Online Robust Principal Component Analysis With Change Point Detection

Wei Xiao, Xiaolin Huang, *Senior Member, IEEE*, Fan He, Jorge Silva, *Member, IEEE*, Saba Emrani, and Arin Chaudhuri

***Abstract*—Robust principal component analysis (PCA) is a key technique for dynamical high-dimensional data analysis, including background subtraction for surveillance video. Typically, robust PCA requires all observations to be stored in memory before processing. The batch manner makes robust PCA inefficient for big data. In this paper, we develop an efficient online robust PCA method, namely, online moving window robust principal component analysis (OMWRPCA). Unlike the existing algorithms, OMWRPCA can successfully track not only slowly changing subspaces but also abruptly changing subspaces. Embedding hypothesis testing into the algorithm enables OMWRPCA to detect change points of the underlying subspaces. Extensive numerical experiments, including real-time background subtraction, demonstrate the superior performance of OMWRPCA compared with other state-of-the-art approaches.**

***Index Terms*—Change point detection, online, principal component analysis.**

I. Introduction

LOW-RANK subspace models are powerful tools to analyze high-dimensional data from dynamical systems. The applications include but are not limited to tracking [1], face recognition [2], recommender systems [3], cloud removal in satellite images [4], anomaly detection [5], and background subtraction for surveillance video [6]–[8]. For finding low-rank subspaces from high dimensional data, principal component analysis (PCA, [9]) is the most widely used tool. PCA finds r dimensional linear subspaces that minimize the squared error between the data vector and its projection. Although PCA has been widely used, it has the drawback that it is not robust against outliers. Even with one grossly corrupted entry, the low-rank subspace that is estimated from classic PCA can be arbitrarily far from the true subspace. This shortcoming puts the application of the PCA technique into jeopardy for practical usage because outliers are often observed in the modern world's massive data. For example, data collected through sensors, cameras, and websites are often heavily noisy and contain error entries or outliers.

Various versions of robust PCA have been developed in the past few decades, see, e.g., [6], [10]–[13]. Among them, the Robust PCA based on Principal Component Pursuit (RPCA-PCP) [6], [14] is the most promising one, as it shares both good practical performance and solid theoretical analysis. RPCA-PCP decomposes the observed matrix into a low-rank matrix and a sparse matrix by principal component pursuit. Under a mild condition, both the low-rank matrix and the sparse matrix can be recovered exactly. A comprehensive review of the application of RPCA-PCP on surveillance video can be found in [8], [15]. RPCA-PCP has also been extended to handle high dimensional data with randomly sampled rows/columns [16], [17]. Some works finds the robust subspace by optimizing a slightly different L1-norm maximization criterion [18], [19], and have been extended to compressed-sensed surveillance videos [20]. Other related interesting works could be found in [21]–[25].

Most robust PCA methods, including RPCA-PCP, are implemented in a batch manner, which requires all data to be available before processing. Thus, for big data, these methods become infeasible on both memory storage and computational time. An online version of robust PCA method is highly necessary to process incremental data, where the tracked subspace can be updated whenever a new sample arrives. Only with online version, robust PCA is applicable to video analysis, change point detection, abnormal events detection, and network monitoring.

These are the main existing online robust PCA methods: Grassmannian Robust Adaptive Subspace Tracking Algorithm (GRASTA, [26]), Recursive Projected Compress Sensing (ReProCS, [27]–[29]), and Online Robust PCA via Stochastic Optimization (RPCA-STOC, [30]). GRASTA applies incremental gradient descent method on Grassmannian manifold for robust PCA. Its foundation is Grassmannian Rank-One Update Subspace Estimation (GROUSE, [31]), a widely used online subspace tracking algorithm. GRASTA uses the ℓ_1 cost function to replace the ℓ_2 cost function in GROUSE. ReProCS is designed mainly for foreground and background separation for video sequence, of which the key ideas are on perpendicular projection and sparse recovery. RPCA-STOC is based on stochastic

Manuscript received June 14, 2018; revised April 8, 2019 and May 28, 2019; accepted June 2, 2019. Date of publication June 14, 2019; date of current version December 31, 2019. The work of X. Huang was supported by the National Natural Science Foundation of China under Grant 61603248 and Grant IMR2018QY01. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Jingdong Wang. *(Corresponding authors: W. Xiao and X. Huang).*

W. Xiao was with SAS Institute, Cary, NC 27513 USA. He is now with Amazon, Inc., Seattle, WA 98109 USA (e-mail: wxiao@ncsu.edu).

X. Huang and F. He are with the Institute of Image Processing and Pattern Recognition, Institute of Medical Robotics, Shanghai Jiao Tong University, Shanghai 200240, China, and also with MOE Key Laboratory of System Control and Information Processing, Shanghai 200240, China (e-mail: xiaolinhuang@sjtu.edu.cn; hf-inspire@sjtu.edu.cn).

J. Silva and A. Chaudhuri are with SAS Institute Inc., Cary, NC 27513 USA (e-mail: jorge.silva@sas.com; arin.chaudhuri@sas.com).

S. Emrani was with SAS Institute Inc., Cary, NC 27513 USA. She is now with Apple Inc, Cupertino, CA 95014 USA (e-mail: semrani@ncsu.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TMM.2019.2923097

optimization on a reformulation of RPCA-PCP. Specifically, RPCA-STOC splits the nuclear norm of the low-rank matrix as the sum of Frobenius norm, and iteratively updates the low-rank subspace to find the sparse vector.

GRASTA and ReProCS are only able to handle slowly changing subspaces; while RPCA-STOC works only with a stable subspace. However, in practice, the low-rank subspace might change suddenly, and the corresponding change points are usually of great interest. For example, in the application of background subtraction from video, each scene change corresponds to a change point in the underlying subspace. Another application is in failure detection of mechanical systems based on sensor readings from the equipment. When a failure happens, the underlying subspace is changed. Accurately identifying the change point is useful for failure diagnosis. Other applications include human activity recognition, intrusion detection in computer networks, and so on.

To handle both slowly and abruptly changing subspaces, in this paper we propose an efficient online and robust method, called Online Moving Window Robust Principal Component Analysis (OMWRPCA). It embeds hypothesis testing into an online robust PCA framework and then identifies the change points of the underlying subspace, by which the low-rank subspace and sparse error are estimated at the same time. To the best of the authors' knowledge, OMWRPCA is the first algorithm that is able to simultaneously detect change points and compute RPCA in an online fashion. It is also the first online RPCA algorithm that can handle both slowly changing and abruptly changing subspaces. Our method is also closely linked with Low-Rank Representation [32], [33], which recovers the authentic data that lie on a union of multiple subspaces with outliers.

Recently, a new Robust Subspace Tracking (RST) method, namely (NORST, [34]) has been invented, which is equipped with change point detection capability. However, unlike our method, NORST can only handle piecewise constant subspaces and it requires the subspaces to be of a fixed rank. Moreover, it is not able to locate the exact position of a change point as it searches for a change point every α observations (where α is a parameter set by the user and usually takes a value above 50).

The remainder of this paper is organized as follows. Section II describes the problem and reviews the related methods. The OMWRPCA algorithm is developed in Section III. In Section IV, the proposed methods are evaluated on extensive numerical experiments, including an application on real-world video surveillance data. Section V ends the paper with conclusions and discussions.

II. Problem Definition and Related Works

A. Notations and Problem Definition

In this paper, bold letters stand for vectors and matrices, $\|\boldsymbol{a}\|_1$ and $\|\boldsymbol{a}\|_2$ for the ℓ_1-norm and ℓ_2-norm of a vector $\boldsymbol{a}$, respectively, $\|\boldsymbol{A}\|_F$ for the Frobenius norm of matrix an arbitrary real matrix $\boldsymbol{A}$, $\|\boldsymbol{A}\|_*$ for the nuclear norm (sum of all singular values), and $\|\boldsymbol{A}\|_1$ for the ℓ_1-norm.

Let T denote the number of observed samples, and t the index of the sample instance. We assume that the inputs are streaming observations $\boldsymbol{m}_t \in \mathbb{R}^m$, $t = 1, \ldots, T$, which can be decomposed as two parts $\boldsymbol{m}_t = \boldsymbol{l}_t + \boldsymbol{s}_t$. The first part $\boldsymbol{l}_t$ is a vector from a low-rank subspace $\boldsymbol{U}_t$, and the second part $\boldsymbol{s}_t$ is a sparse error with support size c_t. Here c_t represents the number of nonzero elements of $\boldsymbol{s}_t$, i.e., $c_t = \sum_{i=1}^{m} I_{\boldsymbol{s}_t[i]\neq 0}$. The underlying subspace $\boldsymbol{U}_t$ might or might not change with time t. When $\boldsymbol{U}_t$ does not change over time, we say the data are generated from a stable subspace. Otherwise, the data are generated from a changing subspace. We assume $\boldsymbol{s}_t$ satisfies the sparsity assumption, i.e., $c_t \ll m$ for all $t = 1, \ldots, T$. We use $\boldsymbol{M}_t = [\boldsymbol{m}_1, \ldots, \boldsymbol{m}_t] \in \mathbb{R}^{m\times t}$ to denote the matrix of observed samples until time t. Let $\boldsymbol{M}, \boldsymbol{L}, \boldsymbol{S}$ denotes $\boldsymbol{M}_T, \boldsymbol{L}_T, \boldsymbol{S}_T$ respectively.

