Low Rank and Sparse Decomposition for Image and Video Applications

Nematollah Zarmehi, *Student Member, IEEE*, Arash Amini, *Senior Member, IEEE*, and Farokh Marvasti, *Senior Member, IEEE*

***Abstract*—The matrix decomposing into a sum of low-rank and sparse components has found extensive applications in many areas including video surveillance, computer vision, and medical imaging. In this paper, we propose a new algorithm for recovery of low rank and sparse components of a given matrix. We have also proved the convergence of the proposed algorithm. The simulation results with synthetic and real signals such as image and video signals indicate that the proposed algorithm has a better performance with lower run-time than the conventional methods.**

***Index Terms*—Background modeling, Gradient Projection (GP), low rank recovery, sparse, smoothed ℓ_0 norm, video, video surveillance.**

I. Introduction

THE decompotion of a matrix into low-rank and sparse (sometimes referred to as noise) components is used in various applications such as background extraction in video surveillance [1], [2], removing shadows and specularities from face images [3]–[5], data compression [6], matrix rigidity in computational complexity [7], link prediction in social networks [8], and subspace clustering [9]. A common mathematical model in these applications is to assume a low-rank matrix $\mathbf{L}_o \in \mathbb{R}^{m\times n}$ for the signal of interest which is corrupted by a sparse (noise) matrix $\mathbf{E}_o \in \mathbb{R}^{m\times n}$:

$$\mathbf{Y} = \mathbf{L}_o + \mathbf{E}_o, \quad \mathbf{Y} \in \mathbb{R}^{m\times n}, \tag{1}$$

where $\mathbf{Y}$ is the available data matrix. The recovery problem here refers to extracting $\mathbf{L}_o$ and $\mathbf{E}_o$ from their mixture $\mathbf{Y}$. Oftentimes, the primary goal is to achieve an estimate of the low-rank component $\mathbf{L}_o$; however, the sparse component $\mathbf{E}_o$ might also carry some information such as the moving objects in the video surveillance example. As the noise component is not necessarily Gaussian, the conventional recovery techniques based on Tikhonov regularizers [10] are no longer applicable here. Further, we assume that the rank of $\mathbf{L}_o$ and the sparsity number of $\mathbf{E}_o$ are unknown.

A. Approach

Considering the data model in (1), this paper proposes an algorithm for recovery of low rank and noise components of a given matrix $\mathbf{Y}$ by suggesting an optimization problem that approximates the following optimization problem:

$$\mathcal{P}_0: \quad \begin{array}{ll} \underset{\mathbf{L},\mathbf{E}}{\operatorname{argmin}} & \operatorname{rank}(\mathbf{L}) + \lambda\|\mathbf{E}\|_0, \\ \text{subject to} & \mathbf{Y} = \mathbf{L} + \mathbf{E}, \end{array} \tag{2}$$

where rank of $\mathbf{L}$ is number of non-zero Singular Values (SVs) of $\mathbf{L}$, $\|\mathbf{E}\|_0$ denotes the entry-wise ℓ_0-(pseudo) norm of the noise matrix $\mathbf{E}$. The noise component $\mathbf{E}$ is assumed to be sparse; however, no further information about its support set or the distribution of its non-zero elements is available.

B. Related Works

Principal Component Analysis (PCA) [11] is possibly the most popular technique to extract the low-rank component when the noise component is Gaussian. It is known that the solution to

$$\begin{array}{ll} \min\limits_{\mathbf{L}} & \|\mathbf{E}\|_F, \\ \text{subject to} & \begin{cases} \operatorname{rank}(\mathbf{L}) \le r, \\ \mathbf{Y} = \mathbf{L} + \mathbf{E}, \end{cases} \end{array} \tag{3}$$

is found by the first r principal components of $\mathbf{Y}$; the computational approach is via finding the Singular Value Decomposition (SVD) of $\mathbf{Y}$ and zeroing the $\min(m,n) - r$ right hand SVs. In (3), $\|.\|_F$ denotes the Frobenius norm. PCA works well for high dimensional data that approximately form a low dimensional linear subspace. A typical example is when a low dimensional set of data is contaminated with a high dimensional Gaussian noise. Although the outcome is of high dimensions, the energy is mainly concentrated in the subspace of the original data. PCA gets the optimal solution when the noise components (the entries of $\mathbf{E}$) are i.i.d. Gaussian. Moreover, in this method, r should be known. However, it is also known that PCA fails to recover the low-rank component when just a single entry is substantially perturbed (sparse noise). Indeed, this type of noise is very common in real applications dealing with image and video signals. There are several techniques to make PCA robust against sparse noise [12]–[16]. The Principal Component Pursuit (PCP) introduced in [3] is one of the successful methods that can deal with sparse perturbations with unknown support and arbitrarily large magnitudes. Under certain conditions, the solution to the following convex optimization is guaranteed to yield the original low-rank component:

$$\mathcal{P}_1: \quad \begin{array}{ll} \underset{\mathbf{L},\mathbf{E}}{\operatorname{argmin}} & \|\mathbf{L}\|_* + \lambda\|\mathbf{E}\|_1, \\ \text{subject to} & \mathbf{Y} = \mathbf{L} + \mathbf{E}, \end{array} \tag{4}$$

Manuscript received on December xx, 2017.

The authors are with the Advanced Communication Research Institute (ACRI), Electrical Engineering Department, Sharif University of Technology, Tehran, Iran. (e-mail: zarmehi_n@ee.sharif.edu, aamini@sharif.edu, marvasti@sharif.edu).

where $\|.\|_*$ is the nuclear norm and $\lambda > 0$ is a weighting factor balancing the sparsity and the rank. In this optimization problem, the rank and the ℓ_0-norm of $\mathcal{P}_0$ in (2) are approximated by the nuclear norm and the ℓ_1-norm, respectively. For PCP to be successful, the rank of $\mathbf{L}_o$ and the sparsity level of $\mathbf{E}_o$ need to be sufficiently small; furthermore, $\mathbf{L}_o$ and $\mathbf{E}_o$ should be incoherent. An Augmented Lagrange Multiplier (ALM) algorithm is proposed to solve $\mathcal{P}_1$ [17], [18]. Interestingly, it is shown in [19] that $\mathcal{P}_1$ with suitable λ can recover the low-rank component even when the noise is not so sparse. Although $\mathcal{P}_0$ is a non-convex optimization problem and finding its minimizer is difficult task, it might lead to a suitable result in many instances that $\mathcal{P}_1$ fails. For example, one can simply verify that $\mathcal{P}_1$ fails to recover the original low rank component when

$$\mathbf{Y} = \begin{bmatrix} 0 & 2 & 3 \\ 2 & 0 & 6 \\ 3 & 6 & 0 \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & 2 & 3 \\ 2 & 4 & 6 \\ 3 & 6 & 9 \end{bmatrix}}_{\mathbf{L}_0} + \underbrace{\begin{bmatrix} -1 & 0 & 0 \\ 0 & -4 & 0 \\ 0 & 0 & -9 \end{bmatrix}}_{\mathbf{E}_0}. \tag{5}$$

Apart from convex relaxation approach [20]–[23], non-convex surrogates of low-rank functions are also introduced in the literature [24]–[26]. In [27], an empirical Bayesian approach is proposed for non-convex rank minimization using a variational approximation and marginalization. In this method, the log-determinant function is used to replace the rank function. Moreover, this method assumes a direct Gaussian prior on low-rank matrix. A low-rank matrix factorization model for matrix completion is proposed in [28], [29] whose implementation is called Low-rank Matrix Fitting (LMaFit). The LMaFit algorithm follows a non-linear and non-convex model which is solved using a non-linear successive over-relaxation algorithm. An Alternating Direction Method of Multipliers (ADMM) for matrix separation based on low-rank factorization is proposed in [30] which uses the LMaFit algorithm. This method is limited to the problems where the sparse matrix does not dominate the low-rank matrix in magnitude.

In this paper, unlike the common approach of approximating the rank with the nuclear norm and relaxing the ℓ_0-norm with the ℓ_1-norm, we adopt a smoothing technique which has been recently used for sparse representation [31], [32]. This technique consists of sequentially approximating the ℓ_0-norm with the family of smoothed ℓ_0-norm functions [33]. We explain more details in Section II-B. We also analytically study the convergence of the proposed algorithm and compare it with the state-of-the-art methods under different scenarios and apply it to real applications such as background modeling in video surveillance and removing shadows and specularities from the face images. We shall show by simulations that the proposed algorithm has better performance and lower run-time than the previous works. We believe that our double-smoothing technique as well as the mathematical analysis is rather new.

C. Outline

The paper is organized as follows: In Section II, we provide some preliminaries regarding the employed smoothing technique. In Section III, we explain our proposed algorithm for the low-rank/sparse decomposition. A discussion on the uniqueness of the solution and the proof of convergence of the proposed method are provided in Section IV. The experimental results are presented in Section V. We test the proposed algorithm on synthetic and real data such as face images and video frames. Finally, Section VI concludes the paper.

