal{S}_\mu \left(D - \hat{X}\hat{Y}^\top \right) + \mathcal{S}_\mu \left(D - (\hat{X} + \Delta_X)(\hat{Y} + \Delta_Y)^\top \right) \\ &= -\mathcal{S}_\mu \left(D - \hat{X}\hat{Y}^\top \right) + \mathcal{S}_\mu \left(D - \hat{X}\hat{Y}^\top - \Delta_Z \right) \end{aligned} \tag{3.15}$$

where

$$\Delta_Z := \Delta_X \hat{Y}^\top + \hat{X}\Delta_Y^\top + \Delta_X \Delta_Y^\top. \tag{3.16}$$

With fixed μ and $D - \hat{X}\hat{Y}^\top$, we define the operator $\mathbf{P}_{\mathbf{Z}}(\cdot) : \mathbb{R}^{m\times n} \mapsto \mathbb{R}^{m\times n}$ as the following

$$\mathbf{P}_{\mathbf{Z}}(A) := \mathcal{S}_\mu\left(D - \hat{X}\hat{Y}^\top\right) - \mathcal{S}_\mu\left(D - \hat{X}\hat{Y}^\top - A\right), \qquad A \in \mathbb{R}^{m\times n}. \tag{3.17}$$

Specially, when A takes the form of Δ_Z, it holds that $\mathbf{P}_{\mathbf{Z}}(\Delta_Z) = -\tilde{\Delta}_S$.

For sufficiently small Δ_Z, it is easy to verify the following equality

$$\langle \mathbf{P}_{\mathbf{Z}}(\Delta_Z), \Delta_Z - \mathbf{P}_{\mathbf{Z}}(\Delta_Z)\rangle = 0. \tag{3.18}$$

On the other hand we have

$$\begin{aligned}
& f(\hat{S}+\Delta_S, \hat{X}+\Delta_X, \hat{Y}+\Delta_Y) - f(\hat{S},\hat{X},\hat{Y}) \;\geq\; f(\hat{S}+\tilde{\Delta}_S, \hat{X}+\Delta_X, \hat{Y}+\Delta_Y) - f(\hat{S},\hat{X},\hat{Y}) \\
=\; & \frac{1}{2}\|D - (\hat{X}+\Delta_X)(\hat{Y}+\Delta_Y)^\top - (\hat{S}+\tilde{\Delta}_S)\|_{\mathrm{F}}^2 \;+\; \mu\|\mathcal{S}_\mu\left(D - (\hat{X}+\Delta_X)(\hat{Y}+\Delta_Y)^\top\right)\|_1 \\
& -\frac{1}{2}\|D - \hat{X}\hat{Y}^\top - \hat{S}\|_{\mathrm{F}}^2 - \mu\|\hat{S}\|_1 \;=\; \frac{1}{2}\|(D - \hat{X}\hat{Y}^\top - \hat{S}) - (\Delta_Z + \tilde{\Delta}_S)\|_{\mathrm{F}}^2 \\
& -\frac{1}{2}\|D - \hat{X}\hat{Y}^\top - \hat{S}\|_{\mathrm{F}}^2 \;+\; \mu\|\mathcal{S}_\mu\left(D - (\hat{X}+\Delta_X)(\hat{Y}+\Delta_Y)^\top\right)\|_1 - \mu\|\hat{S}\|_1 \\
=\; & -\langle D - \hat{X}\hat{Y}^\top - \hat{S}, \Delta_Z + \tilde{\Delta}_S\rangle + \frac{1}{2}\|\Delta_Z + \tilde{\Delta}_S\|_{\mathrm{F}}^2 \;+\; \mu\|\mathcal{S}_\mu\left(D - \hat{X}\hat{Y}^\top - \Delta_Z\right)\|_1 - \mu\|\hat{S}\|_1 \\
=\; & -\langle D - \hat{X}\hat{Y}^\top - \hat{S}, \Delta_Z - \mathbf{P}_{\mathbf{Z}}(\Delta_Z)\rangle + \frac{1}{2}\|\Delta_Z - \mathbf{P}_{\mathbf{Z}}(\Delta_Z)\|_{\mathrm{F}}^2 \;+\; \mu\|\mathcal{S}_\mu\left(D - \hat{X}\hat{Y}^\top - \Delta_Z\right)\|_1 - \mu\|\hat{S}\|_1 \\
\geqslant\; & \Psi_2,
\end{aligned} \tag{3.19}$$

where $\Psi_2 := \sum\limits_{1\leq i\leq m,\, 1\leq j\leq n} (\Psi_2)_{ij}$, and

$$\begin{aligned}
(\Psi_2)_{ij} := & -(D - \hat{X}\hat{Y}^\top - \hat{S})_{ij}\cdot(\Delta_Z - \mathbf{P}_{\mathbf{Z}}(\Delta_Z))_{ij} + \frac{1}{2}\cdot(\Delta_Z - \mathbf{P}_{\mathbf{Z}}(\Delta_Z))_{ij}^2 \\
& +\mu\cdot\left|\mathcal{S}_\mu\left(D - \hat{X}\hat{Y}^\top - \Delta_Z\right)_{ij}\right| - \mu\cdot|\hat{S}_{ij}|, \quad 1\leq i\leq m, \quad 1\leq j\leq n.
\end{aligned}$$

Here, the first inequality follows the fact that $\hat{S}+\tilde{\Delta}_S$ solves $\min\limits_{S\in\mathbb{R}^{m\times n}} f(S, \hat{X}+\Delta_X, \hat{Y}+\Delta_Y)$.

Invoking the assumption that the SLRP subproblem (2.3) be solved to global optimality, we have:

$$||D - Z' - S^{k_j}||_{\mathrm{F}}^2 \geq ||D - X^{k_j}(Y^{k_j})^\top - S^{k_j}||_{\mathrm{F}}^2,$$

which gives

$$\langle Z' - X^{k_j}(Y^{k_j})^\top, -D + X^{k_j}(Y^{k_j})^\top + S^{k_j}\rangle \geq -\frac{1}{2}||Z' - X^{k_j}(Y^{k_j})^\top||_{\mathrm{F}}^2.$$

Take the limit $j \to \infty$, and let $Z' = (\hat{X} + \Delta_X)(\hat{Y} + \Delta_Y)^\top$, we obtain

$$\langle \Delta_Z, -D + \hat{X}\hat{Y}^\top + \hat{S} \rangle \geq -\frac{1}{2}||\Delta_Z||_{\mathrm{F}}^2.$$

Together with (3.18), we have

$$-\langle D - \hat{X}\hat{Y}^\top - \hat{S}, \Delta_Z \rangle + \frac{1}{2}\|\Delta_Z - \mathbf{P}_{\mathbf{Z}}(\Delta_Z)\|_{\mathrm{F}}^2 \geq -\frac{1}{2}||\mathbf{P}_{\mathbf{Z}}(\Delta_Z)||_{\mathrm{F}}^2. \tag{3.20}$$

We denote $\Psi_1 := \sum_{1 \leq i \leq m,\, 1 \leq j \leq n} (\Psi_1)_{ij}$ where

$$(\Psi_1)_{ij} := -(D - \hat{X}\hat{Y}^\top - \hat{S})_{ij} \cdot (\Delta_Z)_{ij} + \frac{1}{2}((\Delta_Z)_{ij} - \mathbf{P}_{\mathbf{Z}}(\Delta_Z)_{ij})^2 + \frac{1}{2}(\mathbf{P}_{\mathbf{Z}}(\Delta_Z)_{ij})^2, \quad 1 \leq i \leq m, \quad 1 \leq j \leq n.$$

We notice that both Ψ_1 and Ψ_2 are separable with the entries. We now divide the pair $((D - \hat{X}\hat{Y}^\top)_{ij}, (\Delta_Z)_{ij})$ into seven cases and three groups.