Algorithm 1: Principal Component Pursuit by Augumented Largrangian Multiplier [6],[36]

Input: $\boldsymbol{M}$ (observed data), λ, μ (regularization parameters);
Initialize: $\boldsymbol{S}_0 = \boldsymbol{Y}_0 = \boldsymbol{0}$;
while *not converged* **do**
  1) $\boldsymbol{L}^{(k+1)} \leftarrow \mathcal{D}_{\mu^{-1}}(\boldsymbol{M} - \boldsymbol{S}^{(k)} + \mu^{-1}\boldsymbol{Y}^{(k)})$;
  2) $\boldsymbol{S}^{(k+1)} \leftarrow \mathcal{S}_{\lambda\mu^{-1}}(\boldsymbol{M} - \boldsymbol{L}^{(k+1)} + \mu^{-1}\boldsymbol{Y}^{k})$;
  3) $\boldsymbol{Y}^{(k+1)} \leftarrow \boldsymbol{Y}^{(k)} + \mu(\boldsymbol{M} - \boldsymbol{L}^{(k+1)} - \boldsymbol{S}^{(k+1)})$;
return $\boldsymbol{L}$ *(low-rank data matrix),* $\boldsymbol{S}$ *(sparse noise matrix)*

B. Robust Principal Component Analysis

Robust Principal Component Analysis based on Principal Component Pursuit (RPCA-PCP, [6], [14]) which decomposes the observed matrix $\boldsymbol{M}$ into a low-rank matrix $\boldsymbol{L}$ and a sparse matrix $\boldsymbol{S}$ by principal component pursuit:

$$\min_{\boldsymbol{L},\boldsymbol{S}} \|\boldsymbol{L}\|_* + \lambda\|\boldsymbol{S}\|_1 \text{ subject to } \boldsymbol{L} + \boldsymbol{S} = \boldsymbol{M}. \tag{1}$$

Theoretically, $\boldsymbol{L}$ and $\boldsymbol{S}$ can be recovered exactly if 1) the rank of $\boldsymbol{L}$ and the support size of $\boldsymbol{S}$ are small enough; 2) $\boldsymbol{L}$ is sufficiently "dense"; 3) any element in the sparse matrix $\boldsymbol{S}$ is nonzero with probability ρ and 0 with probability $1-\rho$ independent over time [6], [26]. RPCA-PCP is able to estimate a low-rank matrix to approximate the observed data when the inputs are corrupted with gross but sparse errors.

Various algorithms have been developed to solve RPCA-PCP, including Accelerated Proximal Gradient (APG, [35]) and Augmented Lagrangian Multiplier (ALM, [36]). According to experiments reported by [6], ALM achieves higher accuracy than APG, in a fewer iterations. Accordingly, ALM is utilized in this paper and is outlined in Algorithm 1. In Algorithm 1, $S_\tau[x] = \text{sgn}(x)\max(|x| - \tau, 0)$ denotes a shrinkage operator and S_τ is extended to matrices by applying it to each element. Besides, $\mathcal{D}_\tau(\boldsymbol{X})$ denotes the singular value thresholding operator given by $\mathcal{D}_\tau(\boldsymbol{X}) = \boldsymbol{U}\mathcal{S}_\tau(\boldsymbol{\Sigma})\boldsymbol{V}^*$, where $\boldsymbol{X} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^*$ is any singular value decomposition (SVD).

As a batch algorithm, RPCA-PCP iteratively solves SVD with all data. Thus, it requires a huge amount of memory and does not scale well with big data.

C. Moving Window Robust PCA

In practice, the underlying subspace can change over time, which makes RPCA-PCP infeasible as the rank of matrix $\boldsymbol{L}$ (which includes all observed samples) will increase over time. One solution is to apply a sliding window to the original data, resulting in Moving Window Robust Principal Component Analysis (MWRPCA), which iteratively calculates batch RPCA-PCP using only the latest n_{win} data column vectors, where n_{win} is a user specified window size.

MWRPCA generally performs well for subspace tracking with the capability of tracking on both slowly changing subspaces and abruptly changed subspaces. However, MWRPCA often becomes too slow to deal with real-world big data problems. For example, to track the subspace on a 200×10000 matrix, if the window size is set to be $n_{\text{win}} = 1000$, we need to run RPCA-PCP 9000 times on 200×1000 matrices, which is computationally very expensive.

D. Robust PCA via Stochastic Optimization

An online algorithm is proposed in [30] for robust PCA-PCP by solving the following problem

$$\min_{\boldsymbol{L},\boldsymbol{S}} \frac{1}{2}\|\boldsymbol{M}-\boldsymbol{L}-\boldsymbol{S}\|_F^2 + \lambda_1\|\boldsymbol{L}\|_* + \lambda_2\|\boldsymbol{S}\|_1, \tag{2}$$

where λ_1, λ_2 are tuning parameters. The main difficulty in developing an online algorithm for the above problem is that the nuclear norm couples all the samples tightly. Following the discussions in [37], [30] considers the problem based on the equivalence for the nuclear norm of a matrix $\boldsymbol{L}$ with a upper bounded rank r:

$$\|\boldsymbol{L}\|_* = \inf_{\boldsymbol{U},\boldsymbol{V}} \left\{ \frac{1}{2}\|\boldsymbol{U}\|_F^2 + \frac{1}{2}\|\boldsymbol{V}\|_F^2 : \boldsymbol{L} = \boldsymbol{U}\boldsymbol{V} \right\}. \tag{3}$$

Plugging (3) into (2), we see the problem (2) is equivalent to

$$\min_{\boldsymbol{U},\boldsymbol{V},\boldsymbol{S}} \frac{1}{2}\|\boldsymbol{M}-\boldsymbol{U}\boldsymbol{V}-\boldsymbol{S}\|_F^2 + \frac{\lambda_1}{2}(\|\boldsymbol{U}\|_F^2 + \|\boldsymbol{V}\|_F^2) + \lambda_2\|\boldsymbol{S}\|_1, \tag{4}$$

where $\boldsymbol{U}$ can be seen as the basis for low-rank subspace and $\boldsymbol{V}$ represents the coefficients of observations with respect to the basis. Based on this observation, a RPCA-STOC algorithm is developed in [30], which minimizes the empirical version of (4) and processes one sample per time instance. Given a finite set of samples $\boldsymbol{M}_t = [\boldsymbol{m}_1, \ldots, \boldsymbol{m}_t]$, the empirical version of (4) at time point t is

$$f_t(\boldsymbol{U}) = \frac{1}{t}\sum_{i=1}^{t} \ell(\boldsymbol{m}_i, \boldsymbol{U}) + \frac{\lambda_1}{2t}\|\boldsymbol{U}\|_F^2,$$

where the loss for each sample is defined as follows:

$$\ell(\boldsymbol{m}_i, \boldsymbol{U}) \triangleq \min_{\boldsymbol{v},\boldsymbol{s}} \frac{1}{2}\|\boldsymbol{m}_i - \boldsymbol{U}\boldsymbol{v} - \boldsymbol{s}\|_2^2 + \frac{\lambda_1}{2}\|\boldsymbol{v}\|_2^2 + \lambda_2\|\boldsymbol{s}\|_1.$$

With fixed $\boldsymbol{U} = \boldsymbol{U}_{t-1}$, $\boldsymbol{v}_t$ and $\boldsymbol{s}_t$ can be obtained by solving the following optimization problem:

$$\min_{\boldsymbol{v},\boldsymbol{s}} \frac{1}{2}\|\boldsymbol{m}_t - \boldsymbol{U}_{t-1}\boldsymbol{v} - \boldsymbol{s}\|_2^2 + \frac{\lambda_1}{2}\|\boldsymbol{v}\|_2^2 + \lambda_2\|\boldsymbol{s}\|_1.$$

Then with known $\{\boldsymbol{v}_i, \boldsymbol{s}_i\}_{i=1}^t$, the basis $\boldsymbol{U}_t$ is updated by minimizing the following function:

$$g_t(\boldsymbol{U}) \triangleq \frac{1}{t}\sum_{i=1}^{t}\left(\frac{1}{2}\|\boldsymbol{m}_i - \boldsymbol{U}\boldsymbol{v}_i\|_2^2 + \frac{\lambda_1}{2}\|\boldsymbol{v}_i\|_2^2 + \lambda_2\|\boldsymbol{s}_i\|_1\right) + \frac{\lambda_1}{2t}\|\boldsymbol{U}\|_F^2,$$

of which the minimum can be explicitly obtained as follows:

$$\boldsymbol{U}_t = \left[\sum_{i=1}^{t}(\boldsymbol{m}_i - \boldsymbol{s}_i)\boldsymbol{v}_i^{\mathsf{T}}\right]\left[\left(\sum_{i=1}^{t}\boldsymbol{v}_i\boldsymbol{v}_i^{\mathsf{T}}\right) + \lambda_1 I\right]^{-1}.$$

In practice, $\boldsymbol{U}_t$ can be quickly updated by block-coordinate descent with warn restarts. The details of the RPCA-STOC algorithm can be found in [30].

III. Online Moving Window RPCA

A. Basic Algorithm

RPCA-STOC strongly assumes the existence of a unique stable subspace, which is generally too strict and excludes many real-world applications such as failure detection in mechanical systems, intrusion detection in computer networks, fraud detection in financial transaction, and background subtraction for surveillance video. The essential problem is that RPCA-STOC updates the basis of subspace by minimizing an empirical loss involving all previously observed samples with equal weights, from which it follows that the obtained subspace is actually an "average" subspace of observations. This is clearly not suitable if the underlying subspace is changing over time.