II. Preliminaries

A. Notations

Throughout the paper, all the scalar variables, column vectors, matrices, and sets will be denoted by italic lower-case, boldface lower-case, boldface upper-case, and black-board-font upper-case letters, respectively. For example, x, $\mathbf{x}$, $\mathbf{X}$, and $\mathbb{X}$ are scalar, vector, matrix, and set, respectively. The elements of vectors and matrices are denoted by subscripts; *i.e.*, x_i is the i-th element of vector $\mathbf{x}$ and $a_{i,j}$ is the element of matrix $\mathbf{A}$ at the intersection of i-th row and j-th column. We also use $[x_i]_{i=1}^n$ and $[a_{i,j}]_{i=1,j=1}^{m,n}$ to denote vector $\mathbf{x}$ of size n and matrix $\mathbf{A}$ of size $m \times n$, respectively. A sequence is shown by $\{x_i\}_{i=1}^{\infty}$. $\mathbf{A}^T$ and $\mathbf{A}^{\curlywedge 2}$ denote the transpose and entry-wise square of matrix $\mathbf{A}$, respectively. For a matrix $\mathbf{A} \in \mathbb{R}^{m\times n}$, $e^{\curlywedge \mathbf{A}}$ indicates a matrix with entries $\{e^{a_{i,j}}\}_{i=1,j=1}^{m,n}$. We shall use $\mathbf{A} \odot \mathbf{B}$ and $\mathbf{A} \oslash \mathbf{B}$ to denote the entry-wise multiplication and division of equi-size matrices $\mathbf{A}$ and $\mathbf{B}$, respectively. We further denote the Frobenius, nuclear, ℓ_∞, ℓ_1, and ℓ_0 norms by $\|.\|_F$, $\|.\|_*$, $\|.\|_\infty$, $\|.\|_1$, and $\|.\|_0$, respectively. Finally, for a matrix $\mathbf{A} \in \mathbb{R}^{m\times n}$, $\mathrm{vec}(\mathbf{A})$ refers to the vector in $\mathbb{R}^{mn}$ obtained by stacking the columns of the matrix $\mathbf{A}$ on bottom of one another and $\mathrm{sum}(\mathbf{A})$ is the sum of all entries of $\mathbf{A}$.

B. The Family of Smoothed ℓ_0-norm Functions

Due to non-convexity and discontinuity, ℓ_0-norm minimization is computationally very challenging. The most popular remedy in problems that deal with ℓ_0-norm minimization is to replace the ℓ_0-norm with ℓ_1-norm. Another approach devised simultaneously in [31] and [32] is to approximate the ℓ_0-norm by a family of smooth functions that tend to the Kronecker delta function in the limit. Because of the smoothness of the approximating functions, the resulting cost function also becomes smooth. In this approach, a zero-mean Gaussian family of functions is used to approximate the ℓ_0-norm of a vector $\mathbf{x} \in \mathbb{R}^n$ as follows:

$$\|\mathbf{x}\|_0 = \sum_{i=1}^{n} \big(1 - \delta[x_i]\big) = n - \lim_{\delta\to 0} \sum_{i=1}^{n} f_\delta(x_i), \tag{6}$$

where $f_\delta(x) = \mathrm{e}^{-\frac{x^2}{2\delta^2}}$ and $\delta[.]$ is the discrete delta function. Besides smoothness, the main property of $f_\delta(\cdot)$ as an approximation of the Kronecker delta function is that

$$f_\delta(x) \approx \begin{cases} 1 & |x| \ll \delta \\ 0 & |x| \gg \delta \end{cases}. \tag{7}$$

Obviously, the Gaussian function family is not the only option here. The family of *triangular* functions, the family

of *truncated hyperbolic* functions, and *homographic* functions of the form $\delta^2/(x^2+\delta^2)$ are also considered for smoothly approximating the ℓ_0-norm [33].

Below, we define a general family $\{f_\delta(\cdot)\}$ of functions that approximate the Kronecker delta function.

Definition 1. Let $f:\mathbb{R}\to[0,1]$ be a smooth function that

1) is analytic, unimodal,
2) $f(x)=1 \Leftrightarrow x=0$, and
3) $\lim_{|x|\to\infty} f(x)=0$,

then, we define the family $\{f_\delta(\cdot)\}$ by $f_\delta(x)=f(x/\delta)$.

III. Proposed Algorithm

A. Main Idea

We recall that our goal is to recover the low-rank matrix $\mathbf{L}_o$ and the noise component $\mathbf{E}_o$ from their mixture $\mathbf{Y}$, by finding the solution to the $\mathcal{P}_0$ problem in (2). We approximate the rank function and the ℓ_0-norm using a family $\{f_\delta(\cdot)\}$ of smooth functions. For the rank of a matrix $\mathbf{L}\in\mathbb{R}^{m\times n}$, we use

$$\begin{aligned}\operatorname{rank}(\mathbf{L}) &= \big\|\sigma(\mathbf{L})\big\|_0 \\ &\approx R_\delta(\mathbf{L}) = h_\delta(\sigma(\mathbf{L})) = q-\sum_{i=1}^{q} f_\delta\big(\sigma_i(\mathbf{L})\big),\end{aligned} \tag{8}$$

where $\sigma(\mathbf{L})=[\sigma_1(\mathbf{L}),\cdots,\sigma_q(\mathbf{L})]^T$ with $\sigma_i(\mathbf{L})$ being the i-th largest SV of $\mathbf{L}$, $h_\delta(\mathbf{a}_{l\times 1})=\sum_{i=1}^{l}\big(1-f_\delta(a_i)\big)$, and $q=\min(m,n)$. Similarly, the sparsity level (ℓ_0-norm) of the matrix $\mathbf{E}\in\mathbb{R}^{m\times n}$ can be approximated as

$$\|\mathbf{E}\|_0 \approx S_\delta(\mathbf{E}) = h_\delta(\operatorname{vec}(\mathbf{E})) = mn-\sum_{i=1}^{mn} f_\delta\big([\operatorname{vec}(\mathbf{E})]_i\big). \tag{9}$$

As (8) and (9) indicate, $R_\delta(\mathbf{L})$ is the smoothed rank function which estimates the number of non-zero SVs of $\mathbf{L}$, while $S_\delta(\mathbf{E})$ is the smoothed ℓ_0-norm function that estimates the number of non-zero entries of $\mathbf{E}$. Now, we approximate the problem (2) as follows:

$$\begin{aligned}(\mathcal{G}):\quad &\underset{\mathbf{L},\mathbf{E}}{\operatorname{argmin}} && G(\mathbf{L},\mathbf{E},\delta)=R_\delta(\mathbf{L})+\lambda S_\delta(\mathbf{E}),\\ &\text{subject to} && \mathbf{Y}=\mathbf{L}+\mathbf{E}.\end{aligned} \tag{10}$$

We use the Gradient Projection (GP) [34] to solve this optimization problem. We suggest the updating and projecting onto the feasible set as follows:

$$\mathcal{G}:\begin{cases}\mathbf{L}\leftarrow \mathbf{L}-\mu_i\nabla_{\mathbf{L}}G(\mathbf{L},\mathbf{E},\delta)\\ \mathbf{E}\leftarrow \mathbf{Y}-\mathbf{L}\\ \mathbf{E}\leftarrow \mathbf{E}-\rho_i\nabla_{\mathbf{E}}G(\mathbf{L},\mathbf{E},\delta)\\ \mathbf{L}\leftarrow \mathbf{Y}-\mathbf{E}\end{cases} \tag{11}$$

where μ_i and ρ_i are the step sizes of the i-th iteration of GP algorithm.

As $\delta\to 0$, $R_\delta(.)$ and $S_\delta(.)$ result in better approximations for the rank and sparsity. But for small values of δ, they have many local minima and the GP algorithm may get trapped in one of the local minima. Hence, we start with a large δ and decrease it at each iteration of the algorithm. The output of the i-th iteration will be used as the initial point for the $(i+1)$-th iteration. This is the Graduated Non-Convexity (GNC) approach for non-convex optimization [35].

Remark 1. One can select a sequence of decreasing δ_i to get a better approximation for the rank and sparsity. We decrease δ during the iterations of the algorithm by $\delta_i=\alpha\delta_{i-1}$, where $\alpha\in[0.5,1)$ is the decreasing factor and δ_1 is set to $g\|\sigma(\hat{\mathbf{L}}_\mathbf{0})\|_\infty$ where $g>1$ is a constant.

Remark 2. In our algorithm, we set the initial point as $(\mathbf{L}_0,\mathbf{E}_0)=(\frac{\lambda}{1+\lambda}\mathbf{Y},\frac{1}{1+\lambda}\mathbf{Y})$ because it satisfies the Karush Kuhn Tucker (KKT) conditions for the optimization problem $\mathcal{G}$ when $\delta\to\infty$. To show this, suppose that the SVD of $\mathbf{L}$ is $\mathbf{L}=\mathbf{U}\operatorname{diag}(\sigma_1,\cdots,\sigma_q)\mathbf{V}^T$. Using the method of Lagrange multipliers, we have the following Lagrangian function

$$\mathcal{L}(\mathbf{L},\mathbf{E},\mathbf{G})=R_\delta(\mathbf{L})+\lambda S_\delta(\mathbf{E})-\operatorname{sum}(\mathbf{G}\odot(\mathbf{Y}-\mathbf{L}-\mathbf{E})), \tag{12}$$

where $\mathbf{G}$ is the Lagrange multiplier. The gradients of the Lagrangian function at $\mathbf{L}$, $\mathbf{E}$, and $\mathbf{G}$ are as follows:

$$\begin{cases}\nabla_{\mathbf{L}}\mathcal{L}(\mathbf{L},\mathbf{E},\mathbf{G})=-\delta^2\mathbf{U}\operatorname{diag}\left(\{f'_\delta(\sigma_i)\}_{i=1}^q\right)\mathbf{V}^T+\delta^2\mathbf{G}\\ \nabla_{\mathbf{E}}\mathcal{L}(\mathbf{L},\mathbf{E},\mathbf{G})=-\lambda\delta^2 f'_\delta(\mathbf{E})+\delta^2\mathbf{G}\\ \nabla_{\mathbf{G}}\mathcal{L}(\mathbf{L},\mathbf{E},\mathbf{G})=\mathbf{Y}-\mathbf{L}-\mathbf{E}\end{cases} \tag{13}$$

Moreover, note that for both homographic and Gaussian smoothed functions we have

$$\lim_{\delta\to\infty}\delta^2 f'_\delta(x)=-\kappa x, \tag{14}$$

where $\kappa=2,1$ for the homographic and Gaussian families, respectively. Therefore, when $\delta\to\infty$, we have

$$\begin{cases}\nabla_{\mathbf{L}}\mathcal{L}(\mathbf{L},\mathbf{E},\mathbf{G})=\kappa\mathbf{L}+\delta^2\mathbf{G}\\ \nabla_{\mathbf{E}}\mathcal{L}(\mathbf{L},\mathbf{E},\mathbf{G})=\lambda\kappa\mathbf{E}+\delta^2\mathbf{G}\\ \nabla_{\mathbf{G}}\mathcal{L}(\mathbf{L},\mathbf{E},\mathbf{G})=\mathbf{Y}-\mathbf{L}-\mathbf{E}\end{cases} \tag{15}$$

One can easily check that for $(\mathbf{L},\mathbf{E},\mathbf{G})=(\frac{\lambda}{1+\lambda}\mathbf{Y},\frac{1}{1+\lambda}\mathbf{Y},-\frac{\lambda\kappa}{\delta^2(1+\lambda)}\mathbf{Y})$, the gradient terms will be zero.