- **Group 1.**
  (1) $(D - \hat{X}\hat{Y}^\top)_{ij} > \mu$;
  (2) $(D - \hat{X}\hat{Y}^\top)_{ij} = \mu$, $(\Delta_Z)_{ij} \leq 0$.
  Denote Ω_1 as the index set with all indices in this group. We have $\mathbf{P}_{\mathbf{Z}}(\Delta_Z)_{ij} = (\Delta_Z)_{ij}$, hence

$$\begin{aligned}(\Psi_1)_{ij} &= -\mu(\Delta_Z)_{ij} + \frac{1}{2}((\Delta_Z)_{ij})^2, \quad \text{for all} (i,j) \in \Omega_1;\\ (\Psi_2)_{ij} &= -\mu(\Delta_Z)_{ij}, \quad \text{for all} (i,j) \in \Omega_1.\end{aligned}$$

- **Group 2.**
  (3) $(D - \hat{X}\hat{Y}^\top)_{ij} < -\mu$;
  (4) $(D - \hat{X}\hat{Y}^\top)_{ij} = -\mu$, $(\Delta_Z)_{ij} \geq 0$.
  Denote Ω_2 as the index set with all indices in this group. We have $\mathbf{P}_{\mathbf{Z}}(\Delta_Z)_{ij} = (\Delta_Z)_{ij}$, hence

$$\begin{aligned}(\Psi_1)_{ij} &= \mu(\Delta_Z)_{ij} + \frac{1}{2}((\Delta_Z)_{ij})^2, \quad \text{for all} (i,j) \in \Omega_2;\\ (\Psi_2)_{ij} &= \mu(\Delta_Z)_{ij}, \quad \text{for all} (i,j) \in \Omega_2.\end{aligned}$$

- **Group 3.**
  (5) $|(D - \hat{X}\hat{Y}^\top)_{ij}| < \mu$;
  (6) $(D - \hat{X}\hat{Y}^\top)_{ij} = \mu$, $(\Delta_Z)_{ij} > 0$;
  (7) $(D - \hat{X}\hat{Y}^\top)_{ij} = -\mu$, $(\Delta_Z)_{ij} < 0$;
  Denote Ω_3 as the index set with all indices in this group. We have $\mathbf{P}_{\mathbf{Z}}(\Delta_Z)_{ij} = 0$, hence

$$(\Psi_1)_{ij} = (\Psi_2)_{ij} = -(D - \hat{X}\hat{Y}^\top)_{ij} \cdot (\Delta_Z)_{ij} + \frac{1}{2}((\Delta_Z)_{ij})^2, \quad \text{for all} (i,j) \in \Omega_3.$$

For convenience, we denote $S_1 = \sum_{(i,j)\in\Omega_1} -\mu(\Delta_Z)_{ij}$, $S_2 = \sum_{(i,j)\in\Omega_1} \frac{1}{2}((\Delta_Z)_{ij})^2$, $S_3 = \sum_{(i,j)\in\Omega_2} \mu(\Delta_Z)_{ij}$, $S_4 = \sum_{(i,j)\in\Omega_2} \frac{1}{2}((\Delta_Z)_{ij})^2$, $S_5 = \sum_{(i,j)\in\Omega_3} -(D - \hat{X}\hat{Y}^\top)_{ij} \cdot (\Delta_Z)_{ij}$, $S_6 = \sum_{(i,j)\in\Omega_3} \frac{1}{2}((\Delta_Z)_{ij})^2$. Therefore, (3.20) implies that

$$\Psi_1 = S_1 + S_2 + S_3 + S_4 + S_5 + S_6 \geq 0. \tag{3.21}$$

On the other hand, it is easy to verify that

$$\Psi_2 = S_1 + S_3 + S_5 + S_6. \tag{3.22}$$

Clearly, $|S_2 + S_4 + S_6| = o(|S_1 + S_3 + S_5|)$, when Δ_Z is sufficiently small. It then directly follows from (3.21) that $S_1 + S_3 + S_5 \geq 0$, when Δ_Z is sufficiently small. Together with $S_6 \geq 0$ and (3.22), we have proved the first assertion.

Suppose $(\hat{S}, \hat{X}, \hat{Y})$ is not a strictly local optimizer, there exists $(\hat{S}+\Delta_S, \hat{X}+\Delta_X, \hat{Y}+\Delta_Y)$ with $(\Delta_S, \Delta_X, \Delta_Y) \neq 0$ in a neighborhood of $(\hat{S}, \hat{X}, \hat{Y})$ satisfying $f(\hat{S} + \Delta_S, \hat{X} + \Delta_X, \hat{Y} + \Delta_Y) = f(\hat{S}, \hat{X}, \hat{Y})$. Therefore we have $S_1 = S_3 = S_5 = 0$, which yields $\Delta_Z = 0$ and $\Delta_S \neq 0$. Moreover, we have both $\hat{S}$ and $\hat{S} + \Delta_S$ solves $\min_{S \in \mathbb{R}^{m \times n}} f(S, \hat{X}, \hat{Y})$, which implies that either $\hat{S} = 0$ or $\Delta_S = 0$. Since $\Delta_S \neq 0$, we have $\hat{S} = 0$ which is a contradiction with the assumption. We complete the proof. □

4. Numerical Experiments. In this section, we demonstrate the numerical performance of our proposed algorithm through solving matrix separation problem with synthetic and real data.

4.1. Setup of experiments. To construct test problems with synthetic data, we mainly generate the low-rank and sparse matrices obeying certain random techniques. Each observe matrix is the summations of one low-rank matrix and one sparse matrix. For problem with real data, we formulate the natural images and surveillance videos into data matrices. For image problem, we corrupt part of the entries of the natural images by impulsive noise, then the corrupted images can be seen as the sum of a approximately low-rank matrix and a sparse matrix. For video problem, surveillance videos can also be regarded as the sum of a low-rank matrix (background) and a sparse matrix (moving objects). We run the test algorithms for a fixed number of iterations, then compare the computing times and visual qualities of the results.

We compare our algorithm with IALM [7], and LMaFit[8]. All codes are in MATLAB, and the experiments are performed in MATLAB version R2015a on a desktop computer with Intel Core i7 at 3.6Ghz, 8GB memory and Windows 7 operating system. The parameters in IALM and LMaFit were left to their default settings, while that in our algorithm were tuned to optimize the performance.