To deal with this difficulty, we propose a new method which combines the idea of moving window RPCA and RPCA-STOC to track changing subspaces. At time t, we update $\boldsymbol{U}_t$ by minimizing an empirical loss based only on the most recent n_{win} samples. Here n_{win} is a user specified window size and the new empirical loss is defined as

$$g_t^*(\boldsymbol{U}) \triangleq \frac{1}{n_{\text{win}}}\sum_{i=t-n_{\text{win}}+1}^{t}\left(\frac{1}{2}\|\boldsymbol{m}_i - \boldsymbol{U}\boldsymbol{v}_i\|_2^2 + \frac{\lambda_1}{2}\|\boldsymbol{v}_i\|_2^2 + \lambda_2\|\boldsymbol{s}_i\|_1\right) + \frac{\lambda_1}{2n_{\text{win}}}\|\boldsymbol{U}\|_F^2.$$

The proposed method is called Online Moving Window RPCA (OMWRPCA). Compared with RPCA-STOC, the major advantage of OMWRPCA is that OMWRPCA can quickly update the subspace when the underlying subspace is changing. Moreover, it has two other differences: 1) In RPCA-STOC, the dimension of the subspace, namely r, should be known. In OMWRPCA, we estimate r by computing batch RPCA-PCP with burn-in samples. 2) The number of burn-in samples n_{burnin} is a user specified parameter in RPCA-STOC. While, we use the estimated $\boldsymbol{U}$ from the burn-in samples in OMWRPCA. The basic algorithm for OMWRPCA is summarized in Algorithm 2.

Algorithm 2: Online Moving Window RPCA

Input: $\{\boldsymbol{m}_1, \ldots, \boldsymbol{m}_T\}$ (observed data revealed sequentially), λ_1, λ_2 (regularization parameters), T (number of iterations); $\boldsymbol{M}^b = [\boldsymbol{m}_{-(n_{\text{burnin}}-1)}, \ldots, \boldsymbol{m}_0]$ (burn-in samples);

Initialize: Compute batch RPCA-PCP on burn-in samples $\boldsymbol{M}^b$ to get r, $\boldsymbol{U}_0$, A_0 and B_0.

for *t=1 to T* **do**

  1) Reveal the sample $\boldsymbol{m}_t$.

  2) Project new sample: $(\boldsymbol{v}_t, \boldsymbol{s}_t) \leftarrow \operatorname{argmin} \frac{1}{2}\|\boldsymbol{m}_t - \boldsymbol{U}_{t-1}\boldsymbol{v} - \boldsymbol{s}\|_2^2 + \frac{\lambda_1}{2}\|\boldsymbol{v}\|_2^2 + \lambda_2\|\boldsymbol{s}\|_1$.

  3) $A_t \leftarrow A_{t-1} + \boldsymbol{v}_t\boldsymbol{v}_t^\top - \boldsymbol{v}_{t-n_{\text{win}}}\boldsymbol{v}_{t-n_{\text{win}}}^\top$,

  $B_t \leftarrow B_{t-1} + (\boldsymbol{m}_t - \boldsymbol{s}_t)\boldsymbol{v}_t^\top - (\boldsymbol{m}_{t-n_{\text{win}}} - \boldsymbol{s}_{t-n_{\text{win}}})\boldsymbol{v}_{t-n_{\text{win}}}^\top$.

  4) Compute $\boldsymbol{U}_t$ with $\boldsymbol{U}_{t-1}$ as warm restart $\boldsymbol{U}_t = \operatorname{argmin} \frac{1}{2}\operatorname{Tr}[\boldsymbol{U}^\top(A_t + \lambda_1 I)\boldsymbol{U}] - \operatorname{Tr}(\boldsymbol{U}^\top B_t)$.

return $\boldsymbol{L}_T = \{\boldsymbol{U}_0\boldsymbol{v}_1, \ldots, \boldsymbol{U}_{(T-1)}\boldsymbol{v}_T\}$, $\boldsymbol{S}_T = \{\boldsymbol{s}_1, \ldots, \boldsymbol{s}_T\}$

B. Change Point Detection in Online Moving Window RPCA With Hypothesis Testing

One limitation of the basic OMWRPCA is that it can only handle slowly changing subspaces. When a subspace changes suddenly, the basic OMWRPCA algorithm will fail to update the subspace correctly. This is because when the new subspace is dramatically different from the original one, online updates of the subspace might not be possible or it may take some time to finish the updates. A special case is when the new subspace is in a higher dimension space than the original subspace, the basic OMWRPCA algorithm is unable to update the subspace correctly because the rank of the subspaces are held fixed during updates. This drawback is suffered by almost all the existing online RPCA algorithms.

Therefore, change point detection is very crucial for subsequent identification, and an online RPCA algorithm that can correctly identify a change point is very desirable. In fact, users sometimes are more interested in pinpointing the change points than in estimating the underlying subspace. Thus, we designed a variant of our OMWRPCA algorithm to simultaneously detect change points and compute RPCA in an online fashion. The algorithm is robust to dramatic subspace changes. Since this goal is achieved by embedding hypothesis testing, we call the new algorithm OMWRPCA-CP.

OMWRPCA-CP is based on a simple yet important observation: when the new observation can not be modeled well with the current subspace, a sparse vector $\hat{\boldsymbol{s}}_t$ with an abnormally large support size $\hat{c}_t$ will be estimated from the OMWRPCA algorithm. So by monitoring the support size of the sparse vector calculated from OMWRPCA, we can identify the change point of the subspace, which is usually the starting point of level increases in the time series of $\hat{c}_t$.

Fig. 1 demonstrates this observation on four simulated cases. In each plot, support sizes of estimated sparse vectors are plotted. For each case, burn-in samples are between the indices -200 and 0, and two change points are at $t = 1000$ and $t = 2000$. The timeline is separated into three sections by the change points.

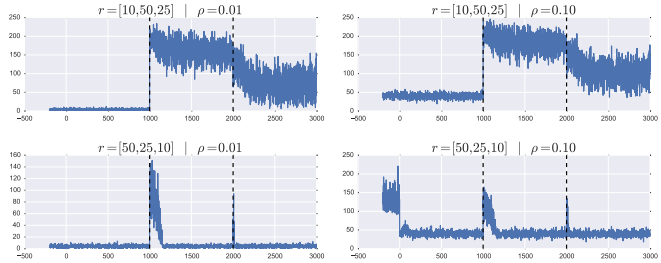


Fig. 1. Line plots of support sizes of estimated sparse vector from OMWRPCA based on four simulated data.

In each section, we have a constant underlying subspace whose dimension is given by r. The elements of the true sparse vector have a probability of ρ of being nonzero and are independently generated. The dimension of the samples is 400. We choose a window size of 200 in OMWRPCA. Theoretically, when the underlying subspace is accurately estimated, the support sizes of the estimated sparse vectors $\hat{c}_t$ should be around 4 and 40 for the $\rho = 0.01$ case and the $\rho = 0.1$ case, respectively. For the upper two cases in Fig. 1, $\hat{c}_t$ blows up after the first change point and never returns to a value in the normal range. This is because after fitting RPCA-PCP on burn-in samples, the dimension of $\boldsymbol{U}_t$ is fixed as 10. Thus, the estimated subspace $\boldsymbol{U}_t$ cannot approximate well the true subspace in the later two sections of the timeline as the true subspaces have a dimension larger than the fixed dimension. On the other hand, for the lower two cases in Fig. 1, $\hat{c}_t$ blows up immediately after each change point, and then drops back to the normal range after a while when the estimated subspace has converged to the true new subspace.

We now give an intuitive explanation for the above phenomenon. At time t, the observation is $\boldsymbol{m}_t = \boldsymbol{U}_t\boldsymbol{v}_t + \boldsymbol{s}_t$, and $(\hat{\boldsymbol{v}}_t, \hat{\boldsymbol{s}}_t)$ can be estimated as the optimum of the following problem:

$$\min_{\boldsymbol{v},\boldsymbol{s}} \frac{1}{2}\|\boldsymbol{m}_t - \hat{\boldsymbol{U}}_{t-1}\boldsymbol{v} - \boldsymbol{s}\|_2^2 + \frac{\lambda_1}{2}\|\boldsymbol{v}\|_2^2 + \lambda_2\|\boldsymbol{s}\|_1,$$

where $\hat{\boldsymbol{U}}_{t-1}$ is the estimated subspace at time point $t-1$. The above problem does not have an explicit solution but can be iteratively solved by alternatively optimizing $\boldsymbol{s}_t$ and $\boldsymbol{v}_t$. Approximating the solution with a one-step update from a reasonable initial point $\hat{\boldsymbol{s}}_t = \boldsymbol{s}_t$, we have:

$$\hat{\boldsymbol{v}}_t = (\hat{\boldsymbol{U}}_{t-1}^\top\hat{\boldsymbol{U}}_{t-1} + \lambda_1 I)^{-1}\hat{\boldsymbol{U}}_{t-1}^\top\boldsymbol{U}_t\boldsymbol{v}_t$$

$$\hat{\boldsymbol{s}}_t = \mathcal{S}_{\lambda_2}\Big\{\boldsymbol{s}_t + \Big[I - \hat{\boldsymbol{U}}_{t-1}(\hat{\boldsymbol{U}}_{t-1}^\top\hat{\boldsymbol{U}}_{t-1} + \lambda_1 I)^{-1}\hat{\boldsymbol{U}}_{t-1}\Big]\boldsymbol{U}_t\boldsymbol{v}_t\Big\}.$$