Remark 3. We use $\|\hat{\mathbf{L}}_i-\hat{\mathbf{L}}_{i-1}\|_F\le\epsilon$ as the main stop criterion, where $\hat{\mathbf{L}}_i$ and $\hat{\mathbf{L}}_{i-1}$ are, respectively, the estimated low rank matrices at i-th and $(i-1)$-th iterations, and ϵ is a predetermined parameter. For the inner loop, we set a fixed parameter K as the maximum number of iterations. In Section V, we show that only 3 or 4 iterations is enough for the internal loop.

B. The Proposed Algorithm

Now it is time to present the algorithm. The main algorithm is shown in Algorithm 1. We name it as the Low rank and Sparse Decomposition using Smoothed ℓ_0-Norm (LSD-SN).

One may use some families of smoothed ℓ_0-norm functions in lines 15 and 17 of the Algorithm 1. In case we are using the family of homographic smoothed functions, these two lines would be as follows:

$$\begin{aligned}&\mathbf{L}\leftarrow \mathbf{L}-\mu_i\mathbf{U}\operatorname{diag}\left(\left\{\frac{2\sigma_j\delta_i^4}{(\sigma_j^2+\delta_i^2)^2}\right\}_{j=1}^{q}\right)\mathbf{V}^T,\\ &\mathbf{E}\leftarrow \mathbf{E}-\rho_i\lambda\mathbf{E}\odot\delta^4\oslash(\mathbf{E}^{\curlywedge 2}+\delta^2)^{\curlywedge 2},\end{aligned} \tag{16}$$

Algorithm 1 LSD-SN

1: **input:**
2: Data matrix $\mathbf{Y} \in \mathbb{R}^{m\times n}$
3: Stopping threshold ϵ
4: Decreasing factor α
5: Step size constants γ_μ, γ_ρ
6: Maximum number of iterations of the inner loop K
7: **initialization:**
8: $i \leftarrow 1,\ e \leftarrow \infty,\ \delta_1 \leftarrow 4\|\sigma(\hat{\mathbf{L}}_0)\|_\infty$
9: $\lambda \leftarrow 1/\sqrt{\max(m,n)},\ \hat{\mathbf{L}}_0 \leftarrow \frac{\lambda}{1+\lambda}\mathbf{Y}$
10: **while** $e > \epsilon$ **do**
11: $\mathbf{L} \leftarrow \hat{\mathbf{L}}_{\mathbf{i-1}},\quad \mu_i \leftarrow \gamma_\mu \delta_i^2,\quad \rho_i \leftarrow \gamma_\rho \delta_i^2$
12: $\alpha_\delta \leftarrow \left|f_\delta^{-1}\left(\lambda/n^2\right)\right|,\quad \beta_\delta \leftarrow \left|f_\delta^{-1}\left(1/mn^2\right)\right|$
13: **for** $k = 1 : K$ **do**
14: $[\mathbf{U}, \mathbf{\Sigma}, \mathbf{V}] \leftarrow \text{svd}(\mathbf{L})$
15: $\mathbf{L} \leftarrow \mathbf{L} - \mu_i \nabla_{\mathbf{L}} R_{\delta_i}(\mathbf{L})$
16: $\mathbf{E} \leftarrow \mathbf{Y} - \mathbf{L}$
17: $\mathbf{E} \leftarrow \mathbf{E} - \rho_i \lambda \nabla_{\mathbf{E}} S_{\delta_i}(\mathbf{E})$
18: $\mathbf{E} \leftarrow \mathcal{T}_{\beta_\delta}(\mathbf{E})$
19: $\mathbf{L} \leftarrow \mathbf{Y} - \mathbf{E}$
20: $\mathbf{L} \leftarrow \mathcal{D}_{\alpha_\delta}(\mathbf{L})$
21: **end for**
22: $\hat{\mathbf{E}}_i \leftarrow \mathbf{E}$
23: $\hat{\mathbf{L}}_i \leftarrow \mathbf{Y} - \mathbf{E}$
24: $e \leftarrow \|\hat{\mathbf{L}}_i - \hat{\mathbf{L}}_{i-1}\|_F$
25: $i \leftarrow i + 1$
26: $\delta_i \leftarrow \alpha \delta_{i-1}$
27: **end while**
28: $\hat{\mathbf{L}}_o \leftarrow \hat{\mathbf{L}}_i,\quad \hat{\mathbf{E}}_o \leftarrow \mathbf{Y} - \hat{\mathbf{L}}_o$
29: **return** $\hat{\mathbf{L}}_o, \hat{\mathbf{E}}_o$

and if we are using the family of Gaussian smoothed functions, then, we have

$$\begin{aligned}\mathbf{L} &\leftarrow \mathbf{L} - \mu_i \mathbf{U}\text{diag}\left(\left\{\sigma_j e^{-\frac{\sigma_j^2}{2\delta^2}}\right\}_{j=1}^{q}\right)\mathbf{V}^T,\\ \mathbf{E} &\leftarrow \mathbf{E} - \rho_i \lambda \mathbf{E} \odot e^{\lambda(-\frac{\mathbf{E}^{\lambda 2}}{2\delta^2})}.\end{aligned} \tag{17}$$

To distinguish between these two families, we name the first one as LSD-HSN and the second one as LSD-GSN, where "H" and "G" denote homographic and Gaussian, respectively.

To better approximate the ℓ_0-norm, we threshold the updated matrices $\mathbf{E}$ and $\mathbf{L}$ in lines 18 and 20 of the Algorithm 1, respectively. For a matrix $\mathbf{X} \in \mathbb{R}^{m\times n}$ with SVD $\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$, the operators $\mathcal{D}_\tau(\mathbf{X})$ and $\mathcal{T}_\tau(\mathbf{X})$ are defined as $\mathcal{D}_\tau(\mathbf{X}) = \mathbf{U}\mathcal{T}_\tau(\mathbf{\Sigma})\mathbf{V}^T$, where $\mathcal{T}_\tau(\mathbf{X}) = [\max(x_{i,j} - \tau, 0)]_{i=1,j=1}^{m,n}$. This thresholding expedites the convergence. An example of using and banning this operator is shown in Fig. 1 which shows the SNR versus the number of iterations in the outer loop.

IV. Convergence Analysis

In this section, we provide the proof of convergence of the proposed algorithm. In Section III, several considerations were made to avoid GP algorithm getting trapped in local minima. Hence, it is assumed that the internal loop converges to the global minimum.

First, consider the following definitions and assumptions.

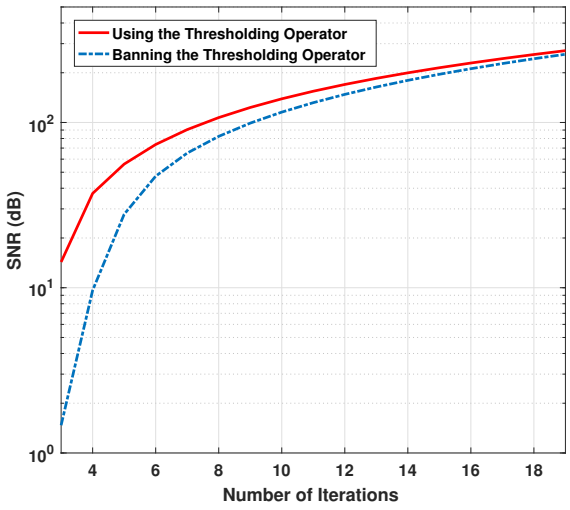


Fig. 1. An example of using and banning the thresholding operator ($n = 500$, $r = 0.05 \times n$, and $k = 0.3 \times n^2$). The number of iterations refer to the outer loop.

Definition 2. For $\epsilon > 0$, define the following set

$$\mathbb{S}_\epsilon \triangleq \left\{(\mathbf{L},\mathbf{E}) \;\middle|\; \begin{array}{rl}(\mathbf{L},\mathbf{E}) \in \underset{\mathbf{A},\mathbf{B}}{\text{argmin}} & \text{rank}(\mathbf{A}) + \lambda\|\mathbf{B}\|_0 \\ \text{subject to} & \|\mathbf{A}+\mathbf{B}-\mathbf{Y}\|_F \le \epsilon\end{array}\right\}.$$

Also, define $\mathbb{S}_0$ as

$$\mathbb{S}_0 \triangleq \left\{(\mathbf{L},\mathbf{E}) \;\middle|\; \begin{array}{rl}(\mathbf{L},\mathbf{E}) \in \underset{\mathbf{A},\mathbf{B}}{\text{argmin}} & \text{rank}(\mathbf{A}) + \lambda\|\mathbf{B}\|_0 \\ \text{subject to} & \|\mathbf{A}+\mathbf{B}-\mathbf{Y}\|_F = 0\end{array}\right\}.$$

Definition 3. For each $\mathbb{S}_\epsilon$, define $v(\mathbb{S}_\epsilon) \triangleq \text{rank}(\mathbf{L}) + \lambda\|\mathbf{E}\|_0$ which is constant for all $(\mathbf{L},\mathbf{E}) \in \mathbb{S}_\epsilon$.

Assumption 1. Assume that $\mathbb{S}_\epsilon$ is bounded by the Frobenius norm. Moreover, assume that $\mathbb{S}_0$ has a unique solution $(\mathbf{L}_0, \mathbf{E}_0)$, i.e., $\mathbb{S}_0 = \{(\mathbf{L}_0, \mathbf{E}_0)\}$.

