Online Robust Principal Component Analysis with Change Point Detection

Wei Xiao[†][*], Xiaolin Huang[‡], Jorge Silva[†], Saba Emrani[†] and Arin Chaudhuri[†]

[†]*SAS Institute, Cary, NC USA*

[‡]*Shanghai Jiao Tong University, Shanghai China*

wei.xiao@sas.com   xiaolinhuang@sjtu.edu.cn   jorge.silva@sas.com   saba.emrani@sas.com   arin.chaudhuri@sas.com

Abstract

Robust PCA methods are typically batch algorithms which requires loading all observations into memory before processing. This makes them inefficient to process big data. In this paper, we develop an efficient online robust principal component methods, namely online moving window robust principal component analysis (OMWRPCA). Unlike existing algorithms, OMWRPCA can successfully track not only slowly changing subspace but also abruptly changed subspace. By embedding hypothesis testing into the algorithm, OMWRPCA can detect change points of the underlying subspaces. Extensive simulation studies demonstrate the superior performance of OMWRPCA comparing with other state-of-art approach. We also apply the algorithm for real-time background subtraction of surveillance video.

1   Introduction

Low-rank subspace models are powerful tools to analyze high-dimensional data from dynamical systems. Applications include tracking in radar and sonar [1], face recognition [2], recommender system [3], cloud removal in satellite images [4], anomaly detection [5] and background subtraction for surveillance video [6, 7, 8]. Principal component analysis (PCA) is the most widely used tool to obtain a low-rank subspace from high dimensional data. For a given rank r, PCA finds the r dimensional linear subspace which minimizes the square error loss between the data vector and its projection onto that subspace. Although PCA is widely used, it has the drawback that it is not robust against outliers. Even with one grossly corrupted entry, the low-rank subspace 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 as outliers are often observed in the modern world's massive

---

[*]Corresponding author.

data. For example, data collected through sensors, cameras and websites are often very noisy and contains error entries or outliers.

Various versions of robust PCA have been developed in the past few decades, including [9, 6, 10, 11, 12]. Among them, the Robust PCA based on Principal Component Pursuit (RPCA-PCP) [6, 13] is the most promising one, as it has both good practical performance and strong theoretical performance guarantees. A comprehensive review of the application of Robust PCA based on Principal Component Pursuit in surveillance video can be found in [8]. RPCA-PCP decomposes the observed matrix into a low-rank matrix and a sparse matrix by solving Principal Component Pursuit. Under mild condition, both the low-rank matrix and the sparse matrix can be recovered exactly.

Most robust PCA methods including RPCA-PCP are implemented in the batch manner. Batch algorithms not only require huge amount of memory storage as all observed data needs to be stored in the memory before processing, but also become too slow to process big data. An online version of robust PCA method is highly necessary to process incremental dataset, where the tracked subspace can be updated whenever a new sample is received. The online version of robust PCA can be applied to video analysis, change point detection, abnormal events detection and network monitoring.

There are mainly three versions of online robust PCA proposed in the literature. They are Grassmannian Robust Adaptive Subspace Tracking Algorithm (GRASTA) [14], Recursive Projected Compress Sensing (ReProCS) [15, 16] and Online Robust PCA via Stochastic Optimization (RPCA-STOC) [17]. GRASTA applies incremental gradient descent method on Grassmannian manifold to do robust PCA. It is built on Grassmannian Rank-One Update Subspace Estimation (GROUSE) [18], a widely used online subspace tracking algirithm. GRASTA uses the ℓ_1 cost function to replace the ℓ_2 cost function in GROUSE. In each step, GRASTA updates the subspace with gradient descent. ReProCS is designed mainly to solve the foreground and background separation problem for video sequence. ReProCS recursively finds the low-rank matrix and the sparse matrix through four steps: 1. perpendicular projection of observed vector to low-rank subspace; 2. sparse recovery of sparse error through optimization; 3. recover low-rank observation; 4. update low-rank subspace. Both GRASTA and Prac-ReProCS can only handle slowly changing subspaces. RPCA-STOC is based on stochastic optimization for a close reformulation of the Robust PCA based on Principal

Component Pursuit [17]. In particular, RPCA-STOC splits the nuclear norm of the low-rank matrix as the sum of Frobenius norm of two matrices, and iteratively updates the low-rank subspace and finds the sparse vector. RPCA-STOC works only with stable subspace.