It is worthy of mentioning that for LMaFit and our algorithm, the rank estimate p is an input. It can either be a priorly known upper bound or adaptively tuned in the algorithm. In the numerical experiments here, we assume p is prior information. Hence, in some senses, it is not completely fair to compare with IALM if there is no prior information can be used. Here we include the numerical performance of IALM is just for reference, and give the readers a perception about how the required CPU times vary with the increasing of problem scale in nuclear norm based approach and low-rank decomposition based approach.

It is important to set a fair stopping criterion for different algorithms. Therefore, the experiments were carried out as follows: all variables in the test algorithms are initialized with zero variables and stopped when either a maximal number of iterations "$maxit$" is reached, or the relative error of Z is smaller than some prescribed tolerance tol, i.e.,

$$\text{err}(Z) < tol. \tag{4.1}$$

Since the stopping rules in the codes of LMaFit and IALM are not the same as ours, these two codes were slightly modified by the present authors.

[7] downloaded from `http://perception.csl.uiuc.edu/matrix-rank/Files/inexact_alm_rpca.zip`
[8] downloaded from `http://lmafit.blogs.rice.edu/`

We put emphasize on the comparison of algorithms' time efficiencies for various of problems, and the reported running times are wall-clock times. To compare the quality of the results obtained by different algorithms, we use the relative error with original matrices Z^* i.e.,

$$\text{err}(Z) := \frac{\|Z - Z^*\|_F}{\|Z^*\|_F}, \tag{4.2}$$

as performance indices. Note the RPCA model (1.4) or (1.2) usually does not yield a solution which is exactly the same with the original matrix, i.e., the exact solution of RPCA model is not equivalent to Z^* and S^*. To reach high accuracy, IALM and LMaFit implement a "continuation" strategy on its penalty parameter. Unless otherwise specified, our Algorithm 1 also utilizes such technique by decreasing the weighting parameter μ during the iterative progress in our experiments.

The continuation strategy on parameter μ in our algorithm is implemented as follows:

- Set $\mu^0 = \mu$, $\mu^{\min}$, decreasing factor $\rho \in (0, 1)$, and violation change threshold $\texttt{thresh}$;
- Record "relative violations" $\texttt{vio}(k) = \|D - S^k - Z^k\|_F / \|D\|_F$ for each step;
- When $k = 1, 2, ..., 10$, set $\mu^k = \mu^{k-1}$;
- When $k > 10$,
  (A) When $\texttt{vio}(k)/\texttt{vio}(k-1)) > \texttt{thresh}$ is true for $k, k-1, k-2, k-3, k-4$, i.e., the relative violations do not decrease much in the recent 5 iterations, we activate "continuation": $\mu^k = \max(\rho\mu^{k-1}, \mu^{\min})$ for $k, k+1, ...$;
  (B) Otherwise $\mu^k = \mu^{k-1}$.

The default values of parameters are: $\rho = 0.4$, $\mu^{\min} = \mu^0 * 1e{-}8$, $\texttt{thresh} = 0.9$. We usually take $\mu^0 = 30/\sqrt{m}$, but its setting still depends on the test problem.

4.2. Comparison on random problems. In this subsection, the experiments with synthetic data were done. The setup of our experiments is as follows. The test low-rank matrices $Z^* \in R^{m\times n}$ were generated by the procedure: two random matrices $U \in R^{m\times k}$ and $V \in R^{k\times n}$ with i.i.d. Gaussian entries are created and then rank-p matrix $Z^* = UV$ is assembled. The test matrix $S^* \in R^{m\times n} = \tau S$ where the sparse matrix S is constructed by the following steps: (1) randomly choosing q indices from $m \times n$ indices; (2) assigning the corresponding locations with nonzero i.i.d. standard Gaussian with variance $1/n$ (3) selecting τ as a scalar which makes $S^* \in R^{m\times n}$ roughly the same magnitude as Z^*. We denote $q = \|S^*\|_0$ where $\|\cdot\|_0$ is the L_0 pseudo-norm. We finally construct data matrix D by $D = Z^* + S^*$ or $D = Z^* + S^*$. In the following part, we call rank(Z^*)/m and $\|S^*\|_0/mn$ by "rank ratio" and "density", respectively.

We first preform the speed comparison among the testing algorithms under four settings of rank(Z^*) and $\|S^*\|_0$ with $tol = 1e-4$ and $maxit = 500$. The mean values of the number of iterations, CPU times, and relative errors of the corresponding results obtained on 10 randomly generated problems with 3 dimensional settings are reported in Tables 4.1. Note that the relative errors of Z and S of IALM were not shown, its CPU times and number of iterations are shown as references. Due to excessive computational load, we did not let IALM join in our experiments with problems of size larger than 4000.

Table 4.1 Results of speed performance test for random problems with $tol = 1e-4$

Dimension	Problem settings		LMaFit				New Algorithm				IALM	
	Rank(Z^*)	$\|S^*\|_0$	err(Z)	err(S)	iter	time	err(Z)	err(S)	iter	time	iter	time
m=n=1000	0.01m	0.01mn	8.45e-5	2.71e-2	18.20	0.90	5.47e-5	3.75e-2	17.20	0.78	27.10	7.61
	0.01m	0.05mn	7.13e-5	9.69e-3	20.00	0.98	6.30e-5	1.74e-2	18.00	0.82	28.90	9.18
	0.05m	0.01mn	8.08e-5	2.67e-2	20.10	1.02	6.79e-5	4.71e-2	19.20	1.03	28.00	8.48
	0.05m	0.05mn	6.53e-5	9.11e-3	21.90	1.12	3.95e-5	1.08e-2	20.10	1.08	29.90	10.27
m=n=2000	0.01m	0.01mn	8.46e-5	2.72e-2	18.20	6.54	6.78e-5	4.69e-2	17.30	3.38	28.10	45.13
	0.01m	0.05mn	6.54e-5	9.00e-3	20.00	7.23	3.15e-5	8.71e-3	19.00	3.89	30.00	59.97
	0.05m	0.01mn	8.60e-5	2.87e-2	20.00	7.26	6.17e-5	4.31e-2	18.00	4.38	29.00	64.75
	0.05m	0.05mn	6.05e-5	8.44e-3	22.00	7.68	2.76e-5	7.52e-3	20.00	4.94	30.80	72.60
m=n=4000	0.01m	0.01mn	8.95e-5	2.88e-2	18.00	47.06	3.87e-5	2.68e-2	17.00	13.84	29.00	288.60
	0.01m	0.05mn	6.29e-5	8.60e-3	20.00	48.48	3.94e-5	1.09e-2	18.00	15.25	31.00	329.91
	0.05m	0.01mn	8.69e-5	2.91e-2	20.00	51.73	7.65e-5	5.33e-2	18.00	22.02	30.00	552.89
	0.05m	0.05mn	6.97e-5	9.80e-3	22.00	56.49	3.41e-5	9.36e-3	20.00	24.57	32.00	599.99
m=n=6000	0.01m	0.01mn	8.87e-5	2.84e-2	18.00	148.02	4.13e-5	2.85e-2	17.00	33.19	30.00	922.47
	0.01m	0.05mn	6.16e-5	8.45e-3	20.00	151.14	4.19e-5	1.16e-2	18.00	34.54	32.00	1009.52
	0.05m	0.01mn	8.79e-5	2.93e-2	20.00	159.57	5.16e-5	3.59e-2	22.00	70.59	31.00	1799.65
	0.05m	0.05mn	6.64e-5	9.33e-3	22.00	165.58	5.60e-5	1.55e-2	22.00	70.11	33.00	1892.66
m=n=8000	0.01m	0.01mn	8.67e-5	2.82e-2	18.00	388.87	7.68e-5	5.31e-2	17.00	73.81	31.00	2094.25
	0.01m	0.05mn	5.96e-5	8.19e-3	20.00	377.89	3.14e-5	8.70e-3	19.00	71.40	33.00	2292.03
	0.05m	0.01mn	9.41e-5	3.14e-2	20.00	393.77	3.89e-5	2.70e-2	25.00	172.32	31.00	4153.95
	0.05m	0.05mn	7.04e-5	9.92e-3	22.00	435.43	4.24e-5	1.16e-2	26.00	173.32	33.00	4364.58