Remark 4. For $\mathbb{S}_\epsilon$, we have

$$\begin{aligned}&\forall\ \epsilon \ge 0 : v(\mathbb{S}_\epsilon) \le n + \lambda n^2, \quad \text{and}\\ &\forall\ 0 < \epsilon_1 \le \epsilon_2 : v(\mathbb{S}_{\epsilon_1}) \ge v(\mathbb{S}_{\epsilon_2}).\end{aligned} \tag{18}$$

The last inequality is true since the ball defined with ϵ_2 contains the ball defined with ϵ_1; and if the feasible set in an optimization problem is enlarged from a ball of radius r_1 to a ball of radius r_2, the cost associated to the optimal point in the feasible set cannot get worse. Therefore, the limit of $v(\mathbb{S}_\epsilon)$ as $\epsilon \to 0$ exists. Let $\lim_{\epsilon\to 0} v(\mathbb{S}_\epsilon) = v_*$. Notice that $v(.)$ can get finite values. Hence,

$$\begin{aligned}&\exists\ \epsilon_T > 0,\ \forall\ 0 < \epsilon \le \epsilon_T : v(\mathbb{S}_\epsilon) = v_* \quad \Rightarrow\\ &\forall\ \epsilon_1, \epsilon_2,\ \text{such that } 0 < \epsilon_1 \le \epsilon_2 \le \epsilon_T : \mathbb{S}_{\epsilon_1} \subseteq \mathbb{S}_{\epsilon_2},\end{aligned}$$

that means for $0 < \epsilon \le \epsilon_T$, $\mathbb{S}_\epsilon$ are nested sets.

In Assumption 1, we assumed that $\mathbb{S}_\epsilon$ is bounded. In the following Lemma, we prove that it is also closed, thus, it is compact.

Lemma 1. *The set $\mathbb{S}_\epsilon$ defined in Definition 2 is compact.*

Proof. We show that the limit of every convergent sequence in $\mathbb{S}_\epsilon$ is in $\mathbb{S}_\epsilon$. Let $\{(\mathbf{L}_i, \mathbf{E}_i)\}_{i=1}^{\infty}$ be converging sequence (by the Frobenius norm) and let $(\tilde{\mathbf{L}}, \tilde{\mathbf{E}})$ be the limit. Further, let

$$\mathrm{rank}(\tilde{\mathbf{L}}) = r \text{ and } \|\tilde{\mathbf{E}}\|_0 = k.$$

For all $(\mathbf{L}_i, \mathbf{E}_i)$ in the sequence, we have $\|\mathbf{L}_i + \mathbf{E}_i - \mathbf{Y}\| \leq \epsilon$. As $(\tilde{\mathbf{L}}, \tilde{\mathbf{E}})$ is the limit of sequence $\{(\mathbf{L}_i, \mathbf{E}_i)\}_{i=1}^{\infty}$, the same inequality holds for $(\tilde{\mathbf{L}}, \tilde{\mathbf{E}})$, i.e., $\|\tilde{\mathbf{L}}+\tilde{\mathbf{E}}-\mathbf{Y}\| \leq \epsilon$. To prove that $(\tilde{\mathbf{L}}, \tilde{\mathbf{E}}) \in \mathbb{S}_\epsilon$, it is sufficient to show that it is the minimizer of $\mathrm{rank}(.)+\lambda\|.\|_0$. The sparsity number of $\tilde{\mathbf{E}}$ is k. Since $\{\mathbf{E}_i\}_{i=1}^{\infty}$ converges to $\tilde{\mathbf{E}}$ element-wise, there exists an integer i_k such that for all $i \geq i_k$, $\mathbf{E}_i$ is non-zero at the support elements of $\tilde{\mathbf{E}}$. Hence, $\mathbf{E}_i$ has at least k non-zero entries. A similar statement holds for the rank of $\{\mathbf{L}_i\}_{i=1}^{\infty}$, as sorted singular vales are continuous functions of the entries: since $\{\mathbf{L}_i\}_{i=1}^{\infty}$ converges to $\tilde{\mathbf{L}}$, there exists an integer i_r such that, for all $i \geq i_r$, $\mathbf{L}_i$ has at least $r = \mathrm{rank}(\tilde{\mathbf{L}})$ SVs. Therefore,

$$\forall\ i \geq \max(i_k, i_r):\ \mathrm{rank}(\mathbf{L}_i) + \lambda\|\mathbf{E}_i\|_0 \geq r + \lambda k. \quad (19)$$

On one hand, $v(\mathbb{S}_\epsilon) = \mathrm{rank}(\mathbf{L}_i)+\lambda\|\mathbf{E}_i\|_0$ is the minimum cost value among all pairs of $(\mathbf{L}, \mathbf{E})$ that satisfy $\|\mathbf{L} + \mathbf{E} - \mathbf{Y}\| \leq \epsilon$. On the other hand, the pair $(\tilde{\mathbf{L}}, \tilde{\mathbf{E}})$ satisfies the constraint while its cost does not exceeds $v(\mathbb{S}_\epsilon)$. Consequently, we shall have $r + \lambda k = v(\mathbb{S}_\epsilon)$, which implies that $(\tilde{\mathbf{L}}, \tilde{\mathbf{E}}) \in \mathbb{S}_\epsilon$. This completes the proof. ∎

Lemma 2. *The intersection of S_ϵ for $0 < \epsilon \leq \epsilon_T$ consists of a single element; more precisely, $\bigcap_{0<\epsilon\leq\epsilon_T} \mathbb{S}_\epsilon = \mathbb{S}_0$.*

Proof. According to Remark 4 and Lemma 1, for all $0 < \epsilon \leq \epsilon_T$, $\mathbb{S}_\epsilon$ are nonempty nested compact sets. As a result of nested compact sets theorem [36, Theorem 2.36], the intersection is not empty. Let $\hat{\mathbb{S}} = \bigcap_{0<\epsilon\leq\epsilon_T} \mathbb{S}_\epsilon$ and $(\hat{\mathbf{L}}, \hat{\mathbf{E}})$ be an arbitrary pair in $\mathbb{S}_\epsilon$; this implies that $(\hat{\mathbf{L}}, \hat{\mathbf{E}}) \in \mathbb{S}_\epsilon$ or alternatively, $\|\hat{\mathbf{L}} + \hat{\mathbf{E}} - \mathbf{Y}\| \leq \epsilon$, for all $0 < \epsilon \leq \epsilon_T$. Therefore, we conclude that $\|\hat{\mathbf{L}} + \hat{\mathbf{E}} - \mathbf{Y}\|_F = 0$. In addition, according to Remark 4, we have $v_* = v(\hat{\mathbb{S}}) = v(\mathbb{S}_{\epsilon_T}) \leq v(\mathbb{S}_0)$. However, $v(\mathbb{S}_0)$ is the minimum cost value among all pairs of $(\mathbf{L}, \mathbf{E})$ that fulfill $\|\mathbf{L} + \mathbf{E} - \mathbf{Y}\|_F = 0$. This proves that $v_* = v(\mathbb{S}_0)$ and $(\hat{\mathbf{L}}, \hat{\mathbf{E}})$ is indeed the unique element of $\mathbb{S}_0$. In summary:

$$(\hat{\mathbf{L}}, \hat{\mathbf{E}}) = \bigcap_{0<\epsilon\leq\epsilon_T} \mathbb{S}_\epsilon = \mathbb{S}_0$$

∎

Next, we show that the output $(\hat{\mathbf{L}}_\delta, \hat{\mathbf{E}}_\delta)$ of our algorithm at the iteration corresponding to the value δ, belongs to $\mathbb{S}_{\epsilon_\delta}$ for some $\epsilon_\delta > 0$. Furthermore, when $\delta \to 0$ we have that $\epsilon_\delta \to 0$. Since $\{(\hat{\mathbf{L}}_\delta, \hat{\mathbf{E}}_\delta)\}_\delta$ converge to the optimal solution $(\mathbf{L}_0, \mathbf{E}_0)$ as δ tends to 0.

Theorem 1. *Let $(\hat{\mathbf{L}}_\delta, \hat{\mathbf{E}}_\delta)$ be the minimizer of the cost $G(\mathbf{L}, \mathbf{E}, \delta)$ with $\lambda = 1/n^2$, that is subject to the thresholding effects described in lines 18 and 20 of Algorithm 1. Then, for*

$$\alpha_\delta = \left|f_\delta^{-1}\left(\lambda/n^2\right)\right|, \quad \beta_\delta = \left|f_\delta^{-1}\left(1/n^3\right)\right|, \quad \epsilon_\delta = \alpha_\delta n+\beta_\delta n^2$$

we know that $(\hat{\mathbf{L}}_\delta, \hat{\mathbf{E}}_\delta) \in \mathbb{S}_{\epsilon_\delta}$. Moreover, $\lim_{\delta\to 0} \epsilon_\delta = 0$.

Proof. Let $(\mathbf{L}_\delta, \mathbf{E}_\delta)$ be the minimizer of $R_\delta(.) + \lambda S_\delta(.)$ and $(\mathbf{L}_0, \mathbf{E}_0)$ be the unique solution of $\mathbb{S}_0$ with $\mathrm{rank}(\mathbf{L}_0) = r_0$ and $\|\mathbf{E}_0\|_0 = k_0$; Therefore,

$$R_\delta(\mathbf{L}_\delta) + \lambda S_\delta(\mathbf{E}_\delta) \leq R_\delta(\mathbf{L}_0) + \lambda S_\delta(\mathbf{E}_0) \leq r_0 + \lambda k_0, \quad (20)$$

where the last inequality follows from the fact that $R_\delta(\mathbf{L}) \leq \mathrm{rank}(\mathbf{L})$ and $S_\delta(\mathbf{E}) \leq \mathbf{E}$, for all $\mathbf{L}$ and $\mathbf{E}$.