Most of the previously developed online robust PCA algorithms only deal with stable subspaces or slowly changing subspaces. However, in practice, the low-rank subspace may change suddenly, and the corresponding change points are usually of great interests. 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. Accurate identification of the change point is useful for failure prediction. Some other applications include human activity recognition, intrusion detection in computer networks, etc. In this paper, we propose an efficient online robust principal component methods, namely online moving window robust principal component analysis (OMWRPCA), to handle both slowly changing and abruptly changed subspace. The method, which embeds hypothesis testing into online robust PCA framework, can accurately identify the change points of underlying subspace, and simultaneously estimate the low-rank subspace and sparse error. To the limit of the authors' knowledge, OMWRPCA is the first algorithm developed to be 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 changed subspaces.

The remainder of this paper is organized as follows. Section 2 gives the problem definition and introduces the related methods. The OMWRPCA algorithm is developed in Section 3. In Section 4, we compare OMWRPCA with RPCA-STOC via extensive numerial experiments. We also demonstrate the OMWRPCA algorithm with an application to a real-world video surveillance data. Section 5 concludes the paper and gives some discussion.

2   Problem Definition and Related Works

2.1   Notation and Problem Definition

We use bold letters to denote vectors and matrices. We use $\|\boldsymbol{a}\|_1$ and $\|\boldsymbol{a}\|_2$ to represent the ℓ_1-norm and ℓ_2-norm of vector $\boldsymbol{a}$, respectively. For an arbitrary real matrix $\boldsymbol{A}$, $\|\boldsymbol{A}\|_F$ denotes the Frobenius norm of matrix $\boldsymbol{A}$, $\|\boldsymbol{A}\|_*$ denotes the nuclear norm of matrix $\boldsymbol{A}$ (sum of all

singular values) and $\|A\|_1$ stands for the ℓ_1-norm of matrix A where we treat A as a vector.

Let T denote the number of observed samples, and t is the index of the sample instance. We assume that our inputs are streaming observations $m_t \in \mathbb{R}^m$, $t = 1, \ldots, T$, which can be decomposed as two parts $m_t = l_t + s_t$. The first part l_t is a vector from a low-rank subspace U_t, and the second part s_t is a sparse error with support size c_t. Here c_t represents the number of nonzero elements of s_t, and $c_t = \sum_{i=1}^m I_{s_t[i] \neq 0}$. The underlying subspace U_t may or may not change with time t. When 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 s_t satisfy the sparsity assumption, i.e., $c_t \ll m$ for all $t = 1, \ldots, T$. We use $M_t = [m_1, \ldots, m_t] \in \mathbb{R}^{m \times t}$ to denote the matrix of observed samples until time t. Let M, L, S denotes M_T, L_T, S_T respectively.

2.2 Robust Principal Component Analysis

Robust PCA based on Principal Component Pursuit (RPCA-PCP) [6, 13] is the most popular RPCA algorithm which decomposes the observed matrix M into a low-rank matrix L and a sparse matrix S by solving Principal Component Pursuit:

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

Theoretically, we can prove that L and S can be recovered exactly if (a) the rank of L and the support size of S are small enough; (b) L is "dense"; (c) any element in sparse matrix S is nonzero with probability ρ and 0 with probability $1 - \rho$ independent over time [6, 14]. RPCA-PCP is often applied to estimate the low-rank matrix which approximates 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) [19] and Augmented Lagrangian Multiplier (ALM) [20]. Bases on the experiment of [6], ALM achieves higher accuracy than APG, in fewer iterations. We implemented ALM in this paper. Let $\mathcal{S}_\tau$ denote the shrinkage operator $S_\tau[x] = \text{sgn}(x) \max(|x| - \tau, 0)$, and extend it to matrices by applying it to each element. Let $\mathcal{D}_\tau(X)$ denote the singular value thresholding operator given by $\mathcal{D}_\tau(X) = U \mathcal{S}_\tau(\Sigma) V^*$, where $X = U \Sigma V^*$ is any

singular value decomposition. ALM is outlined in Algorithm 1.