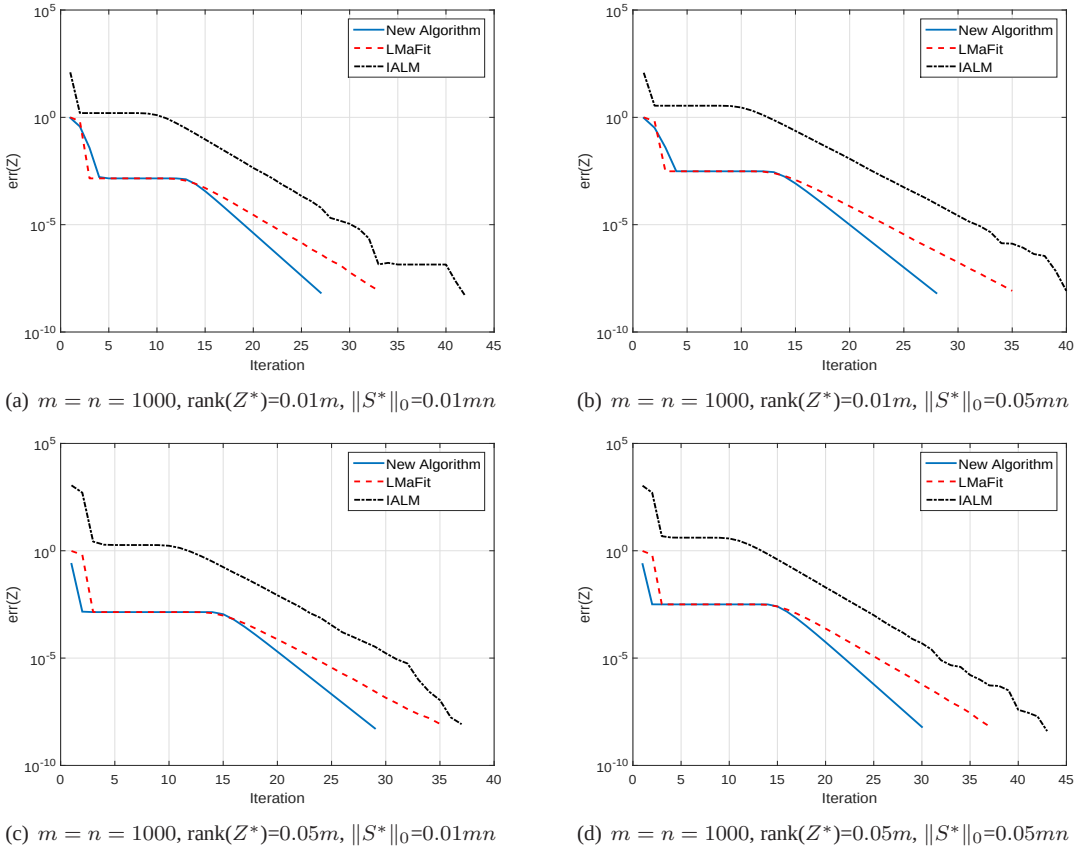
We can observe from Table 4.1 that our algorithm shows satisfactory speed performance compared with other test algorithms. In terms of computing time, our algorithm is at least 10 times faster than IALM, and 10%-130% faster than LMaFit. In terms of convergence rate, our algorithm is around 50% faster than IALM, and 5%-10% faster than LMaFit. On one hand, our algorithm benefits from the low per-iteration cost by avoiding the expensive SVD computation. On the other hand, according to Theorem 3.5, the new iterate obtained by our algorithm satisfies local optimality under mild assumptions while that obtained by LMaFit only satisfies first order optimality condition, hence our algorithm can produce an iterating sequence with better "quality" than that of LMaFit, and it converges faster than LMaFit. Additionally, although both LMaFit and our algorithm are SVD-free algorithms, the per-iteration cost of our algorithm is around 30%-70% of that of LMaFit, and the ratio between the per-iteration costs of two algorithms has a downward tendency as dimension increases. This implies that our algorithm is dimensionally more scalable than LMaFit.

The above experimental results are obtained under mild tolerance setting, we then investigate the performance of test algorithms under more formidable tolerance setting, and report the mean values of the number of iterations, CPU times, and relative errors of the corresponding results obtained on 10 randomly generated problems with 2 dimensional settings with $tol = 1e-8$ in Table 4.2.

Table 4.2 Results of speed performance test for random problems with $tol = 1e-8$

Dimension	Problem settings		LMaFit				New Algorithm				IALM	
	Rank(Z^*)	$\|S^*\|_0$	err(Z)	err(S)	iter	time	err(Z)	err(S)	iter	time	iter	time
m=n=1000	0.01m	0.01mn	8.82e-9	3.00e-6	33.20	1.51	6.59e-9	4.56e-6	27.00	1.23	35.75	13.11
	0.01m	0.05mn	7.67e-9	9.20e-7	35.00	1.61	5.72e-9	1.56e-6	28.25	1.27	49.75	20.28
	0.05m	0.01mn	5.85e-9	1.13e-6	35.50	1.67	3.76e-9	2.61e-6	29.50	1.47	37.75	15.16
	0.05m	0.05mn	7.12e-9	8.67e-7	37.00	1.75	6.13e-9	1.63e-6	30.00	1.52	43.00	17.84
m=n=2000	0.01m	0.01mn	6.56e-9	1.86e-6	34.00	9.16	4.51e-9	3.09e-6	28.25	5.21	48.25	87.19
	0.01m	0.05mn	6.87e-9	9.64e-7	35.00	9.48	5.33e-9	1.47e-6	29.00	5.45	44.25	87.85
	0.05m	0.01mn	7.87e-9	2.34e-6	35.50	10.15	4.66e-9	3.23e-6	28.50	6.77	44.50	109.90
	0.05m	0.05mn	6.82e-9	8.95e-7	37.00	10.40	5.24e-9	1.41e-6	29.50	7.00	49.00	120.70
m=n=4000	0.01m	0.01mn	6.75e-9	2.23e-6	33.70	57.98	3.62e-9	2.49e-6	27.70	21.74	132.50	1430.30
	0.01m	0.05mn	7.25e-9	9.67e-7	35.00	59.61	6.64e-9	1.83e-6	28.00	22.23	164.70	1888.64
	0.05m	0.01mn	8.56e-9	2.76e-6	35.40	65.16	7.07e-9	4.90e-6	28.30	33.27	76.00	1446.63
	0.05m	0.05mn	7.18e-9	9.60e-7	37.00	66.51	3.73e-9	1.01e-6	30.00	35.31	89.50	1709.37