Assume that the SVs of $\mathbf{L}_\delta$ and the entries of $\mathbf{E}_\delta$ are sorted as $\hat{\sigma}_1 \geq \cdots \geq \hat{\sigma}_n$ and $\tilde{e}_1 \leq \cdots \leq \tilde{e}_{n^2}$, respectively. Define $\mathbb{I}_{\alpha_\delta} = \left\{i \in \{1, \cdots, n\} \mid \hat{\sigma}_i > \alpha_\delta\right\}$ and $\mathbb{I}_{\beta_\delta} = \left\{i \in \{1, \cdots, n^2\} \mid \tilde{e}_i > \beta_\delta\right\}$. This results in

$$\begin{aligned}
R_\delta(\mathbf{L}_\delta) + \lambda S_\delta(\mathbf{E}_\delta) &= n - \sum_{i=1}^{n} f_\delta(\hat{\sigma}_i) + \lambda n^2 - \lambda \sum_{i=1}^{n^2} f_\delta(\tilde{e}_i) \\
&= n - \underbrace{\sum_{i\in\mathbb{I}_{\alpha_\delta}} \underbrace{f_\delta(\hat{\sigma}_i)}_{<\frac{\lambda}{n^2}}}_{<n\frac{\lambda}{n^2}=\frac{\lambda}{n}} - \underbrace{\sum_{i\notin\mathbb{I}_{\alpha_\delta}} \underbrace{f_\delta(\hat{\sigma}_i)}_{\leq 1}}_{\leq n-|\mathbb{I}_{\alpha_\delta}|} \\
&\quad + \lambda n^2 - \lambda \underbrace{\sum_{i\in\mathbb{I}_{\beta_\delta}} \underbrace{f_\delta(\tilde{e}_i)}_{<\frac{1}{n^3}}}_{<n^2\frac{1}{n^3}=\frac{1}{n}} - \lambda \underbrace{\sum_{i\notin\mathbb{I}_{\beta_\delta}} \underbrace{f_\delta(\tilde{e}_i)}_{\leq 1}}_{\leq n^2-|\mathbb{I}_{\beta_\delta}|} \\
&> |\mathbb{I}_{\alpha_\delta}| + \lambda|\mathbb{I}_{\beta_\delta}| - \frac{2\lambda}{n}.
\end{aligned}$$

Since $R_\delta(\mathbf{L}_\delta) + \lambda S_\delta(\mathbf{E}_\delta) \leq r_0 + \lambda k_0$, we have

$$|\mathbb{I}_{\alpha_\delta}| + \lambda|\mathbb{I}_{\beta_\delta}| < r_0 + \lambda k_0 + \frac{2\lambda}{n},$$

or $(|\mathbb{I}_{\alpha_\delta}|-r_0)+\lambda(|\mathbb{I}_{\beta_\delta}|-k_0) < \frac{2}{n^3}$ (we recall that $\lambda = 1/n^2$). Consequently,

$$\underbrace{n^2\left(|\mathbb{I}_{\alpha_\delta}| - r_0 + \lambda(|\mathbb{I}_{\beta_\delta}| - k_0)\right)}_{A} < \frac{2}{n}.$$

Note that both $|\mathbb{I}_{\alpha_\delta}| - r_0$ and $|\mathbb{I}_{\beta_\delta}| - k_0$ are integers. This reveals that A is also integer. Since $\frac{2}{n} < 1$ for $n > 2$, we conclude that $A \leq 0$, or

$$|\mathbb{I}_{\alpha_\delta}| + \lambda|\mathbb{I}_{\beta_\delta}| \leq r_0 + \lambda k_0.$$

If we threshold the SVs of $\mathbf{L}_\delta$ and the entries of $\mathbf{E}_\delta$ by α_δ and β_δ, respectively, we get new matrices $\hat{\mathbf{L}}_\delta$ and $\hat{\mathbf{E}}_\delta$ for which we have

$$\|\sigma(\hat{\mathbf{L}}_\delta)\|_0 + \lambda\|\hat{\mathbf{E}}_\delta\|_0 = |\mathbb{I}_{\alpha_\delta}| + \lambda|\mathbb{I}_{\beta_\delta}| \leq r_0 + \lambda k_0. \quad (21)$$

Because of the thresholding operators, the sum of $\hat{\mathbf{L}}_\delta$ and $\hat{\mathbf{E}}_\delta$ no longer equal to $\mathbf{Y}$. Hence, we have an error:

$$\mathbf{U}_\delta = \hat{\mathbf{L}}_\delta + \hat{\mathbf{E}}_\delta - \mathbf{Y}.$$

In the thresholding stage, we threshold at most n SVs of $\mathbf{L}_\delta$ and n^2 entries of $\mathbf{E}_\delta$. Therefore, $\|\mathbf{U}_\delta\|_F \leq \alpha_\delta n+\beta_\delta n^2 = \epsilon_\delta$. If $\epsilon_\delta \leq \epsilon_T$, (21) demonstrates that the cost of $(\hat{\mathbf{L}}_\delta, \hat{\mathbf{E}}_\delta)$ is indeed optimal and equal to v_*, hence, $(\hat{\mathbf{L}}_\delta, \hat{\mathbf{E}}_\delta) \in \mathbb{S}_{\epsilon_\delta}$. It should be mentioned that based on Definition 2, for any $\zeta > 0$, we can choose sufficiently small δ such that

$$\alpha_\delta = \left|f_\delta^{-1}\left(\lambda/n^2\right)\right| < \zeta \implies \lim_{\delta\to 0} \alpha_\delta = 0 \quad \text{and}$$
$$\beta_\delta = \left|f_\delta^{-1}\left(1/n^3\right)\right| < \zeta \implies \lim_{\delta\to 0} \beta_\delta = 0.$$

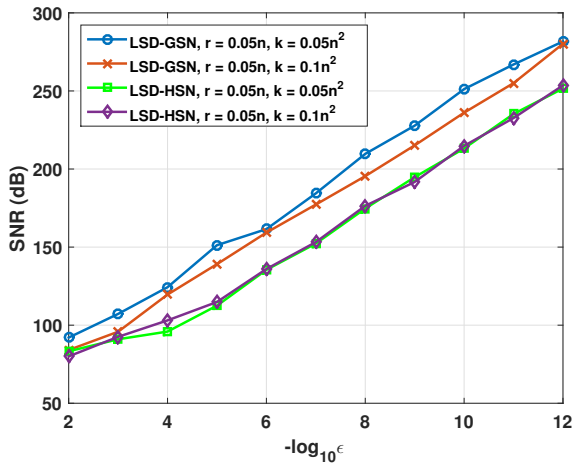


Fig. 2. Performance of the proposed algorithm versus stopping threshold ϵ.

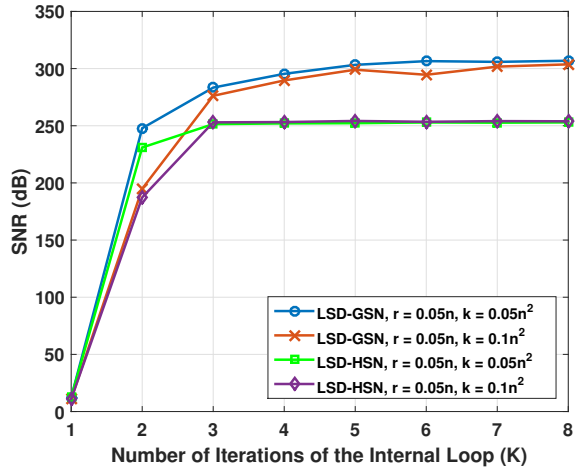


Fig. 3. Performance of the proposed algorithm versus number of iterations of the internal loop K. ($\epsilon = 10^{-12}, \alpha = 0.8$)

This implies that $\epsilon_\delta \to 0$ as $\delta \to 0$, as well as $\epsilon_\delta \leq \epsilon_T$ for sufficiently small δ values. ■

V. Numerical Experiments and Applications

In this section, we present numerical experiments and compare the proposed algorithm empirically with some well known algorithms. In all numerical experiments, we use both homographic and Gaussian smoothed functions. As mentioned before, we use LSD-HSN for the former method and use LSD-GSN for the latter one. All simulations are done by MATLAB R2015a on Intel(R) Core(TM) i7-5960X @ 3GHz with 32GB-RAM. We use $\mathrm{SNR}(\mathbf{L}_o, \hat{\mathbf{L}}_o) = 20\log_{10}\left(\|\mathbf{L}_o\|_F / \|\mathbf{L}_o - \hat{\mathbf{L}}_o\|_F\right)$ in dB as the evaluation criterion.

A. Parameter Effects

First, we investigate the effect of the parameters ϵ, α, and K on the performance of the proposed algorithm. We produce r-rank square matrices of dimensions $m = n = 100$ as the product of two $n \times r$ matrices, i.e., $\mathbf{L}_o = \mathbf{A}\mathbf{B}^T$ where both $\mathbf{A}$ and $\mathbf{B}$ are independently sampled from a $\mathcal{N}(0, 1/n)$ distribution.

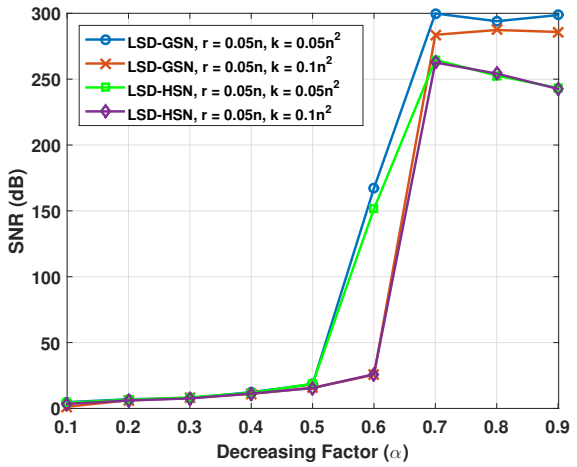


Fig. 4. Performance of the proposed algorithm versus decreasing factor α. ($\epsilon = 10^{-12}, K = 3$)

The noise component $\mathbf{E}_o$ has a support size of k generated uniformly at random whose values are independently selected from the set $\{-1, +1\}$ with equal probabilities. Two random problems are considered in the following experiments with different values of rank and sparsity.