Algorithm 1: Principal Component Pursuit by Augmented Largrangian Multiplier [20, 6]

---

1 Input: $\boldsymbol{M}$ (observed data), $\lambda, \mu \in \mathbb{R}$ (regularization parameters)
2 Initialize: $\boldsymbol{S}_0 = \boldsymbol{Y}_0 = \boldsymbol{0}$.
3 **while** *not converged* **do**
4 $\quad$ 1) Compute $\boldsymbol{L}^{(k+1)} \leftarrow \mathcal{D}_{\mu^{-1}}(\boldsymbol{M} - \boldsymbol{S}^{(k)} + \mu^{-1}\boldsymbol{Y}^{(k)})$;
5 $\quad$ 2) Compute $\boldsymbol{S}^{(k+1)} \leftarrow \mathcal{S}_{\lambda\mu^{-1}}(\boldsymbol{M} - \boldsymbol{L}^{(k+1)} + \mu^{-1}\boldsymbol{Y}^{k})$;
6 $\quad$ 3) Compute $\boldsymbol{Y}^{(k+1)} \leftarrow \boldsymbol{Y}^{(k)} + \mu(\boldsymbol{M} - \boldsymbol{L}^{(k+1)} - \boldsymbol{S}^{(k+1)})$;
7 **return** $\boldsymbol{L}$ *(low-rank data matrix),* $\boldsymbol{S}$ *(sparse noise matrix)*

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

2.3 Moving Window Robust Principal Component Analysis

In practice, the underlying subspace can change over time. This makes the RPCA-PCP infeasible to do after some time as the rank of matrix $\boldsymbol{L}$ (includes all observed samples) will increase over time. One solution is Moving Window Robust Principal Component Analysis (MWRPCA) which applies a sliding window approach to the original data and iteratively computes 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. It can do subspace tracking on both slowly changing subspace and abruptly changed subspace. However, MWRPCA often becomes too slow to deal with real-world big data problem. For example, suppose we want to track the subspace on an $\boldsymbol{M}$ matrix with the size 200×10000. If we chose a window size $n_{\text{win}} = 1000$, we need to run RPCA-PCP 9000 times on matrices with size 200×1000, which is computationally too expensive.

2.4 Robust PCA via Stochastic Optimization

Feng et al. [17] proposed an online algorithm to solve robust PCA based on Principal Component Pursuit. It starts by minimizing the following loss function

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

where λ_1, λ_2 are tuning parameters. The main difficulty in developing an online algorithm to solve the above equation is that the nuclear norm couples all the samples tightly. Feng

et al. [17] solve the problem by using an equivalent form of the nuclear norm following [21], which states that the nuclear norm for a matrix $\boldsymbol{L}$ whose rank is upper bounded by r has an equivalent form

$$\|\boldsymbol{L}\|_* = \inf_{\boldsymbol{U} \in \mathbb{R}^{m \times r}, \boldsymbol{V} \in \mathbb{R}^{r \times n}} \left\{ \frac{1}{2} \|\boldsymbol{U}\|_F^2 + \frac{1}{2} \|\boldsymbol{V}\|_F^2 : \boldsymbol{L} = \boldsymbol{U}\boldsymbol{V} \right\}. \tag{3}$$

Substituting $\boldsymbol{L}$ by $\boldsymbol{U}\boldsymbol{V}$ and plugging (3) into (2), we have

$$\min_{\boldsymbol{U} \in \mathbb{R}^{m \times r}, \boldsymbol{V} \in \mathbb{R}^{n \times r}, \boldsymbol{S}} \frac{1}{2} \|\boldsymbol{M} - \boldsymbol{U}\boldsymbol{V} - \boldsymbol{S}\|_F^2 + \frac{\lambda_1}{2} (\|\boldsymbol{U}\|_F^2 + \|\boldsymbol{V}\|_F^2) + \lambda_2 \|\boldsymbol{S}\|_1, \tag{4}$$

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. They then propose their RPCA-STOC algorithm which minimizes the empirical version of loss function (4) and processes one sample per time instance [17]. Given a finite set of samples $\boldsymbol{M}_t = [\boldsymbol{m}_1, \ldots, \boldsymbol{m}_t] \in \mathbb{R}^{m \times t}$, the empirical version of loss function (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 function for each sample is defined as

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

Fixing $\boldsymbol{U}$ as $\boldsymbol{U}_{t-1}$, $\boldsymbol{v}_t$ and $\boldsymbol{s}_t$ can be obtained by solving the optimization problem

$$(\boldsymbol{v}_t, \boldsymbol{s}_t) = \operatorname*{argmin}_{\boldsymbol{v}, \boldsymbol{s}} \frac{1}{2} \|\boldsymbol{m}_t - \boldsymbol{U}\boldsymbol{v} - \boldsymbol{s}\|_2^2 + \frac{\lambda_1}{2} \|\boldsymbol{v}\|_2^2 + \lambda_2 \|\boldsymbol{s}\|_1.$$

Assuming $\{\boldsymbol{v}_i, \boldsymbol{s}_i\}_{i=1}^{t}$ are known, the basis $\boldsymbol{U}_t$ can be updated by minimizing the following function

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

which gives the explicit solution

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

In practice, $\boldsymbol{U}_t$ can be quickly updated by block-coordinate descent with warn restarts [17]. The RPCA-STOC algorithm is summarized in Algorithm 2.