For a small λ_1, we have the approximations $\hat{\boldsymbol{v}}_t = (\hat{\boldsymbol{U}}_{t-1}^\top\hat{\boldsymbol{U}}_{t-1})^{-1}\hat{\boldsymbol{U}}_{t-1}^\top\boldsymbol{U}_t\boldsymbol{v}_t$ and $\hat{\boldsymbol{s}}_t = \mathcal{S}_{\lambda_2}\{\boldsymbol{s}_t + \mathcal{P}_{\hat{\boldsymbol{U}}_{t-1}^\perp}\boldsymbol{U}_t\boldsymbol{v}_t\}$, where $\mathcal{P}_{\hat{\boldsymbol{U}}_{t-1}^\perp}$ represents the projection to the orthogonal complement subspace of $\hat{\boldsymbol{U}}_{t-1}$. A good choice of λ_1 is of the order of $O(1/\sqrt{\max(m, n_{\text{win}})})$, and λ_1 is small when $\max(m, n_{\text{win}})$ is large. (We will discuss the tuning of parameters in more details later.) Under the mild assumption that the nonzero elements of $\boldsymbol{s}_t$ are not too small (that is:

$\min_{i\in\{i:\,\boldsymbol{s}_t[i]\neq 0\}} \boldsymbol{s}_t[i] > \lambda_2$), we have $\hat{c}_t = c_t$ when $\boldsymbol{U}_t = \hat{\boldsymbol{U}}_{t-1}$, and $\hat{c}_t = c_t + \#\{i:\ \boldsymbol{s}_t[i] = 0 \text{ and } \mathcal{P}_{\hat{\boldsymbol{U}}_{t-1}^{\perp}} \boldsymbol{U}_t \boldsymbol{v}_t[i] > \lambda_2\} > c_t$ when $\max_{i\in\{i:\,\boldsymbol{s}_t[i]=0\}} (\mathcal{P}_{\hat{\boldsymbol{U}}_{t-1}^{\perp}} \boldsymbol{U}_t \boldsymbol{v}_t[i]) > \lambda_2$. This analysis also provides us a clear mathematical description of the term "abruptly changed subspace", i.e., for a piecewise constant subspace, we say that a change point exists at time t when the vector $\mathcal{P}_{\boldsymbol{U}_{t-1}^{\perp}} \boldsymbol{U}_t \boldsymbol{v}_t$ is not close to zero.

Based on the above observations, we propose to monitor $\hat{c}_t$ as a change point detection algorithm. The algorithm determines that a change point exists when it finds $\hat{c}_t$ is abnormally high for a while. There are two user-given parameters: N_{check} and α_{prop}. If the algorithm finds more than α_{prop} observations with abnormally high $\hat{c}_t$ in N_{check} consecutive observations, the algorithm detects a change point and traces back to find the location of the change point. We refer to the cases where the underlying estimated subspace approximates the true subspace well as "normal", otherwise we refer to it as "abnormal". All the collected information of $\{\hat{c}_j\}_{j=1}^{t-1}$ from the normal periods is stored in $H_c \in \mathbb{R}^{m+1}$, where the $(i+1)$-th element of H_c equals the number of times that $\{\hat{c}_j\}$ is i. For calculating $\hat{c}_t$ from a new observation $\boldsymbol{m}_t$, based on H_c we can flag this observation as normal or an abnormal via hypothesis testing. The p-value is computed as $p = \sum_{i=\hat{c}_t+1}^{m+1} H_c[i] / \sum_{i=0}^{m+1} H_c[i]$, which is the probability for observing a sparse vector with at least as many nonzero elements as the current observation under the hypothesis that this observation is normal. If $p \leq \alpha$, the observation is flagged as abnormal, otherwise, this observation is regarded as normal. Here, α is a user-specified threshold and $\alpha = 0.01$ is suggested. The pseudocode of OMWRPCA-CP is displayed in Algorithm 3. Here are the key steps:

1) Initialize H_c as $\mathbf{0}^{m+1}$, the buffer list for flag B_f and the buffer list for count B_c as empty lists.
2) Collect n_{burnin} samples to get $\boldsymbol{M}^b$. Run RPCA-PCP on $\boldsymbol{M}^b$ and start OMWRPCA.
3) For the first $n_{\text{cp-burnin}}$ observations of OMWRPCA, wait for the subspace of the OMWRPCA algorithm to become stable. H_c, B_f and B_c remain unchanged.
4) For the next n_{test} observations of OMWRPCA, calculate $\hat{c}_t$ from $\hat{\boldsymbol{s}}_t$, and update $H_c[\hat{c}_t + 1] \leftarrow H_c[\hat{c}_t + 1] + 1$. B_f and B_c remain unchanged. This stage does not contain hypothesis testing, since the sample size is small and the testing will not be accurate.
5) Perform hypothesis testing on $\hat{c}_t$ with H_c and get the flag f_t for the t-th observation, where $f_t = 1$ if the t-th observation is abnormal, and $f_t = 0$ if the t-th observation is normal. Append f_t and $\hat{c}_t$ to B_f and B_c, respectively. Denote the size of buffer B_f or equivalently the size of buffer B_c as n_b. If n_b satisfies $n_b = N_{\text{check}} + 1$, pop one observation from both B_f and B_c on the left-hand side. If c is the number popped from B_c then update H_c with c ($H_c[c+1] \leftarrow H_c[c+1] + 1$). If $n_b < N_{\text{check}} + 1$, there is no update. The buffer size n_b is kept within N_{check}. When the buffer size reaches N_{check}, it remains in N_{check}. Buffers B_f contain flag information of the most recent N_{check} observations and is used to detect change points.

Algorithm 3: Online Moving Window RPCA with Change Point Detection

Input: $\{\boldsymbol{m}_1, \ldots, \boldsymbol{m}_T\}$ (observed data revealed sequentially), λ_1, λ_2 (regularization parameters)
$t = 1$; c_p is initialized as an empty list.
$t^* \leftarrow \min(t + n_{\text{burnin}} - 1, T)$; Compute batch RPCA-PCP on burn-in samples $\boldsymbol{M}^b = [\boldsymbol{m}_t, \ldots, \boldsymbol{m}_{t^*}]$ to get r, $\boldsymbol{U}_{t^*}$, A_{t^*} and B_{t^*}; $t \leftarrow t^* + 1$.
$t_{\text{start}} \leftarrow t$; $H_c \leftarrow \mathbf{0}^{m+1}$; B_f and B_c are initialized as empty lists.
while $t \leq T$ **do**
  1) Reveal the sample $\boldsymbol{m}_t$. 2) Project the new sample: $(\boldsymbol{v}_t, \boldsymbol{s}_t) \leftarrow \operatorname{argmin} \frac{1}{2}\|\boldsymbol{m}_t - \boldsymbol{U}_{t-1}\boldsymbol{v} - \boldsymbol{s}\|_2^2 + \frac{\lambda_1}{2}\|\boldsymbol{v}\|_2^2 + \lambda_2\|\boldsymbol{s}\|_1$.
  3) $A_t \leftarrow A_{t-1} + \boldsymbol{v}_t\boldsymbol{v}_t^{\mathsf{T}} - \boldsymbol{v}_{t-n_{\text{win}}}\boldsymbol{v}_{t-n_{\text{win}}}$,
  $B_t \leftarrow B_{t-1} + (\boldsymbol{m}_t - \boldsymbol{s}_t)\boldsymbol{v}_t^{\mathsf{T}} - (\boldsymbol{m}_{t-n_{\text{win}}} - \boldsymbol{s}_{t-n_{\text{win}}})\boldsymbol{v}_{t-n_{\text{win}}}^{\mathsf{T}}$.
  4) Compute $\boldsymbol{U}_t$ with $\boldsymbol{U}_{t-1}$ as warm restart: $\boldsymbol{U}_t \triangleq \operatorname{argmin} \frac{1}{2}\operatorname{Tr}[\boldsymbol{U}^{\mathsf{T}}(A_t + \lambda_1 I)\boldsymbol{U}] - \operatorname{Tr}(\boldsymbol{U}^{\mathsf{T}}B_t)$.
  5) Compute $c_t \leftarrow \sum_{i=1}^{m} \boldsymbol{s}_t[i]$.
  6) **if** $t < t_{\text{start}} + n_{\text{cp-burnin}}$ **then**
    Go to next loop;
  else if $t_{\text{start}} + n_{\text{cp-burnin}} \leq t < t_{\text{start}} + n_{\text{cp-burnin}} + n_{\text{test}}$ **then**
    $H_c[c_t + 1] \leftarrow H_c[c_t + 1] + 1$; Go to next loop;
  else
    Hypothesis testing on c_t with H_c, and get p-value p; $f_t \leftarrow I_{p\leq\alpha}$.
    Append c_t to B_c, and f_t to B_f.
    if *size of* $B_f = N_{\text{check}} + 1$ **then**
      Pop one element from both B_c and B_f at left-hand side ($c \leftarrow \text{Pop}(\text{B}_\text{c})$, $f \leftarrow \text{Pop}(\text{B}_\text{f})$);
      Update H_c ($H_c[c+1] \leftarrow H_c[c+1] + 1$).
    if *size of* $B_f = N_{\text{check}}$ **then**
      Compute $n_{\text{abnormal}} \leftarrow \sum_{i=1}^{N_{\text{check}}} B_f[i]$
      if $n_{\text{abnormal}} \geq \alpha_{\text{prop}} N_{\text{check}}$ **then**
        Find change point t_0 by looping over B_f; Append t_0 to c_p.
        $t \leftarrow t_0$; Jump to step 3.
      else
        Go to next loop;
return c_p *(list of all change points)*

6) If n_b equals N_{check}, we compute $n_{\text{abnormal}} = \sum_{i=1}^{N_{\text{check}}} B_f[i]$ and compare it with $\alpha_{\text{prop}}\ N_{\text{check}}$. If $n_{\text{abnormal}} \geq \alpha_{\text{prop}} N_{\text{check}}$, a change point exists in the most recent N_{check} observations. Then do a simple loop over the list B_f to find the change point, which is the first instance of n_{positive} consecutive abnormal cases. For example, suppose $B_f[i]$ corresponding to time points t_0, $t_0 + 1$, …, $t_0 + n_{\text{positive}} - 1$ is the first instance of n_{positive} consecutive abnormal cases. The change point is determined as t_0. Here n_{positive} is a parameter specified by the user. In practice, $n_{\text{positive}} = 3$ is suggested. After identifying the change point, OMWRPCA-CP restarts from it.