1) Effect of ϵ: To investigate the effect of ϵ, we vary ϵ from 10^{-2} to 10^{-16} and the recovered SNR is shown in Fig. 2. We fix other parameters as $\alpha = 0.8$ and $K = 3$. As expected, the proposed algorithm could better recover the low rank matrix by decreasing ϵ.

2) Effect of K: To investigate the number of iterations of the internal loop on the performance of the proposed algorithm, we change K from 1 to 8 and fix other parameters as $\epsilon = 10^{-12}$ and $\alpha = 0.8$. The results are shown in Fig. 3. It can be seen that only 3 or 4 iterations is enough for the internal loop.

3) Effect of α: Finally, the effect of decreasing factor α is investigated. For this purpose, we set $\epsilon = 10^{-12}$ and $K = 3$ and change α from 0.1 to 0.9. Fig. 4 shows the results. It can be observed that the performance gets better as α is closer to 1.

In the following, we have taken into consideration the effects of the above parameters to use of the proposed algorithm.

B. Exact Recovery

This subsection presents the numerical results to demonstrate exact recovery of the proposed algorithm. Similar to [3], we produce r-rank square matrices of dimensions $n = 500, 1000, \cdots, 3000$ as described in Subsection V-A. In this subsection, we consider $r = 0.05 \times n$. The noise component is also generated as described in Subsection V-A. Tables I and II show the results for $k = 0.05 \times n^2$ and $k = 0.1 \times n^2$, respectively. As it can be seen, the proposed algorithm could exactly recover the low rank and sparse components, however, because of finite precision we could not get $\mathrm{SNR} = \infty$ dB.

C. Comparison

In this subsection, we compare the proposed algorithm with the Inexact Augmented Lagrange Multiplier (IALM) [17],

TABLE I
EXACT RECOVERY OF LOW RANK AND NOISE COMPONENTS FOR RANDOM PROBLEMS. ($r = 0.05 \times n$, $k = 0.05 \times n^2$)

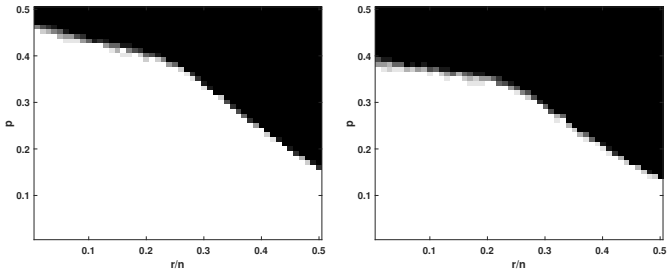
Dimension	**Rank**	**Sparsity**	**SNR (dB)**	
n	r	k	**LSD-HSN**	**LSD-GSN**
500	25	12,500	267.9	262.8
1000	50	50,000	249.5	274.0
1500	75	112,500	255.2	280.5
2000	100	200,000	256.1	251.5
2500	125	312,500	256.8	283.9
3000	150	450,000	256.9	284.6

TABLE II
EXACT RECOVERY OF LOW RANK AND NOISE COMPONENTS FOR RANDOM PROBLEMS. ($r = 0.05 \times n$, $k = 0.1 \times n^2$)

Dimension	**Rank**	**Sparsity**	**SNR (dB)**	
n	r	k	**LSD-HSN**	**LSD-GSN**
500	25	25,000	254.6	274.7
1000	50	100,000	264.5	291.2
1500	75	225,000	265.2	262.2
2000	100	400,000	265.3	264.3
2500	125	625,000	265.0	265.4
3000	150	900,000	265.5	266.0

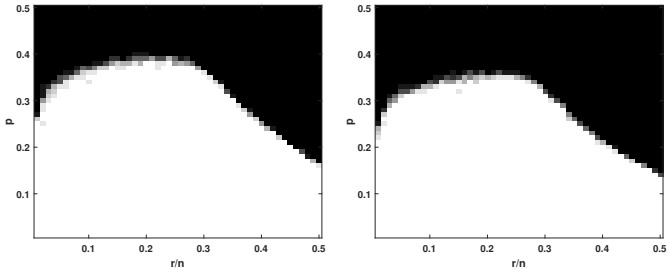
LSSD [37], SpaRCS [38], LRSD-TNNSR [39], and LMaFit [30]. The low rank and sparse matrices are generated in the same way as explained in Subsection V-A. The main drawback of the LMaFit algorithm is its limitation to the problems where the sparse matrix does not dominate the low-rank matrix in magnitude; for this reason, we compare the proposed method with the LMaFit algorithm in different setup. Tables III, IV, and V show the results of comparison with the IALM, LSSD, SpaRCS, and LRSD-TNNSR methods for $r = 0.1 \times n^2$ and $k = 0.2 \times n^2, 0.3 \times n^2$, and $0.4 \times n^2$, respectively. It is clear that the proposed algorithm with both the homographic and Gaussian smoothed functions outperform the other methods. The IALM, LSSD, and LRSD-TNNSR methods have a high computation time in terms of run-time. According to the results of Tables IV and V, the IALM method fails to recover the low rank and sparse components when the sparsity number of the noise is high. Moreover, we can see that the LSSD fails to recover the low rank and sparse components in random problem setup which may be due to the fact that here, the noise component has no structure (random structure) while the LSSD method incorporates the structure sparsity of the noise component. Another important point is that the SpaRCS method needs an approximation of the rank and sparsity of the low rank and sparse components which seems to be unfair comparing to the other methods.

In the following, we compare the proposed algorithm with the LMaFit algorithm [30] which is an ADMM based algorithm for matrix separation based on low rank factorization. The MATLAB code of the LMaFit algorithm is downloaded from the LMaFit website [29] and the all parameters are set to their default values. As mentioned before, the LMaFit algorithm fails to recover the low rank and sparse components when the sparse component dominates the low rank component in magnitude. Therefore, we do the comparison in a distinct setup where the non-zero values of sparse component $\mathbf{E}_o$ are chosen from the set $\{-\sigma, +\sigma\}$ with equal probabilities. We set $r = 0.05 \times n$, $k = 0.1 \times n^2$, and $\sigma = 0.1$ for this comparison. The results are shown in Table VI. It is obvious that the LMaFit algorithm fails to recover the low rank and sparse components even for $\sigma = 0.1$.



(a) LSD-HSN (b) LSD-GSN

Fig. 5. Phase transition of the proposed algorithm with random noise.



(a) LSD-HSN (b) LSD-GSN

Fig. 6. Phase transition of the proposed algorithm with coherent noise.

D. Phase Transition Between Rank and Sparsity

This subsection is devoted to empirically investigate the performance of the proposed algorithm with different values of rank and sparsity. The low rank matrices of size $m = n = 200$ are generated by the same approach explained in Subsection V-A. For obtaining the phase transition between rank and sparsity, each simulation generates 20 random problems and a solution declared successful if the recovered $\hat{\mathbf{L}}_o$ satisfies $\mathrm{SNR}(\mathbf{L}_o, \hat{\mathbf{L}}_o) \geq 60\ dB$. In the phase transition plot, the gray color indicates the recovery rate. Moreover, the white and black colors represent 100% success and failure areas, respectively, while the gray areas show a success probability between 0% to 100%. It should be noted that while, we plot the results in terms of rank or sparsity level, these values were not revealed to the algorithm (no prior information is used in the decomposition procedure). We consider two experiments for this purpose.

1) Random Noise: In this experiment, the noise matrix $\mathbf{E}_o$ has a support that obeying a Bernoulli distribution with random signs. Each entry of noise matrix takes on values of $\{0, -1, +1\}$ with probabilities of $\{1 - p, p/2, p/2\}$. The phase transition of this experiment is depicted in Fig. 5. As expected in most phase-transition curves, the gray area shrinks at high dimensions (asymptotic sharp phase transition). One

TABLE III
COMPARISON WITH THE IALM [17], LSSD [37], AND SPARCS [38]. ($r = 0.1 \times n$, $k = 0.2 \times n^2$)

Dimension	**SNR (dB)**						**Time (s)**					
n	**LSD-HSN**	**LSD-GSN**	**IALM [17]**	**LSSD [37]**	**SpaRCS [38]**	**LRSD-TNNSR [39]**	**LSD-HSN**	**LSD-GSN**	**IALM [17]**	**LSSD [37]**	**SpaRCS [38]**	**LRSD-TNNSR [39]**
500	259.1	259.2	60.8	-19.0	42.2	86.3	56.4	53.5	162.4	77.2	3.2	69.0
1000	258.6	293.5	233.3	-23.9	36.2	87.0	131.0	133.1	371.9	352.4	33.5	153.4
1500	258.4	291.9	247.8	-21.1	39.3	86.7	283.2	284.2	735.7	969.1	97.7	390.3
2000	258.2	290.6	247.5	-24.0	38.2	89.6	498.2	464.8	906.4	1806.6	210.8	921.3
2500	258.1	289.6	247.2	-24.8	41.0	89.2	734.6	670.0	1331.1	7691.2	416.6	1747.6
3000	258.4	288.9	247.6	-25.4	40.4	88.5	1037.6	929.4	1746.1	4960.4	836.5	3743.8

TABLE IV
COMPARISON WITH THE IALM [17], LSSD [37], AND SPARCS [38]. ($r = 0.1 \times n$, $k = 0.3 \times n^2$)

Dimension	**SNR (dB)**						**Time (s)**					
n	**LSD-HSN**	**LSD-GSN**	**IALM [17]**	**LSSD [37]**	**SpaRCS [38]**	**LRSD-TNNSR [39]**	**LSD-HSN**	**LSD-GSN**	**IALM [17]**	**LSSD [37]**	**SpaRCS [38]**	**LRSD-TNNSR [39]**
500	205.69	276.5	12.4	-6.6	36.0	51.2	61.1	61.1	61.7	134.2	3.9	74.9
1000	247.8	284.3	12.8	-6.1	36.1	53.5	245.2	251.1	250.2	381.9	35.5	162.5
1500	249.0	283.0	12.8	-6.5	36.0	55.9	444.2	409.3	401.3	897.1	109.8	412.4
2000	249.2	286.1	12.8	-6.8	36.5	55.4	660.5	627.4	654.1	1657.3	241.8	960.5
2500	249.6	287.3	12.8	-6.0	37.6	56.1	913.4	837.9	876.6	2772.2	536.9	1887.2
3000	249.2	287.9	12.8	-6.4	38.2	55.9	1194.7	1081.7	1122.8	4282.3	875.3	3947.4