Algorithm 2: RPCA-STOC algorithm [17]

1 Input: $\{\boldsymbol{m}_1, \ldots, \boldsymbol{m}_T\}$(observed data which are revealed sequentially),
$\lambda_1, \lambda_2 \in \mathbb{R}$(tuning parameters), T(number of iterations);

2 Initialize: $\boldsymbol{U}_0 = \boldsymbol{0}^{m \times r}$, $A_0 = \boldsymbol{0}^{r \times r}$, $B_0 = \boldsymbol{0}^{m \times r}$;

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

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

5   2) Project the new sample:

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

6   3) $A_t \leftarrow A_{t-1} + \boldsymbol{v}_t\boldsymbol{v}_t^{\mathsf{T}}$, $B_t \leftarrow B_{t-1} + (\boldsymbol{m}_t - \boldsymbol{s}_t)\boldsymbol{v}_t^{\mathsf{T}}$.

7   4) Compute $\boldsymbol{U}_t$ with $\boldsymbol{U}_{t-1}$ as warm restart using Algorithm 3:

$$\boldsymbol{U}_t \triangleq \operatorname{argmin} \frac{1}{2} \operatorname{Tr}[\boldsymbol{U}^{\mathsf{T}}(A_t + \lambda_1 I)\boldsymbol{U}] - \operatorname{Tr}(\boldsymbol{U}^{\mathsf{T}}B_t).$$

8 return $\boldsymbol{L}_T = \{\boldsymbol{U}_1\boldsymbol{v}_1, \ldots, \boldsymbol{U}_T\boldsymbol{v}_T\}$ *(low-rank data matrix)*, $\boldsymbol{S}_T = \{\boldsymbol{s}_1, \ldots, \boldsymbol{s}_T\}$ *(sparse noise matrix)*

Algorithm 3: Fast Basis Update[17]

1 Input: $\boldsymbol{U} = [\boldsymbol{u}_1, \ldots, \boldsymbol{u}_r] \in \mathbb{R}^{m \times r}$, $A = [\boldsymbol{a}_1, \ldots, \boldsymbol{a}_r] \in \mathbb{R}^{r \times r}$, and
$B = [\boldsymbol{b}_1, \ldots, \boldsymbol{b}_r] \in \mathbb{R}^{r \times r}$; $\tilde{A} \leftarrow A + \lambda_1 I$.

2 for *j=1 to r* **do**

3

$$\tilde{\boldsymbol{u}}_j \leftarrow \frac{1}{\tilde{A}[j,j]}(\boldsymbol{b}_j - \boldsymbol{U}\tilde{\boldsymbol{a}}_j) + \boldsymbol{u}_j,$$

$$\boldsymbol{u}_j \leftarrow \frac{1}{\max(\|\tilde{\boldsymbol{u}}_j\|_2, 1)}\tilde{\boldsymbol{u}}_j.$$

4 return U

---

3 Online Moving Window RPCA

3.1 Basic Algorithm

One limitation of RPCA-STOC is that the method assumes a stable subspace, which is generally a too restrict assumption for applications such as failure detection in mechanical systems, intrusion detection in computer networks, fraud detection in financial transaction and background subtraction for surveillance video. At time t, RPCA-STOC updates the basis of subspace by minimizing an empirical loss which involves all previously observed

samples with equal weights. Thus, the subspace we obtain at time t can be viewed as an "average" subspace of observations from time 1 to t. This is clearly not desirable if the underlying subspace is changing over time. We propose a new method which combines the idea of moving window RPCA and RPCA-STOC to track changing subspace. At time t, the proposed method updates $\boldsymbol{U}_t$ 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

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

We call our new method Online Moving Window RPCA (OMWRPCA). The biggest advantage of OMWRPCA comparing with RPCA-STOC is that OMWRPCA can quickly update the subspace when the underlying subspace is changing. It has two other minor differences in the details of implementation compared with RPCA-STOC. In RPCA-STOC, we assume the dimension of subspace (r) is known. In OMWRPCA, we estimate it by computing batch RPCA-PCP with burn-in samples. The size of the burn-in samples n_{burnin} is another user specified parameter. We also use the estimated $\boldsymbol{U}$ from burn-in samples as our initial guess of $\boldsymbol{U}$ in OMWRPCA, while $\boldsymbol{U}$ is initialized as zero matrix in RPCA-STOC. Specifically, in the initialization step, batch RPCA-PCP is computed on burn-in samples to get rank r, basis of subspace $\boldsymbol{U}_0$, A_0 and B_0. For simplicity, we also require $n_{\mathrm{win}} \leq n_{\mathrm{burnin}}$. Details are given as follows.