It can be learnt from Table 4.2 that, with the more accurate tolerance setting $tol = 1e-8$, our algorithm still maintains good performance. In terms of computing time, our algorithm is 10%-160% faster than LMaFit. While in terms of iteration number, our algorithm requires around 20% less than LMafit. Compared with the results shown in



(a) $m = n = 1000$, rank(Z^*)=$0.01m$, $\|S^*\|_0$=$0.01mn$

(b) $m = n = 1000$, rank(Z^*)=$0.01m$, $\|S^*\|_0$=$0.05mn$

(c) $m = n = 1000$, rank(Z^*)=$0.05m$, $\|S^*\|_0$=$0.01mn$

(d) $m = n = 1000$, rank(Z^*)=$0.05m$, $\|S^*\|_0$=$0.05mn$

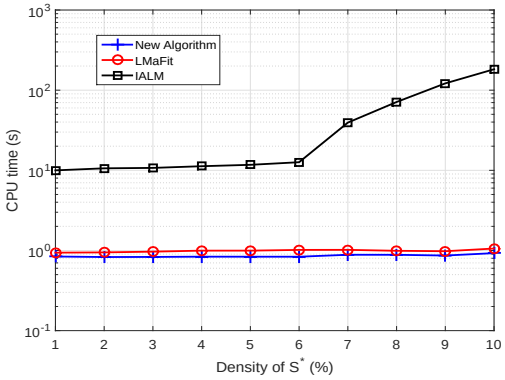
FIG. 1. *Iteration progress of relative error.*

Table 4.1, we observe that the performance gap between our algorithm and LMaFit has a growing trend, this could imply that our algorithm is more favourable for more stringent tolerance setting.

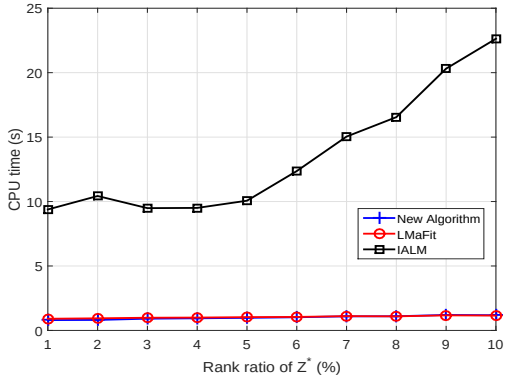
To examine the convergence behavior of the algorithms in a more precise way, in terms of iteration numbers rather than of running time, we ran the algorithms on random problems of sizes $m = n = 1000$ with $tol = 1e-8$ and recorded the relative errors at all iterations. The random problems were generated with 4 different problem settings as described above. The results on these problems are given in Figure 1, where the relative errors are plotted against the iteration numbers. We observe from Figure 1 that all test algorithms have fast and stable convergence rate on the whole iteration progress. We also observe that after the "plateau" between around 5 and 15 iterations, our algorithm has faster convergence rate than LMaFit.

As a supplement to the above experiments, we also investigate the performance change of test algorithms to problem settings. To simplify the experimental setup, we fix size $m = n = 1000$. We set rank(Z^*)=$0.01m$, then let $\|S^*\|_0$ increase from $0.01mn$ to $0.1mn$ with interval $0.01mn$. For each fixed $\|S^*\|_0$, we run 10 random problems and record the mean value of CPU times and relative errors. We then exchange the roles of rank(Z^*) and $\|S^*\|_0$, i.e., we set $\|S^*\|_0 = 0.01mn$, and let rank(Z^*) increase from $0.01m$ to $0.1m$ with interval $0.01m$. For each fixed rank(Z^*), we still run 10 random problems and record the mean value of CPU times and relative errors. The results are reported in Figure 2.

From the first subfigure of Figure 2, with fixed setting of rank(Z^*), we see that our algorithm always achieves the best speed performance under all settings of density $\|S^*\|_0$, and the performance is also stable as $\|S^*\|_0$ varies. LMaFit also performs stably but it is always slower than our algorithm. The results shown in the second subfigure of Figure 2 confirm the stable performance of our algorithm, but the performance gap between LMaFit and our algorithm



(a) $m = n = 1000$, rank(Z^*)=0.01m, $\|S^*\|_0/mn$ between 0.01 and 0.1



(b) $m = n = 1000$, $\|S^*\|_0$=0.01mn, rank$(Z^*)/m$ between 0.01 and 0.1

FIG. 2. *Performance change v.s. problem settings.*

has a slightly decreasing trend.

4.3. Comparison on image problems. In the next part, we do some experiments on real data which are more challenging. We transform the image "slate" [9] and "pencil" [10] into grey-scale images. The underlying matrices of these images are not low-rank, but it is assumed to have a small number of "dominant" singular values, hence it can be approximately considered as a low-rank matrix. We then corrupt a small portion (e.g., 20%) of entries of this matrix which is equivalent to adding a sparse matrix to the original matrix. This type of problem is much more difficult due to the lack of a clear-cut numerical rank, although it is of practical importance since they are perhaps more representative of real-world applications.

Note that high accuracy is not useful for this type of problems since the matrix is not low-rank, instead, visual quality plays more importance in the comparison of results, hence, it is difficult to set a "fair" stopping criterion for the algorithms. Instead, we simply let the algorithms run for a fixed number of iterations with a fixed rank if needed, according to numerical experience. We let Algorithm 1 and LMaFit run 30 steps with fixed rank $k = \text{round}(0.1m)$.

The results for recovering "slate" and "pencil" images from their 20% entries corrupted observations are presented in Figure 3 and 4, respectively. As can be seen from Figure 3, the recovered images by three algorithms are visually comparable in quality. Compared with IALM, our algorithm is around 5 times the speed of IALM in terms of CPU time. Compared LMaFit, costing comparable CPU time, our algorithm obtained slightly better results than that of LMaFit from both visual quality and relative error.