Tuning the parameters properly is important to the success of the proposed algorithms. One can select (λ_1, λ_2) based on cross-validation. A rule of thumb choice is $\lambda_1 = 1/\sqrt{\max(m, n_{\text{win}})}$ and $\lambda_2 = 100/\sqrt{\max(m, n_{\text{win}})}$. N_{check} needs to be kept smaller than $n_{\text{win}}/2$ to avoid missing a change point, yet not too small to avoid generating false alarms. α_{prop} can be chosen based on the user's prior knowledge. Sometimes the assumption that the support size of sparse vector $\boldsymbol{s}_t$ remains stable and much smaller than m is violated in real-world data. For example, in video surveillance data, the foreground might contain significant variations over time. In this case, we add one additional positive tuning parameter n_{tol} and modify the formula of p-value to $p = \sum_{i=\hat{c}_t - n_{\text{tol}}+1}^{m+1} H_t[i] / \sum_{i=0}^{m+1} H_t[i]$, which makes the hypothesis testing more conservative and forces the algorithm to detect fewer change points reducing false alarms.

It is easy to prove that OMWRPCA and OMWRPCA-CP (ignoring the burn-in samples training) have the same computational complexity as STOC-RPCA. The computational cost of each new observation is $O(mr^2)$, which is independent of the sample size and linear in the dimension of observation [30]. In contrast, RPCA-PCP computes an SVD and a thresholding operation in each iteration with the computational complexity $O(nm^2)$. The memory cost of RPCA-PCP is $O(mn)$ and that of OMWRPCA and OMWRPCA-CP are $O(mr)$ and $O(mr + N_{\text{check}})$, respectively. Generally, the proposed methods are well suitable to process big data.

Lastly, the current OMWRPCA-CP does hypothesis testing based on all of the historical information of $\hat{c}_t$. We can easily change H_c to a queue structure which only stores the recent history of $\hat{c}_t$. The user can specify how far in history to trace back. This change makes the algorithm detect a change point only by comparisons with recent history. Since the modification is straightforward, we do not present it here.

IV. Numerical Experiments

In this section, the proposed OMWRPCA algorithm will be evaluated by comparison with RPCA-STOC and NORST on extensive numerical experiments and an application to a real-world video surveillance data. To make a fair comparison between OMWRPCA and RPCA-STOC, we estimate the rank r in RPCA-STOC with burn-in samples following the same steps as OMWRPCA. For NORST, we use the code downloaded from https://github.com/praneethmurthy/NORST. All algorithms are implemented in Matlab and our code is available at https://github.com/wxiao0421/onlineRPCA.git.

We choose the parameters $\lambda_1 = 1/\sqrt{400}$, $\lambda_2 = 100/\sqrt{400}$, $n_{\text{win}} = 200$, $n_{\text{burnin}} = 200$, $n_{\text{cp-burnin}} = 200$, $n_{\text{test}} = 100$, $N_{\text{check}} = 20, \alpha_{\text{prop}} = 0.5, \alpha = 0.01, n_{\text{positive}} = 3, n_{\text{tol}} = 0$ for OMWRPCA. We do a grid search and choose the best parameters for NORST ($\omega_{evals} = 0.002$, $\alpha = 150$, $K = 4$, $\omega_{supp} = 20$) as the default parameters suggested in the paper do not work well.

A. Stable Subspace

The first simulation study is similar to the setting in [30]. The observations are generated through $\boldsymbol{M} = \boldsymbol{L} + \boldsymbol{S}$, where $\boldsymbol{S}$ is a sparse matrix with a fraction of ρ non-zero elements.

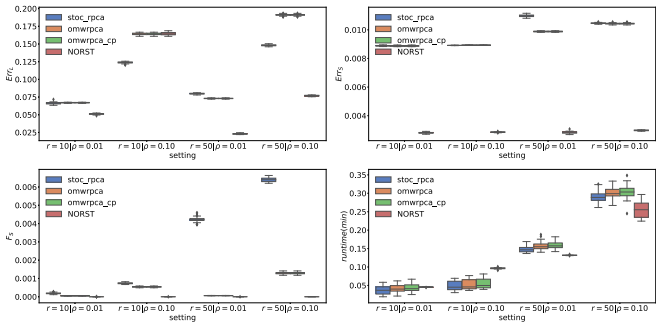


Fig. 2. Box plots of ERR_{L}, ERR_{S}, F_{S}, and running times under the simulation study of a stable subspace. The plots of STOC-RPCA, OMWRPCA, OMWRPCA-CP, and NORST are arranged from left to right.

The non-zero elements of $\boldsymbol{S}$ are randomly chosen and generated from a uniform distribution over the interval of $[-1000, 1000]$. The low-rank subspace $\boldsymbol{L}$ is generated as a product $\boldsymbol{L} = \boldsymbol{U}\boldsymbol{V}$, where the sizes of $\boldsymbol{U}$ and $\boldsymbol{V}$ are $m \times r$ and $r \times T$, respectively. The elements of both $\boldsymbol{U}$ and $\boldsymbol{V}$ are i.i.d. samples from $\mathcal{N}(0, 1)$. Here $\boldsymbol{U}$ is the basis of the constant subspace with dimension r. We fix $T = 5000$ and $m = 400$. Burn-in samples $\boldsymbol{M}^b$ of size 400×200 are generated and $(r,\ \rho)$ are given values $(10, 0.01)$, (10,0.1), (50,0.01), and (50,0.1). For each setting, we run 50 replications.

We compare STOC-RPCA, OMWRPCA, OMWRPCA-CP, and NORST on the following three criteria:

$$\text{ERR}_{\text{L}} = \|\hat{\boldsymbol{L}} - \boldsymbol{L}\|/\|\boldsymbol{L}\|_F;$$

$$\text{ERR}_{\text{S}} = \|\hat{\boldsymbol{S}} - \boldsymbol{S}\|/\|\boldsymbol{S}\|_F;$$

$$\text{F}_{\text{S}} = \#\{(\hat{\boldsymbol{S}} \neq \boldsymbol{0}) \neq (\boldsymbol{S} \neq \boldsymbol{0})\}/(mT).$$

Here ERR_{L} is the relative error of the low-rank matrix $\boldsymbol{L}$, ERR_{S} is the relative error of the sparse matrix $\boldsymbol{S}$, and F_{S} is the proportion of incorrectly identified elements in $\boldsymbol{S}$.

Box plots of ERR_{L}, ERR_{S}, F_{S} along with the running times are shown in Fig. 2. OMWRPCA-CP has the same result as OMWRPCA, and no change point is detected by OMWRPCA-CP in all replications. NORST falsely detected 6 equally spaced change points 750 observations apart. In fact, it adds one more change point whenever the algorithm switches to detect phase. All the three methods STOC-RPCA, OMWRPCA, and OMWRPCA-CP have comparable performance on ERR_{S} and running times. STOC-RPCA has slightly better performance on ERR_{L} when $\rho = 0.1$. NORST performs better than other three methods. All methods take approximately the same amount of time to run, and are all very fast (less than 0.5 minutes per replication for all settings). In contrast, MWRPCA takes around 100 minutes for the settings $(r,\ \rho) = (10, 0.01)$, (10,0.1), (50,0.01), and more than 1000 minutes for the setting $(r,\ \rho) = (50, 0.1)$.

B. Slowly Changing Subspace

In the next experiment, we adopt almost the same setting as previously described except that the underlying subspace $\boldsymbol{U}$ changes linearly over time. We first generate $\boldsymbol{U}_0 \in \mathbb{R}^{m \times r}$

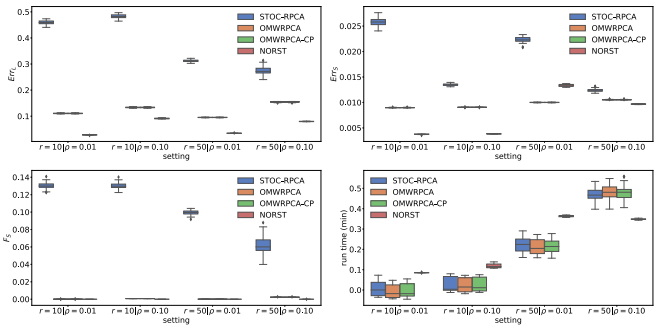


Fig. 3. Box plots of $\mathrm{ERR_L}$, $\mathrm{ERR_S}$, $\mathrm{F_S}$, and running times for the simulation study of a slowly changing subspace. The plots of STOC-RPCA, OMWRPCA, OMWRPCA-CP, and NORST are arranged from left to right.