TABLE V
COMPARISON WITH THE IALM [17], LSSD [37], AND SPARCS [38]. ($r = 0.1 \times n$, $k = 0.4 \times n^2$)

Dimension	**SNR (dB)**						**Time (s)**					
n	**LSD-HSN**	**LSD-GSN**	**IALM [17]**	**LSSD [37]**	**SpaRCS [38]**	**LRSD-TNNSR [39]**	**LSD-HSN**	**LSD-GSN**	**IALM [17]**	**LSSD [37]**	**SpaRCS [38]**	**LRSD-TNNSR [39]**
500	221.8	255.8	3.5	-14.9	19.7	48.8	168.5	168.0	227.2	99.3	4.1	84.5
1000	221.8	259.3	3.6	-14.1	21.8	48.7	68.2	62.5	83.8	420.4	37.4	182.6
1500	221.9	257.4	3.8	-14.0	23.1	49.1	286.7	271.2	367.3	941.3	116.5	510.9
2000	221.8	256.0	3.8	-13.5	23.4	48.7	1101.3	1068.6	1466.0	1828.8	258.7	1006.8
2500	223.3	255.0	3.8	-13.3	23.7	48.4	1647.5	1589.3	2164.9	4216.0	543.3	2077.5
3000	224.7	254.1	3.9	-12.0	23.0	48.1	2292.5	2144.6	3033.1	6019.1	922.8	4152.4

TABLE VI
SNR COMPARISON BETWEEN THE PROPOSED ALGORITHM AND LMAFIT ALGORITHM [30]. ($r = 0.05 \times n$, $k = 0.1 \times n^2$)

Dimension (n)	**LSD-HSN**	**LSD-GSN**	**LMaFit [30]**
500	303.8	311.1	-2.76
1000	303.2	309.9	-5.7
1500	301.1	308.7	-7.5
2000	301.6	307.8	-8.7
2500	301.6	307.4	-9.6
3000	301.7	306.6	-10.4

can see that the phase transition of the proposed algorithm for both homographic and Gaussian smoothed functions are almost the same. Furthermore, for all pairs of $(p, r/n)$ less than $(0.26, 0.35)$, the low rank matrix could be recovered successfully.

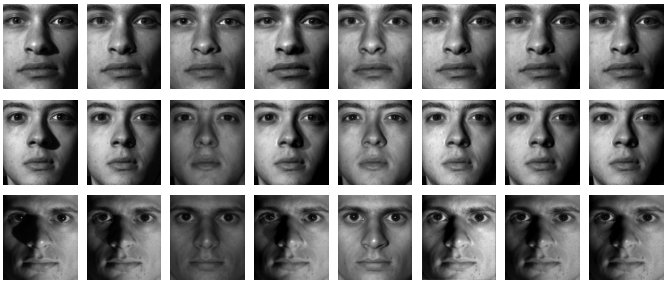
2) Coherent Noise: Unlike the previous experiment, in this experiment, we assume that the noise components are coherent with the low rank matrix. First, a random binary mask $\mathbf{M}$ is generated that takes on values of 0 and 1 with probabilities $1-p$ and p, respectively. Then, the noise matrix $\mathbf{E}_o$ is generated as $\mathbf{E}_o = \mathbf{M} \odot \mathrm{sgn}(\mathbf{L}_o)$. Fig. 6 shows the phase transition of this experiment. Although this experiment seems to be hard, the successful area (white-colored area) of the phase transition plot is not so tight.

E. Applications

There are many applications in which one needs to recover a low rank matrix from the corrupted observations. Here, we consider two applications, removing shadows and specularities from face images and background modeling, and compare our method with ALM [13], LSSD [37], SpaRCS [38], LRSD-TNNSR [39], and RASL [40].

1) Removing Shadows and Specularities from Face Images: If there are enough images from a face with different luminances, shadows, and specularities, one can use the low rank and noise recovery method to get rid of such noises from the faces because this dataset has also low dimension [3]–[5]. We have selected three face images from the Extended Yale Face Database B (B+) dataset [41]. The results are shown in Fig. 7. The run-times of the LSD-HSN, LSD-GSN, ALM, LSSD, SpaRCS, LRSD-TNNSR, and RASL algorithms are 3.7, 3.6, 11.1, 65.9, 2.0, 5.1, and 12.3 seconds, respectively.

2) Background Modeling from Surveillance Video: In video surveillance applications, one needs to detect any activity or change in the successive frames [1], [3]. Each group of picture (GOP) of video has a main background. Therefore,



(a) (b) (c) (d) (e) (f) (g) (h)

Fig. 7. Examples of removing shadows and specularities from face images using low rank and sparse decomposition method. (a) Original images. (b) Results of ALM. (c) Results of LSSD. (d) Results of SpaRCS. (e) Results of LRSD-TNNSR. (f) Results of RASL. (g) Results of LSD-HSN. (h) Results of LSD-GSN.

we get a low rank matrix if vectorize all frames of a GOP and put them in a matrix. In this way, any activity on the foreground can be modeled as a sparse noise that can be separated from the main background. To show this, we used the proposed method, ALM, LSSD, and SpaRCS, LRSD-TNNSR, and RASL methods to model the background of two selected video sequences, Hall of a business building [42] and a nominal sequence introduced in [43]. Fig. 8a shows three frames from the original Hall video. Figs. 8b, 8d, 8f, 8h, 8j, 8l, and 8n show the low rank component (background) and Figs. 8c, 8e, 8g, 8i, 8k, 8m, and 8o show the sparse component (moving object), respectively. The results for the second video sequence is shown in Fig. 9. It should be mentioned that in this application, the primary low rank and the noise components are not purely low rank and sparse, respectively. Although we can not see considerable differences between the results, the proposed algorithm has lower run-time. The run-times of the LSD-HSN, LSD-GSN, ALM, LSSD, SpaRCS, LRSD-TNNSR, and RASL methods are 135.3, 128.5, 252.4, 2786.2, 199.6, 155.9, and 1442.7 seconds, respectively.

VI. Conclusion

A new algorithm for recovery of low rank and noise components of a given data matrix was proposed. Unlike the state-of-the-art methods that have approximated the ℓ_0-norm of the rank and sparsity with nuclear norm and ℓ_1-norm, here, we approximated the ℓ_0-norm by families of smoothed ℓ_0-norm functions. The GP method is used for solving the minimization problem. Many considerations were made to avoid GP algorithm getting trapped in the local minima points. The convergence of the proposed algorithm was analytically provided. The proposed algorithm was compared with the Augmented Lagrange Multiplier (ALM) based method and LMaFit method. For the synthetic signals, the simulation results indicated that the proposed algorithm could exactly recover the low rank and noise matrices at much less complexity in terms of run-time; in some cases, the ALM-based method and LMaFit method failed to recover the low rank and sparse components. We have also applied the proposed algorithm for two real applications: background modeling for video surveillance and removing shadows and specularities from face images. We have also compared all results with those the ALM method. In general, the proposed algorithm had better performance at lower run-time.

Acknowledgment

The author would like to thank Dr. Pedram Pad for his help of reviewing the proof of convergence.

References

[1] K. Min, Z. Zhang, J. Wright, and Y. Ma, "Decomposing background topics from keywords by principal component pursuit," in *Proceedings of the 19th ACM International Conference on Information and Knowledge Management*. New York, NY, USA: ACM, 2010, pp. 269–278.

[2] S. Javed, A. Mahmood, T. Bouwmans, and S. K. Jung, "Spatiotemporal low-rank modeling for complex scene background initialization," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. PP, no. 99, pp. 1–1, 2016.

[3] E. J. Candès, X. Li, Y. Ma, and J. Wright, "Robust principal component analysis?" *J. ACM*, vol. 58, no. 3, pp. 11:1–11:37, Jun. 2011.

[4] J. Wright, G. Arvind, R. Shankar, P. Yigang, and Y. Ma, "Robust principal component analysis: Exact recovery of corrupted low-rank matrices via convex optimization," in *Advances in Neural Information Processing Systems 22*. Curran Associates, Inc., 2009, pp. 2080–2088.

[5] N. S. Aybat, D. Goldfarb, and S. Ma, "Efficient algorithms for robust and stable principal component pursuit problems," *Computational Optimization and Applications*, vol. 58, no. 1, pp. 1–29, 2014.

[6] J. Hou, L. P. Chau, N. Magnenat-Thalmann, and Y. He, "Sparse low-rank matrix approximation for data compression," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 27, no. 5, pp. 1043–1054, May 2017.

[7] L. G. Valiant, *Graph-theoretic arguments in low-level complexity*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1977, pp. 162–176.

[8] K. Min, Z. Zhang, J. Wright, and Y. Ma, "Estimation of simultaneously sparse and low rank matrices," in *29th International conference on machine learning*, 2012, pp. 1351–1358.

[9] L. Han and X.-L. Liu, "Convex relaxation algorithm for a structured simultaneous low-rank and sparse recovery problem," *Journal of the Operations Research Society of China*, vol. 3, no. 3, pp. 363–379, 2015.

[10] A. N. Tikhonov, "Solution of incorrectly formulated problems and the regularization method," *Soviet Math. Dokl.*, vol. 4, pp. 1035–1038, 1963.

[11] I. Jolli, *Principal Component Analysis*. Springer-Verlag, 1986.

[12] P. Huber, *Robust Statistics*. Wiley, New York, 1981.

[13] F. D. la Torre and M. J. Black, "A framework for robust subspace learning," *International Journal of Computer Vision*, vol. 54, no. 1, pp. 117–142, 2003.