4.4. Comparison on video background extraction problems. The experimental data in previous subsection is based on natural image, but the corruption is done artificially. In this subsection, we consider a more challenging problem-video background extraction problem which aims to separate a video clip into a static background and moving objects. Traditional algorithms for this problem often involve complicated statistical models, which can be difficult to handel. For simplicity, we use videos from security cameras in which the background is relatively static, i.e., the background is roughly the same in every frame, and hence can be viewed approximately as a low-rank matrix. The moving objects such as pedestrians and vehicles usually only occupy a small part of the picture, thus together can be regarded as a sparse matrix. As such, separation of the background and moving objects in a video can be modeled as our matrix separation problem. Our test procedure is similar to that given in [24, 33], and the raw data of video clips were downloaded from `http://perception.i2r.a-star.edu.sg/bk_model/bk_index.html`. Each

[9] `http://img.frbiz.com/pic/z87f6c4-0x0-1/slate_veneers_stone_veneers_cultured_stone_wall_cladding.jpg`

[10] `http://pic44.nipic.com/20140725/2531170_084125187000_2.jpg`

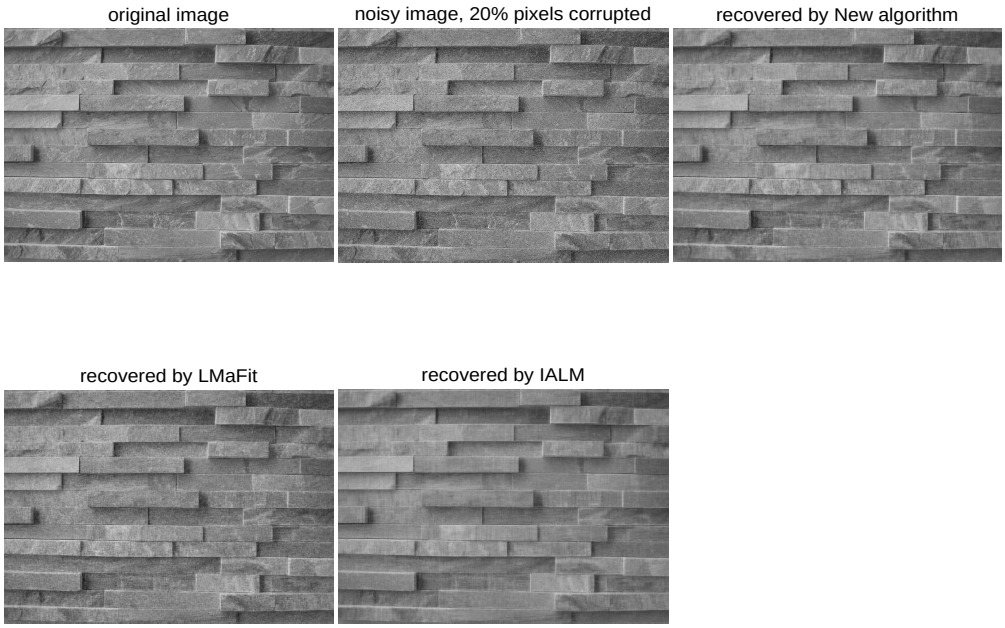


FIG. 3. *Results on "slate": upper left: original image; upper middle: corrupted image; upper right: recovered by our algorithm, err(Z^*)=6.957e-2, CPU=1.801; lower left: recovered by LMaFit, err(Z^*)=8.683e-2, CPU=2.700; lower middle: recovered by IALM, err(Z^*)=7.257e-2, CPU = 7.509.*

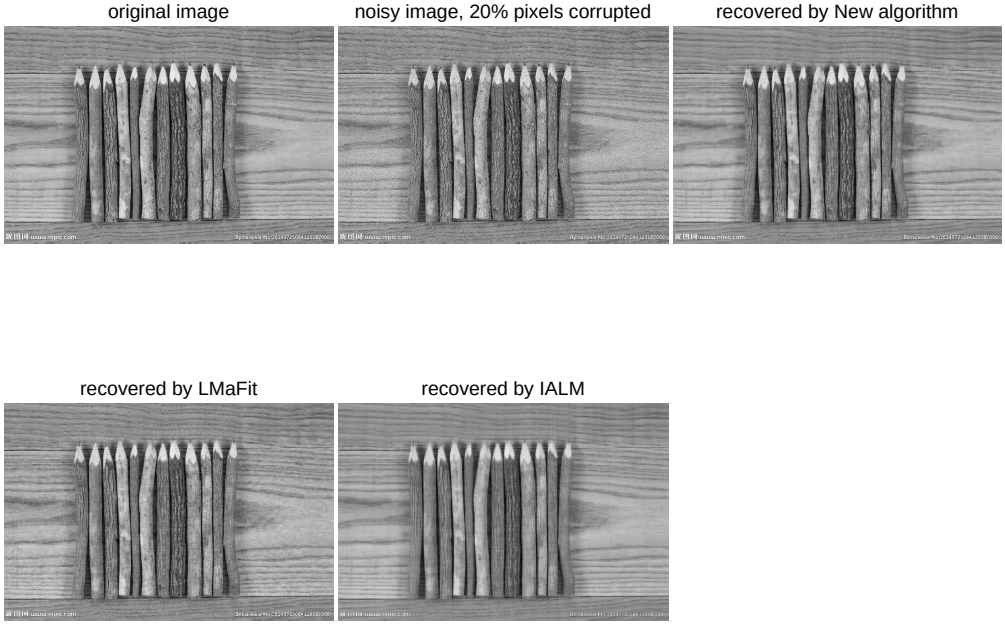


FIG. 4. *Results on "pencil": upper left: original image; upper middle: corrupted image; upper right: recovered by our algorithm, err(Z^*)=9.173e-2, CPU=3.028; lower left: recovered by LMaFit, err(Z^*)=1.060e-1, CPU=3.966; lower middle: recovered by IALM, err(Z^*)=9.926e-2, CPU=11.001.*

video corresponds to a sequence of bitmap files representing the frames of video.

The data matrices are generated by the following procedure: We reshape every frame of video into a long column vector. For color videos, we only take the green channel of every frame as a grayscale image, then reshape it into a column vector. We then collected all the columns into a matrix. As such, the number of rows in a test matrix equals

to the number of pixels and the number of columns equals to the number of frames. For example, a video with 300 frames of 320×240 resolution is converted into a 76800×300 matrix. Since all these video clips have more than 1000 frames, we took a part of each clip with 500 frames or less to reduce the storage and computation required.

We include LMaFit and our algorithm in our experiments while IALM would take excessive amounts of time due to SVD calculations, hence is not included in this part of experiment. The parameters are set as follows: p is fixed to be 1 in our algorithm and LMaFit. Since there is no original solution for this type of problem, we are unable to compute the relative error with original solution. Instead, we simply let two test algorithms stop when relative change between two iterates $err(Z^k, Z^{k+1}) := \|Z^k - Z^{k+1}\|/\|Z^k\|$ is small enough, e.g., we take $err(Z^k, Z^{k+1}) < 1e-5$ in our test. We have to point out that equal relative change does not imply equal visual quality. Hence, the comparison of CPU time is more or less meaningless, we put emphasize on the comparison of results quality, however, the computing times are still listed as references.

We test 4 videos: "Shopping Mall", "Bootstrap", "Hall", "Lobby". The CPU times are reported in Table 1, and the visual results of "Shopping Mall" and "Bootstrap" are shown in Figure 5 and 6 by taking four frames from the results. As seen from Figure 5 and 6, our algorithm can obtain results with better visual quality comparable with that obtained by LMaFit (e.g., observe the first and second rows of Figure 6, the body of a person was not completely recovered by LMaFit while our algorithm can restore the whole body), while it costs slightly less computing time than that of LMaFit according to Table 1.