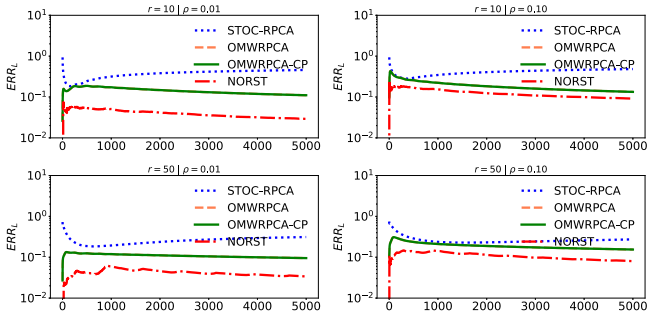


Fig. 4. Line plots of average $\mathrm{ERR_L}$ over time t under the simulation study of a slowly changing subspace. Results of OMWRPCA and OMWRPCA-CP completely overlap with each other.

with i.i.d. samples from $\mathcal{N}(0,1)$ and then generate burn-in samples $\boldsymbol{M}^b$ based on $\boldsymbol{U}_0$. $\boldsymbol{U}$ slowly changes over time by adding new matrices $\{\tilde{\boldsymbol{U}}_k\}_{k=1}^K$ that are generated independently with i.i.d. $\mathcal{N}(0,1)$ elements to the first r_0 columns of $\boldsymbol{U}$, where $\tilde{\boldsymbol{U}}_k \in \mathbb{R}^{m\times r_0}$, $k = 1,\ldots,K$ and $K = T/T_p$, where $T_p = 250$. Specifically, for $t = T_p * i + j$, where $i = 0,\ldots,K-1$, $j = 0,\ldots,T_p-1$, we have:

$$\boldsymbol{U}_t[:,1:r_0] = \boldsymbol{U}_0[:,1:r_0] + \sum_{1\le k\le i} \tilde{\boldsymbol{U}}_k + \frac{j}{n_p}\tilde{\boldsymbol{U}}_{i+1},$$

$$\boldsymbol{U}_t[:,(r_0+1):r] = \boldsymbol{U}_0[:,(r_0+1):r].$$

In this simulation, $r_0 = 5$.

The comparison result displayed in Fig. 3 shows that OMWRPCA, OMWRPCA-CP, and NORST outperform STOC-RPCA which is not able to efficiently track changing subspaces. NORST is slightly better than OMWRPCA-CP and OMWRPCA. OMWRPCA-CP has the same result as OMWRPCA, and no change point is detected. Again, NORST detects 6 equally spaced change points, which is reasonable since NORST assumes piecewise constant subspaces. In Fig. 4, we plot the average of $\mathrm{ERR_L}$ across all replications as a function of number of observations. It verifies that the performance of STOC-RPCA deteriorates over time, while that of OMWRPCA and OMWRPCA-CP is quite stable.

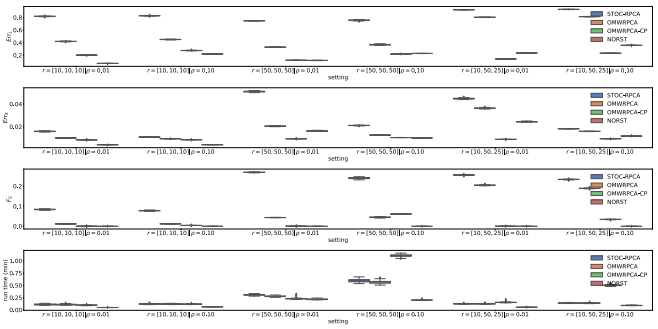


Fig. 5. Box plots of $\mathrm{ERR_L}$, $\mathrm{ERR_S}$, $\mathrm{F_S}$ for the simulation study of slowly changing subspace with change points. The plots of STOC-RPCA, OMWRPCA, OMWRPCA-CP, and NORST are arranged from left to right.

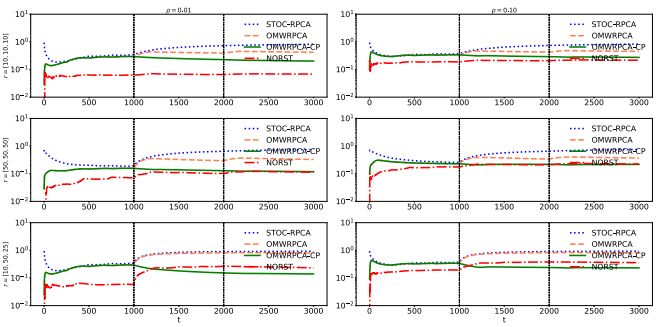


Fig. 6. Average $\mathrm{ERR_L}$ over time t for the simulation study of suddenly changed subspace. Two change points are at time point 1000 and 2000 are marked by vertical lines.

C. Slowly Changing Subspace With Change Points

Based on the previous experiments, we further add two change points at time points 1000 and 2000, i.e., the underlying subspace $\boldsymbol{U}$ is changed and generated completely independently. These two change points cut the timeline into three sections, and in section i the subspace $\boldsymbol{U}$ starts with rank r_i. Let $\boldsymbol{r} = (r_1, r_2, r_3)^\mathsf{T}$. We consider three settings of $\boldsymbol{r}$: $(10,10,10)^\mathsf{T}$, $(50,50,50)^\mathsf{T}$ and $(10,50,25)^\mathsf{T}$. The subspace $\boldsymbol{U}$ slowly changes over time in each section as assumed in numerical experiment B, where $T_p = 250$. In this experiment, $T = 3000$.

The box plots of $\mathrm{ERR_L}$, $\mathrm{ERR_S}$, and $\mathrm{F_S}$ at time point T are shown in Fig. 5. Under almost all settings, OMWRPCA outperforms STOC-RPCA. OMWRPCA-CP has the best performance among these three methods. Fig. 6 plots the average of $\mathrm{ERR_L}$ across all replications as a function of time t to investigate the progress of performance across different methods. The performance of both STOC-RPCA and OMWRPCA deteriorate quickly after the first change point at $t = 1000$, while the performance of OMWRPCA-CP remains stable, indicating that OMWRPCA-CP can track subspaces correctly under the scenario of a suddenly changed subspace. NORST outperforms OMWRPCA-CP in the case $\boldsymbol{r} = (10,10,10)^\mathsf{T}$ where all the subspaces have the same dimension, but underperforms OMWRPCA-CP in the case when $\boldsymbol{r} = (10,50,25)^\mathsf{T}$ where there is a sharp change in dimensions.

Furthermore, OMWRPCA-CP correctly identifies two change points for all replications over all settings. The

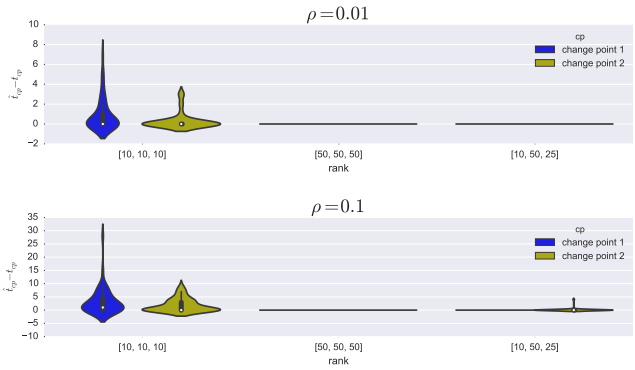


Fig. 7. Violin plots of the difference between detected change points and true change points under the simulation study of suddenly changed subspace.

distribution of the difference between the detected change points and the true change points ($\delta_{cp} = \hat{t}_{cp} - t_{cp}$) is given in Fig. 7. We observe that δ_{cp} is typically 0 when the subspaces before and after the change point are highly distinguishable, which represents the cases $\boldsymbol{r} = (50, 50, 50)^{\mathsf{T}}$ and $(10, 50, 25)^{\mathsf{T}}$. The performance when the change in the subspace is not dramatic ($\boldsymbol{r} = (10, 10, 10)^{\mathsf{T}}$) is also reasonably good. NORST fails to correctly identify the change points.

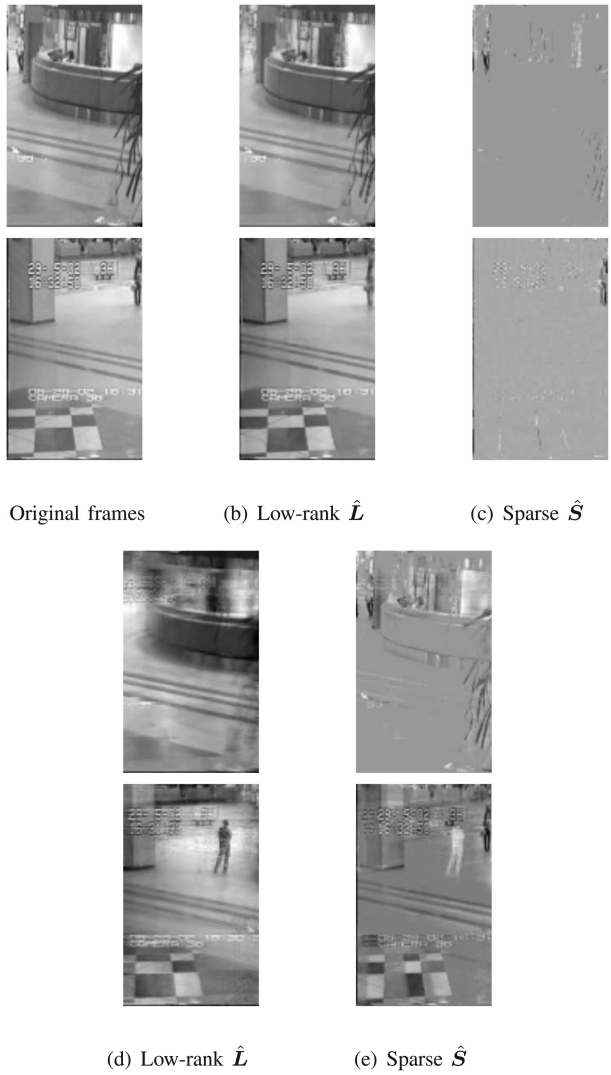
D. Background Subtraction From Surveillance Video