[14] R. Gnanadesikan and J. R. Kettenring, "Robust estimates, residuals, and outlier detection with multiresponse data," *Biometrics*, vol. 28, no. 1, pp. 81–124, 1972.

[15] Q. Ke and T. Kanade, "Robust L1 norm factorization in the presence of outliers and missing data by alternative convex programming," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2005)*, June 2005.

[16] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, vol. 24, no. 6, pp. 381–395, June 1981.

[17] 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*, November 2009.

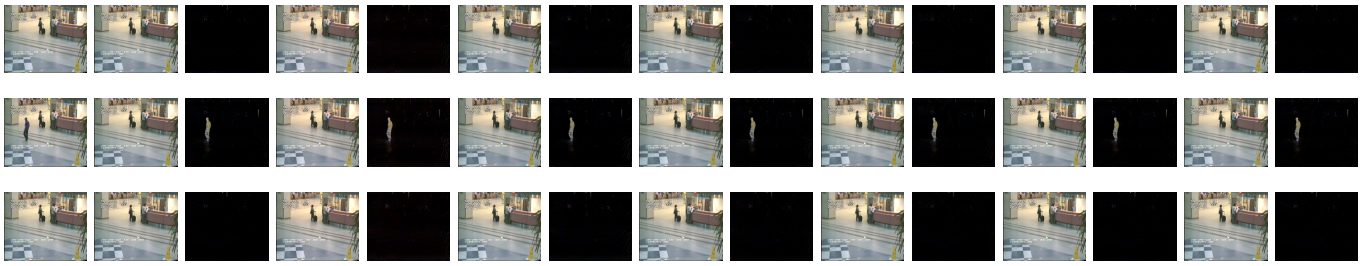
[18] X. Yuan and J. Yang, "Sparse and low-rank matrix decomposition via alternating direction methods," *Optimization Online*, November 2009.

[19] A. Ganesh, J. Wright, X. Li, E. J. Candès, and Y. Ma, "Dense error correction for low-rank matrices via principal component pursuit," in *2010 IEEE International Symposium on Information Theory*, June 2010, pp. 1513–1517.

[20] H. Ji, C. Liu, Z. Shen, and Y. Xu, "Robust video denoising using low rank matrix completion," in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, June 2010, pp. 1791–1798.

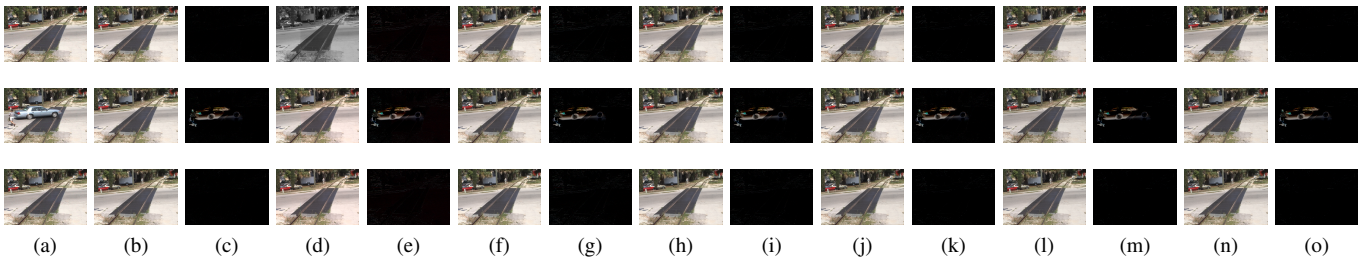
[21] J. F. Cai, E. J. Candes, and Z. Shen, "A singular value thresholding algorithm for matrix completion," *SIAM J. on Optimization*, vol. 20, no. 4, pp. 1956–1982, March 2010.

[22] J. Yao, X. Liu, and C. Qi, "Foreground detection using low rank and structured sparsity," in *2014 IEEE International Conference on Multimedia and Expo (ICME)*, July 2014, pp. 1–6.



(a) (b) (c) (d) (e) (f) (g) (h) (i) (j) (k) (l) (m) (n) (o)

Fig. 8. Background modeling results on a 176×144 video sequence. (a) Original frames. (b) Low rank component (ALM). (c) Sparse component (ALM). (d) Low rank component (LSSD). (e) Sparse component (LSSD). (f) Low rank component (SpaRCS). (g) Sparse component (SpaRCS). (h) Low rank component (LRSD-TNNSR). (i) Sparse component (LRSD-TNNSR). (j) Low rank component (RASL). (k) Sparse component (RASL). (l) Low rank component (LSD-HSN). (m) Sparse component (LSD-HSN). (n) Low rank component (LSD-GSN). (o) Sparse component (LSD-GSN).



(a) (b) (c) (d) (e) (f) (g) (h) (i) (j) (k) (l) (m) (n) (o)

Fig. 9. Background modeling results on a 360×240 video sequence. (a) Original frames. (b) Low rank component (ALM). (c) Sparse component (ALM). (d) Low rank component (LSSD). (e) Sparse component (LSSD). (f) Low rank component (SpaRCS). (g) Sparse component (SpaRCS). (h) Low rank component (LRSD-TNNSR). (i) Sparse component (LRSD-TNNSR). (j) Low rank component (RASL). (k) Sparse component (RASL). (l) Low rank component (LSD-HSN). (m) Sparse component (LSD-HSN). (n) Low rank component (LSD-GSN). (o) Sparse component (LSD-GSN).

[23] M. Rahmani and G. K. Atia, "High dimensional low rank plus sparse matrix decomposition," *IEEE Transactions on Signal Processing*, vol. 65, no. 8, pp. 2004–2019, April 2017.

[24] X. Ding, L. He, and L. Carin, "Bayesian robust principal component analysis," *IEEE Transactions on Image Processing*, vol. 20, no. 12, pp. 3419–3430, December 2011.

[25] C. Lu, J. Tang, S. Yan, and Z. Lin, "Nonconvex nonsmooth low rank minimization via iteratively reweighted nuclear norm," *IEEE Transactions on Image Processing*, vol. 25, no. 2, pp. 829–839, February 2016.

[26] T. H. Oh, Y. Matsushita, I. Kweon, and D. Wipf, "A pseudo-bayesian algorithm for robust pca," in *Advances in Neural Information Processing Systems 29*, D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, Eds. Curran Associates, Inc., 2016, pp. 1390–1398.

[27] D. Wipf, "Non-convex rank minimization via an empirical bayesian approach," in *Proceedings of the Twenty-Eighth Conference on Uncertainty in Artificial Intelligence*, ser. UAI'12. Arlington, Virginia, United States: AUAI Press, 2012, pp. 914–923.

[28] Z. Wen, W. Yin, and Y. Zhang, "Solving a low-rank factorization model for matrix completion by a nonlinear successive over-relaxation algorithm," *Mathematical Programming Computation*, vol. 4, no. 4, pp. 333–361, Dec 2012.

[29] Y. Zhang, "LMaFit: low-rank matrix fitting," 2009, [Online]. Available: http://www.caam.rice.edu/ optimization/L1/LMaFit.

[30] Y. Shen, Z. Wen, and Y. Zhang, "Augmented lagrangian alternating direction method for matrix separation based on low-rank factorization," *Optimization Methods Software*, vol. 29, no. 2, pp. 239–263, March 2014.

[31] G. H. Mohimani, M. Babaie-Zadeh, and C. Jutten, "Fast sparse representation based on smoothed $\ell 0$ norm," in *Proceedings of 7th International Conference on Independent Component Analysis and Signal Separation (ICA2007), LNCS 4666*, September 2007, pp. 389–396.

[32] J. Trzasko and A. Manduca, "Highly undersampled magnetic resonance image reconstruction via homotopic ℓ_0 -minimization," *IEEE Transactions on Medical Imaging*, vol. 28, no. 1, pp. 106–121, January 2009.

[33] H. Mohimani, M. Babaie-Zadeh, and C. Jutten, "A fast approach for overcomplete sparse decomposition based on smoothed ℓ^0 norm," *IEEE Transactions on Signal Processing*, vol. 57, no. 1, pp. 289–301, January 2009.

[34] D. P. Bertsekas, *Nonlinear Programming*. Belmont, MA, USA: Athena Scientific, 1999.

[35] A. Blake and A. Zisserman, *Visual Reconstruction*. Cambridge, MA, USA: MIT Press, 1987.

[36] W. Rudin, *Principles of mathematical analysis*, 3rd ed. New York: McGraw-Hill Book Co., 1976, international Series in Pure and Applied Mathematics.

[37] X. Liu, G. Zhao, J. Yao, and C. Qi, "Background subtraction based on low-rank and structured sparse decomposition," *IEEE Transactions on Image Processing*, vol. 24, no. 8, pp. 2502–2514, August 2015.

[38] A. E. Waters, A. C. Sankaranarayanan, and R. Baraniuk, "SpaRCS: Recovering low-rank and sparse matrices from compressive measurements," in *Advances in Neural Information Processing Systems 24*. Curran Associates, Inc., 2011, pp. 1089–1097.

[39] Z. Xue, J. Dong, Y. Zhao, C. Liu, and R. Chellali, "Low-rank and sparse matrix decomposition via the truncated nuclear norm and a sparse regularizer," *The Visual Computer*, May 2018.

[40] Y. Peng, A. Ganesh, J. Wright, W. Xu, and Y. Ma, "RASL: Robust alignment by sparse and low-rank decomposition for linearly correlated images," in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, June 2010, pp. 763–770.

[41] A. S. Georghiades, P. N. Belhumeur, and D. J. Kriegman, "From few to many: illumination cone models for face recognition under variable lighting and pose," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, no. 6, pp. 643–660, June 2001.

[42] "Statistical modeling of complex background for foreground object detection," 2016, [Online]. Available: http://perception.i2r.a-star.edu.sg/bk_model/bk_index.html. [Accessed: 10-May-2016].

[43] Y. Sheikh and M. Shah, "Bayesian modeling of dynamic scenes for object detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 11, pp. 1778–1792, November 2005.