Video	Resolution	No. frames	New Algorithm	LMaFit
Shopping Mall	320×256	300	11.40	11.59
Bootstrap	160×128	500	13.63	14.44
Hall	176×144	500	17.58	18.85
Lobby	160×128	500	12.24	14.97

TABLE 1
Computing times of video separation problem test.

4.5. Summary on numerical experiments. We have conducted extensive numerical experiments on both synthetic data and real data. The numerical results have confirmed the motivating premise for our approach that avoiding SVD-related calculations can lead to a very efficient algorithm for solving matrix separation/RPCA problem.

With synthetic data, our algorithm has shown advantage in speed performance over some existing algorithms under various problem settings. In the more challenging scenario with real data, the data matrix is approximately low-rank without a clear cut on rank. All algorithms have difficulty to obtain accurate solution, while our algorithm can obtain reasonably good solutions with relatively better visual quality within less computing time.

5. Conclusions. In this paper, we propose a new efficient approach for solving RPCA. Compared with some state-of-the-art algorithms, our new algorithm exhibits the following advantages:

- The framework is easy to implement, and less parameters to be tuned.
- As a BCD method for solving nonconvex nonsmooth optimization problem, the global convergence to local minimizer can be established under mild assumptions.
- It has great potential in solving RPCA to moderate accuracy according to the preliminary numerical results on both synthetic and real test problems.

Acknowledgment. The authors would like to thank Zaiwen Wen from Peking University and Yin Zhang from Rice University for their valuable suggestions on this paper.

[1] E. AMALDI AND V. KANN, *On the approximability of minimizing nonzero variables or unsatisfied relations in linear systems*, Theor. Comput. Sci., 209 (1998), pp. 237–260.

[2] R. BASRI AND D. JACOBS, *Lambertian reflectance and linear subspaces*, IEEE Trans. Pattern Anal. Mach. Intell., 25 (2003), pp. 218–233.

[3] A. BECK AND L. TETRUASHVILI, *On the convergence of block coordinate descent type methods*, SIAM J. Optim., 23 (2013), pp. 2037–2060.

[4] T. BOUWMANS AND E. H. ZAHZAH, *Robust pca via principal component pursuit: A review for a comparative evaluation in video surveillance*, Comput. Vis. Image Und., 122 (2014), pp. 22–34.

[5] S. BOYD, N. PARIKH, E. CHU, B. PELEATO, AND J. ECKSTEIN, *Distributed optimization and statistical learning via the alternating direction method of multipliers*, Found. Trends Mach. Learn., 3 (2011), pp. 1–122.

[6] E. J. CANDÈS, X. LI, Y. MA, AND J. WRIGHT, *Robust principal component analysis?*, J. ACM, 58 (2011), pp. 1– 37.

[7] E. J. CANDÈS AND B. RECHT, *Exact matrix completion via convex optimization*, Foundations of Computational Mathematics, (2009).

[8] E. J. CANDÈS, J. ROMBERG, AND T. TAO, *Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information*, IEEE Trans. Inf. Theory, 52 (2006), pp. 489–509.

[9] E. J. CANDÈS AND T. TAO, *The power of convex relaxation: near-optimal matrix completion*, IEEE Trans. Inform. Theory, 56 (2010), pp. 2053–2080.

[10] V. CHANDRASEKARAN, S. SANGHAVI, P. PARRILO, AND A. WILLSKY, *Sparse and low-rank matrix decompositions*, in 47th Annual Allerton Conference on Communication, Control, and Computing, 2009. Allerton 2009., 2009, pp. 962–967.

[11] V. CHANDRASEKARAN, S. SANGHAVI, P. A. PARRILO, AND A. S. WILLSKY, *Rank-sparsity incoherence for matrix decomposition*, SIAM J. Optim., 21 (2011), pp. 572–596.

[12] C. CHEN, B. HE, Y. YE, AND X. YUAN, *The direct extension of admm for multi-block convex minimization problems is not necessarily convergent*, Math. Program. Series A, 155 (2016), pp. 57–79.

[13] C. CHEN, Y. SHEN, AND Y. YOU, *On the convergence analysis of the alternating direction method of multipliers with three blocks*, Abstr. Appl. Anal., 2013, Article ID 183961 (2013), p. 7 pages.

[14] D. DONOHO, *Compressed sensing*, IEEE Trans. Inform. Theory, 52 (2006), pp. 1289–1306.

[15] L. ELDÉN, *Matrix Methods in Data Mining and Pattern Recognition (Fundamentals of Algorithms)*, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2007.

[16] A. GANESHY, J. WRIGHT, X. LI, E. J. CANDÈS, AND Y. MA, *Dense error correction for low-rank matrices via principal component pursuit*, in IEEE International Symposium on Information Theory Proceedings (ISIT),, June 2010, pp. 1513–1517.

[17] D. HAN AND X. YUAN, *A note on the alternating direction method of multipliers*, J. Optim. Theory Appl., 155 (2012), pp. 227–238.

[18] M. HONG AND Z.-Q. LUO, *On the linear convergence of the alternating direction method of multipliers*. preprint, `http://arxiv.org/abs/1208.3922`, 2012.

[19] M. HONG, Z.-Q. LUO, AND M. RAZAVIYAYN, *Convergence analysis of alternating direction method of multipliers for a family of nonconvex problems*. preprint, `http://arxiv.org/abs/1410.1390`, 2014.

[20] B. JIANG AND S. ZHANG, *Iteration bounds for finding ϵ-stationary points for structured nonconvex optimization*, tech. report, University of Minnesota, 2014.

[21] G. LI AND T. PONG, *Peaceman-rachford splitting for a class of nonconvex optimization problems*, preprint `http://arxiv.org/abs/1507.00887.`, (2015).

[22] G. LI AND T. K. PONG, *Douglas-rachford splitting for nonconvex feasibility problems*, tech. report, University of New South Wales and the Hong Kong Polytechnic University, 2014.

[23] M. LI, D. SUN, AND K.-C. TOH, *A convergent 3-block semi-proximal admm for convex minimization problems with one strongly convex block*, Asia Pac. J. Oper. Res., (2015).

[24] Z. LIN, M. CHEN, L. WU, AND Y. MA, *The augmented lagrange multiplier method for exact recovery of corrupted low-rank matrices*, UIUC Technical Report UILU-ENG-09-2215, submitted to Mathematical Programming, (2009).

[25] Z. LIN, A. GANESH, J. WRIGHT, L. WU, M. CHEN, AND Y. MA, *Fast convex optimization algorithms for exact recovery of a corrupted low-rank matrix*, tech. report, UIUC Technical Report UILU-ENG-09-2214, 2009.

[26] X. LIU, Z. WEN, AND Y. ZHANG, *An efficient gauss-newton algorithm for symmetric low-rank product matrix approximations*, SIAM J. Optim., 25 (2014), pp. 1571–1608.