Low-rank subspace tracking is suitable for surveillance videos because successive frames are highly correlated [6]. In a surveillance video the background is generally stable and might change very slowly due to varying illumination. In this paper, experiments on airport and lobby surveillance video data [6], [7], [38] and the UCSD anomaly detection dataset[1] are considered. To evaluate the effectiveness of OMWRPCA-CP for tracking a slowly changing subspace with change points, we pan a "virtual camera" moving from left to right and from right to left through the video. The "virtual camera" moves at a speed of 1 pixel per 10 frames. The original frame has size 176×144, 160×128 and 238×158 for the airport, the lobby, and the UCSD video data, respectively. The virtual camera has the same height and half the width. We stack each frame to a column and feed it to the algorithms. To make the background subtraction task more difficult, we add one change point to both videos where the "virtual camera" jumps instantly from the right-hand most side to the left-hand most side.[2]

We choose the following parameters in the algorithm, $\lambda_1 = 1/\sqrt{200}$, $\lambda_2 = 100/\sqrt{200}$, $n_{\text{win}} = 20$, $n_{\text{burnin}} = 100$, $n_{\text{cp-burnin}} = 100$, $n_{\text{test}} = 300$, $N_{\text{check}} = 3$, $\alpha_{\text{prop}} = 1$, $\alpha = 0.01$, $n_{\text{positive}} = 3$. We set $n_{\text{tol}} = 1000$ in the first two experiments and 0 in the last one. OMWRPCA-CP catches the true change points exactly in all experiments. The recovered low-rank $\hat{\boldsymbol{L}}$ and sparse $\hat{\boldsymbol{S}}$ at two frames before and after the change points are shown in Figs. 8–10 for the airport, lobby, and UCSD video data, respectively. OMWRPCA-CP has a much sharper

[1]downloaded from http://www.svcl.ucsd.edu/projects/anomaly/dataset.htm

[2]we repeat the same video sequence 5 times in UCSD to get a sequence with enough length



(a) Original frames (b) Low-rank $\hat{\boldsymbol{L}}$ (c) Sparse $\hat{\boldsymbol{S}}$

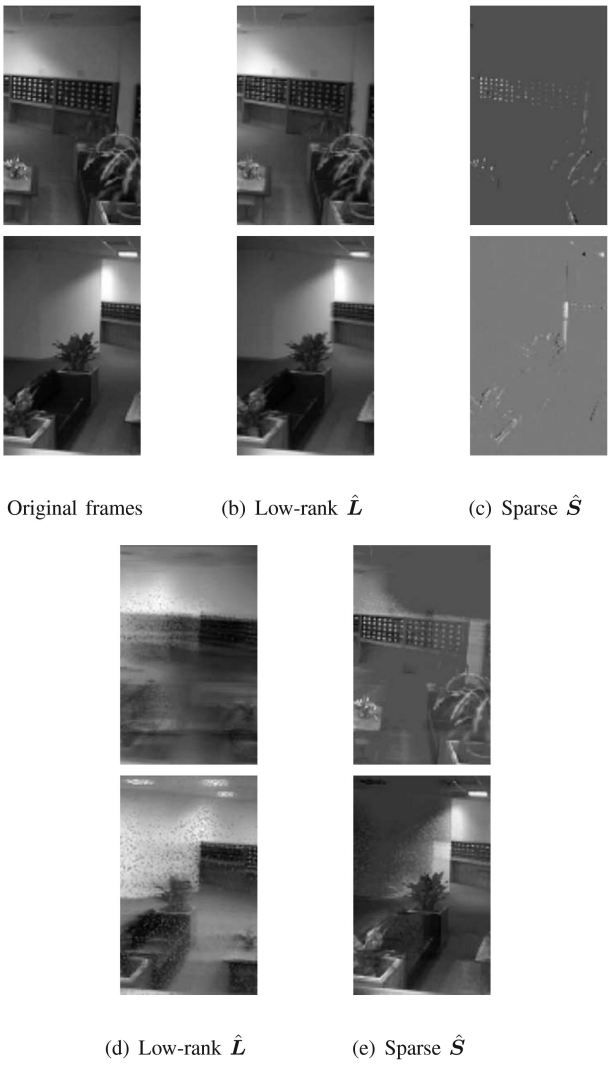
(d) Low-rank $\hat{\boldsymbol{L}}$ (e) Sparse $\hat{\boldsymbol{S}}$

Fig. 8. Background modeling from airport video. The first row shows the result at $t = 878$. The second row shows the result at $t = 882$. The change point is at $t = 880$. (a) Original video $\boldsymbol{M}$. (b)–(c) Low-rank $\hat{\boldsymbol{L}}$ and Sparse $\hat{\boldsymbol{S}}$ obtained from OMWRPCA-CP. (d)–(e) Low-rank $\hat{\boldsymbol{L}}$ and Sparse $\hat{\boldsymbol{S}}$ obtained from STOC-RPCA.

recovered low-rank $\hat{\boldsymbol{L}}$ and sparse $\hat{\boldsymbol{S}}$ compared with STOC_RPCA and NORST.

E. The Influence of Window Size

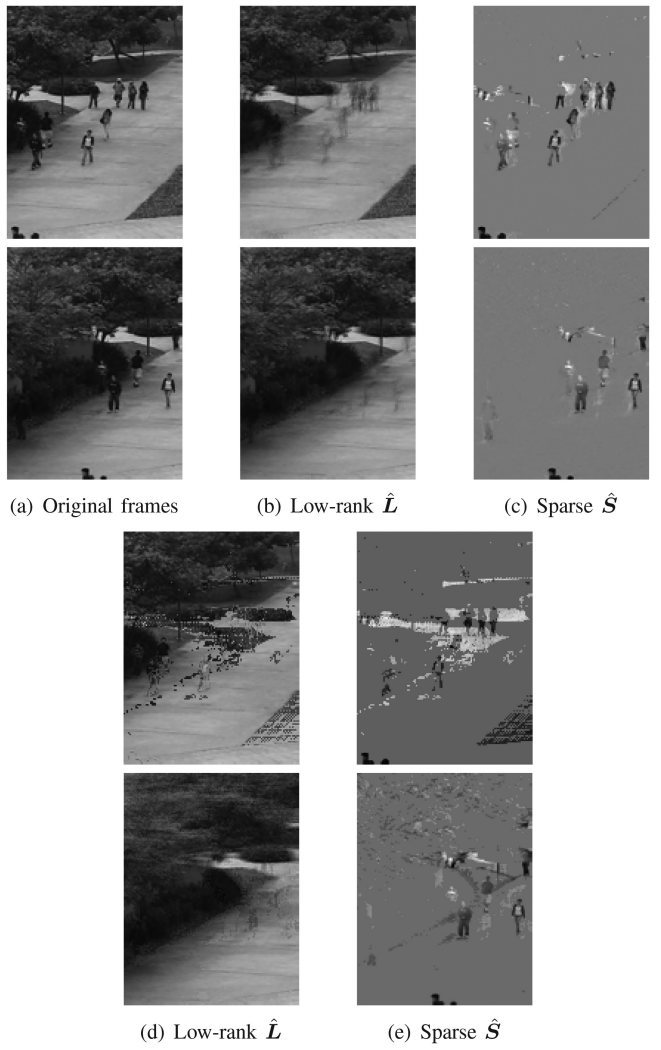
To investigate how the window size affects the final result, we conduct an experiment with different window sizes. Consider data generated as in Section IV-C and set $m = 400, n = 4000, r = [10, 10, 10], \rho = 0.1$. The performance of OMWRPCA-CP with different window sizes is reported in Table I, from which one can observe that the recovery accuracy is quite stable over a relatively large range. As expected, a larger window size implies a longer running time. Besides this key parameter, there are other parameters in OMWRPCA-CP. Generally, the performance is not very sensitive to them, and the users could tune them according to the suggestions in the previous experiments.



(a) Original frames (b) Low-rank $\hat{\boldsymbol{L}}$ (c) Sparse $\hat{\boldsymbol{S}}$

(d) Low-rank $\hat{\boldsymbol{L}}$ (e) Sparse $\hat{\boldsymbol{S}}$

Fig. 9. Background modeling from lobby video. The first row shows the result at $t = 798$. The second row shows the result at $t = 802$. The change point is at $t = 800$. (a) Original video $\boldsymbol{M}$. (b)–(c) Low-rank $\hat{\boldsymbol{L}}$ and Sparse $\hat{\boldsymbol{S}}$ obtained from OMWRPCA-CP. (d)–(e) Low-rank $\hat{\boldsymbol{L}}$ and Sparse $\hat{\boldsymbol{S}}$ obtained from STOC-RPCA.



(a) Original frames (b) Low-rank $\hat{\boldsymbol{L}}$ (c) Sparse $\hat{\boldsymbol{S}}$

(d) Low-rank $\hat{\boldsymbol{L}}$ (e) Sparse $\hat{\boldsymbol{S}}$

Fig. 10. Background modeling from UCSD video. The first row shows the result at $t = 488$. The second row shows the result at $t = 492$. The change point is at $t = 490$. (a) Original video $\boldsymbol{M}$. (b)–(c) Low-rank $\hat{\boldsymbol{L}}$ and Sparse $\hat{\boldsymbol{S}}$ obtained from OMWRPCA-CP. (d)–(e) Low-rank $\hat{\boldsymbol{L}}$ and Sparse $\hat{\boldsymbol{S}}$ obtained from NORST.