[27] Z. LIU AND L. VANDENBERGHE, *Interior-point method for nuclear norm approximation with application to system identification*, SIAM J. Matrix Anal. Appl., 31 (2009), pp. 1235–1256.

[28] T. MORITA AND T. KANADE, *A sequential factorization method for recovering shape and motion from image streams*, IEEE Trans. Pattern Anal. Mach. Intell., 19 (1997), pp. 858–867.

[29] C. MU, Y. ZHANG, J. WRIGHT, AND D. GOLDFARB, *Scalable robust matrix recovery: Frank–wolfe meets proximal methods*, SIAM J. Sci. Comput., 38 (2016), pp. 3291–3317.

[30] Y. NESTEROV, *Introductory lectures on convex optimization: A basic course. 87: xviii+236, 2004.*

[31] Y. NESTEROV, *Efficiency of coordinate descent methods on huge-scale optimization problems*, SIAM J. Optim., 22 (2012), pp. 341–362.

[32] B. W. P. RODRIGUEZ, *Fast principal component pursuit via alternating minimization*, in IEEE International Conference on Image Processing, ICIP 2013., 2013.

[33] Y. SHEN, Z. WEN, AND Y. ZHANG, *Augmented lagrangian alternating direction method for matrix separation based on low-rank factorization*, Optim. Methods Softw., 29 (2014), pp. 239–263.

[34] M. TAO AND X. YUAN, *Recovering low-rank and sparse components of matrices from incomplete and noisy observations*, SIAM J. Optim., 21 (2011), pp. 57–81.

[35] C. TOMASI AND T. KANADE, *Shape and motion from image streams under orthography: a factorization method*, Int. J. Comput. Vision, 9 (1992), pp. 137–154.

[36] P. TSENG, *Convergence of a block coordinate descent method for nondifferentiable minimization*, J. Optim. Theory Appl., 109 (2001), pp. 475–494.

[37] Y. WANG, W. YIN, AND J. ZENG, *Global convergence of admm in nonconvex nonsmooth optimization*. Preprint, 2015.

[38] Z. WANG, M. LAI, Z. LU, W. FAN, H. DAVULCU, AND J. YE, *Orthogonal rank-one matrix pursuit for low rank matrix completion*, SIAM J. Sci. Comput., 37 (2015), pp. 488–514.

[39] Z. WEN, D. GOLDFARB, AND K. SCHEINBERG, *Block coordinate descent methods for semidefinite programming*, in Handbook on Semidefinite, Cone and Polynomial Optimization, 2011.

[40] J. WRIGHT, Y. PENG, Y. MA, A. GANESH, AND S. RAO, *Robust principal component analysis: Exact recovery of corrupted low-rank matrices by convex optimization*, in Neural Information Processing Systems, NIPS 2009., 2009.

[41] S. J. WRIGHT, *Coordinate descent algorithms*, Math. Program., 151 (2015), pp. 3–34.

[42] X. YUAN AND J. YANG, *Sparse and low-rank matrix decomposition via alternating direction method*, Pac. J. Optim., 9 (2013), pp. 167–180.

FIG. 5. *Video separation results. From left to right: original, separated results by New algorithm and LMaFit.*

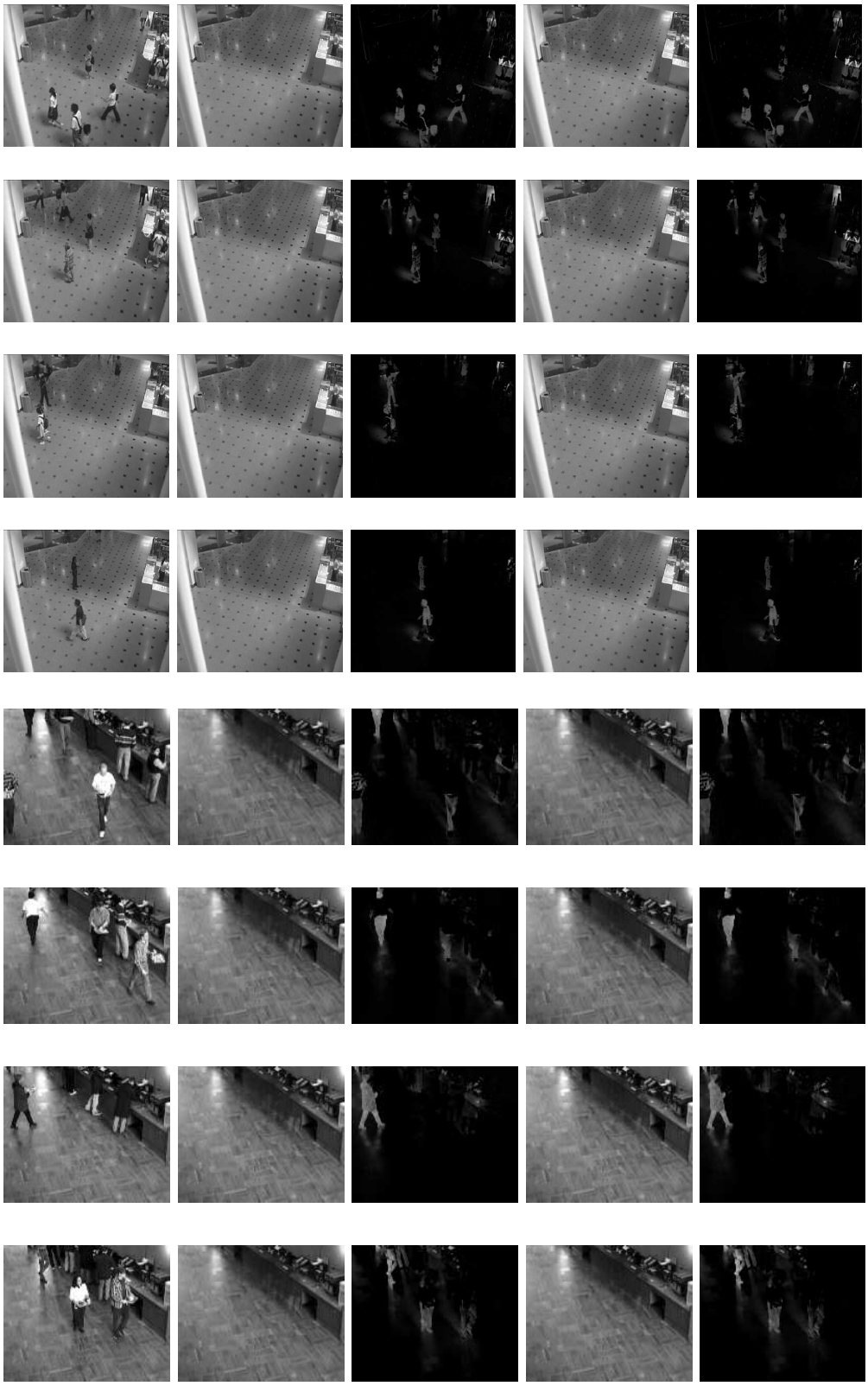


FIG. 6. *Video separation results. From left to right: original, separated results by New algorithm and LMaFit.*