TABLE I
PERFORMANCE OF OMWRPCA WITH DIFFERENT n_{win} WHEN $m = 400, n = 3000, r = [10, 10, 10], \rho = 0.1$

wind. size	errorL	errorS	falseS	run time
100	0.2804	0.0088	0.0061	0.2377
200	0.2874	0.0088	0.0059	0.2293
300	0.2799	0.0083	0.0066	0.1973
400	0.2681	0.0079	0.0075	0.3610
500	0.2546	0.0074	0.0087	0.4246
1000	0.1777	0.0049	0.0119	0.9603
1500	0.2375	0.0065	0.0292	1.4749
2000	0.1909	0.0051	0.0388	1.4074

V. CONCLUSION AND DISCUSSION

In this paper we have proposed an online robust PCA algorithm that can track both slowly and abruptly changed subspaces. By embedding hypothesis testing in the algorithm one can discover the exact locations of change points for the underlying low-rank subspaces. In numerical experiments, the proposed algorithm shows promising performance for tracking slowly changing subspaces with change points in an online fashion. The proposed algorithm is promising for real-time video layering where the video sequence is separated into a slowly changing background and a sparse foreground, which has been evaluated in this paper, as well as in failure detection in mechanical systems, intrusion detection in computer networks, and human activity recognition based on sensor data, which are left for future work.

ACKNOWLEDGMENT

The authors would like to thank Mr. P. Narayanamurthy in Iowa State University for helpful discussions on NORST code, K. Walch from SAS Institute, Inc., for helping edit this paper, and also anonymous reviewers for their insightful comments.

REFERENCES

[1] H. Krim and M. Viberg, "Two decades of array signal processing research: The parametric approach," *IEEE Signal Process. Mag.*, vol. 13, no. 4, pp. 67–94, Jul. 1996.

[2] R. Basri and D. W. Jacobs, "Lambertian reflectance and linear subspaces," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 25, no. 2, pp. 218–233, Feb. 2003.

[3] E. J. Candès and B. Recht, "Exact matrix completion via convex optimization," *Found. Comput. Math.*, vol. 9, no. 6, pp. 717–772, 2009.

[4] A. Aravkin, S. Becker, V. Cevher, and P. Olsen, "A variational approach to stable principal component pursuit," in *Proc. 30th Conf. Uncertainty Artif. Intell.*, Arlington, Virginia, United States: AUAI Press, 2014, pp. 32–41.

[5] X. J. Hunt and R. Willett, "Online data thinning via multi-subspace tracking," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 41, no. 5, pp. 1173–1187, May 2019.

[6] E. J. Candès, X. Li, Y. Ma, and J. Wright, "Robust principal component analysis?" *J. ACM*, vol. 58, no. 3, 2011, Art. no. 11.

[7] J. He, L. Balzano, and A. Szlam, "Incremental gradient on the Grassmannian for online foreground and background separation in subsampled video," in *Proc. IEEE Conf. Comput. Vision Pattern Recognit.*, 2012, pp. 1568–1575.

[8] T. Bouwmans and E. H. Zahzah, "Robust PCA via principal component pursuit: A review for a comparative evaluation in video surveillance," *Comput. Vision Image Understand.*, vol. 122, pp. 22–34, 2014.

[9] I. Jolliffe, *Principal Component Analysis*. Berlin, Germany: Springer, 2011.

[10] F. De La Torre and M. J. Black, "A framework for robust subspace learning," *Int. J. Comput. Vision*, vol. 54, no. 1-3, pp. 117–142, 2003.

[11] S. Roweis, "EM algorithms for PCA and SPCA," in *Proc. Adv. Neural Inf. Process. Syst.*, 1998, pp. 626–632.

[12] T. Zhang and G. Lerman, "A novel M-estimator for robust PCA," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 749–808, 2014.

[13] M. McCoy and J. A. Tropp, "Two proposals for robust PCA using semidefinite programming," *Electron. J. Stat.*, vol. 5, pp. 1123–1160, 2011. [Online]. Available: http://dx.doi.org/10.1214/11-EJS636

[14] J. Wright, A. Ganesh, S. Rao, Y. Peng, and Y. Ma, "Robust principal component analysis: Exact recovery of corrupted low-rank matrices via convex optimization," in *Proc. Adv. Neural Inf. Process. Syst.*, 2009, pp. 2080–2088.

[15] T. Bouwmans, A. Sobral, S. Javed, S. K. Jung, and E.-H. Zahzah, "Decomposition into low-rank plus additive matrices for background/foreground separation: A review for a comparative evaluation with a large-scale dataset," *Comput. Sci. Rev.*, vol. 23, pp. 1–71, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1574013715300459

[16] X. Li and J. Haupt, "Identifying outliers in large matrices via randomized adaptive compressive sampling," *IEEE Trans. Signal Process.*, vol. 63, no. 7, pp. 1792–1807, Apr. 2015.

[17] M. Rahmani and G. K. Atia, "High dimensional low rank plus sparse matrix decomposition," *IEEE Trans. Signal Process.*, vol. 65, no. 8, pp. 2004–2019, Apr. 2017.

[18] N. Kwak, "Principal component analysis based on l1-norm maximization," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 30, no. 9, pp. 1672–1680, Sep. 2008.

[19] F. Nie, H. Huang, C. Ding, D. Luo, and H. Wang, "Robust principal component analysis with non-greedy ℓ_1-norm maximization," in *Proc. Int. Joint Conf. Artif. Intell.*, Jul. 2011, pp. 1433–1438.

[20] Y. Liu and D. A. Pados, "Compressed-sensed-domain l1-PCA video surveillance," *IEEE Trans. Multimedia*, vol. 18, no. 3, pp. 351–363, Mar. 2016.

[21] D. Hsu, S. M. Kakade, and T. Zhang, "Robust matrix decomposition with sparse corruptions," *IEEE Trans. Inf. Theory*, vol. 57, no. 11, pp. 7221–7234, Nov. 2011.

[22] Z. Zhou, X. Li, J. Wright, E. Cands, and Y. Ma, "Stable principal component pursuit," in *Proc. IEEE Int. Symp. Inf. Theory*, Jun. 2010, pp. 1518–1522.

[23] H. Xu, C. Caramanis, and S. Sanghavi, "Robust PCA via outlier pursuit," *IEEE Trans. Inf. Theory*, vol. 58, no. 5, pp. 3047–3064, May 2012.

[24] H. Lee, H. Kim, and J. I. Kim, "Background subtraction using background sets with image- and color-space reduction," *IEEE Trans. Multimedia*, vol. 18, no. 10, pp. 2093–2103, Oct. 2016.

[25] J. Tian, Z. Han, W. Ren, X. Chen, and Y. Tang, "Snowflake removal for videos via global and local low-rank decomposition," *IEEE Trans. Multimedia*, vol. 20, no. 10, pp. 2659–2669, Oct. 2018.

[26] J. He, L. Balzano, and J. Lui, "Online robust subspace tracking from partial information," 2011, *arXiv:1109.3827*.

[27] C. Qiu, N. Vaswani, B. Lois, and L. Hogben, "Recursive robust PCA or recursive sparse recovery in large but structured noise," *IEEE Trans. Inf. Theory*, vol. 60, no. 8, pp. 5007–5039, Aug. 2014.

[28] H. Guo, C. Qiu, and N. Vaswani, "An online algorithm for separating sparse and low-dimensional signal sequences from their sum," *IEEE Trans. Signal Process.*, vol. 62, no. 16, pp. 4284–4297, Aug. 2014.

[29] P. Narayanamurthy and N. Vaswani, "Provable dynamic robust PCA or robust subspace tracking," in *Proc. IEEE Int. Symp. Inf. Theory*, 2018, pp. 376–380.

[30] J. Feng, H. Xu, and S. Yan, "Online robust PCA via stochastic optimization," in *Proc. Adv. Neural Inf. Process. Syst.*, 2013, pp. 404–412. [Online]. Available: http://papers.nips.cc/paper/5131-online-robust-pca-via-stochastic-optimization.pdf

[31] L. Balzano, R. Nowak, and B. Recht, "Online identification and tracking of subspaces from highly incomplete information," in *Proc. 48th Annu. Allerton Conf. Commun. Control, Comput.*, 2010, pp. 704–711.

[32] G. Liu *et al.*, "Robust recovery of subspace structures by low-rank representation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 1, pp. 171–184, Jan. 2013.

[33] G. Liu, Q. Liu, and P. Li, "Blessing of dimensionality: Recovering mixture data via dictionary pursuit," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 1, pp. 47–60, Jan. 2017.

[34] P. Narayanamurthy and N. Vaswani, "Nearly optimal robust subspace tracking," in *Proc. 35th Int. Conf. Mach. Learn.*, Jul. 2018, pp. 3701–3709.

[35] Z. Lin *et al.*, "Fast convex optimization algorithms for exact recovery of a corrupted low-rank matrix," *Proc. Int. Workshop Comput. Adv. Multi-Sensor Adaptive Process.*, vol. 61, 2009.

[36] Z. Lin, M. Chen, and Y. Ma, "The augmented lagrange multiplier method for exact recovery of corrupted low-rank matrices," 2010, *arXiv:1009.5055*.

[37] B. Recht, M. Fazel, and P. A. Parrilo, "Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization," *SIAM Rev.*, vol. 52, no. 3, pp. 471–501, 2010.

[38] L. Li, W. Huang, I. Y.-H. Gu, and Q. Tian, "Statistical modeling of complex backgrounds for foreground object detection," *IEEE Trans. Image Process.*, vol. 13, no. 11, pp. 1459–1472, Nov. 2004.