1. Compute RPCA-PCP on burn-in samples $\boldsymbol{M}^b$, and we have $\boldsymbol{M}^b = \boldsymbol{L}^b + \boldsymbol{S}^b$, where $\boldsymbol{L}^b = [\boldsymbol{l}_{-n_{\mathrm{burnin}}+1}, \ldots, \boldsymbol{l}_0]$ and $\boldsymbol{S}^b = [\boldsymbol{s}_{-n_{\mathrm{burnin}}+1}, \ldots, \boldsymbol{s}_0]$.

2. From SVD, we have $\boldsymbol{L}^b = \hat{\boldsymbol{U}}\hat{\boldsymbol{\Sigma}}\hat{\boldsymbol{V}}$, where $\hat{\boldsymbol{U}} \in \mathbb{R}^{m \times r}$, $\hat{\boldsymbol{\Sigma}} \in \mathbb{R}^{r \times r}$, and $\hat{\boldsymbol{V}} \in \mathbb{R}^{r \times n_{\mathrm{burnin}}}$.

3. $\boldsymbol{U}_0 = \hat{\boldsymbol{U}}\hat{\boldsymbol{\Sigma}}^{1/2} \in \mathbb{R}^{m \times r}$, $A_0 = \sum_{i=-(n_{\mathrm{win}}-1)}^{0} \boldsymbol{v}_i\boldsymbol{v}_i^{\mathsf{T}} \in \mathbb{R}^{r \times r}$ and $B_0 = \sum_{i=-(n_{\mathrm{win}}-1)}^{0} (\boldsymbol{m}_i - \boldsymbol{s}_i)\boldsymbol{v}_i^{\mathsf{T}} \in \mathbb{R}^{m \times r}$;

OMWRPCA is summarized in Algorithm 4.

Algorithm 4: Online Moving Window RPCA

---

1 Input: $\{\boldsymbol{m}_1, \ldots, \boldsymbol{m}_T\}$ (observed data which are revealed sequentially),
$\lambda_1, \lambda_2 \in \mathbb{R}$ (regularization parameters), T (number of iterations);
$\boldsymbol{M}^b = [\boldsymbol{m}_{-(n_{\mathrm{burnin}}-1)}, \ldots, \boldsymbol{m}_0]$ (burn-in samples);

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

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

4 $\quad$ 1) Reveal the sample $\boldsymbol{m}_t$.

5 $\quad$ 2) Project the new sample:

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

6 $\quad$ 3) $A_t \leftarrow A_{t-1} + \boldsymbol{v}_t\boldsymbol{v}_t^\mathsf{T} - \boldsymbol{v}_{t-n_{\mathrm{win}}}\boldsymbol{v}_{t-n_{\mathrm{win}}}$,
$\quad\quad B_t \leftarrow B_{t-1} + (\boldsymbol{m}_t - \boldsymbol{s}_t)\boldsymbol{v}_t^\mathsf{T} - (\boldsymbol{m}_{t-n_{\mathrm{win}}} - \boldsymbol{s}_{t-n_{\mathrm{win}}})\boldsymbol{v}_{t-n_{\mathrm{win}}}^\mathsf{T}$.

7 $\quad$ 4) Compute $\boldsymbol{U}_t$ with $\boldsymbol{U}_{t-1}$ as warm restart using Algorithm 3:

$$\boldsymbol{U}_t \triangleq \operatorname{argmin} \frac{1}{2}\operatorname{Tr}[\boldsymbol{U}^\mathsf{T}(A_t + \lambda_1 I)\boldsymbol{U}] - \operatorname{Tr}(\boldsymbol{U}^\mathsf{T}B_t).$$

8 return $\boldsymbol{L}_T = \{\boldsymbol{U}_1\boldsymbol{v}_1, \ldots, \boldsymbol{U}_T\boldsymbol{v}_T\}$ *(low-rank data matrix)*, $\boldsymbol{S}_T = \{\boldsymbol{s}_1, \ldots, \boldsymbol{s}_T\}$ *(sparse noise matrix)*

3.2 Change Point Detection in Online Moving Window RPCA with Hypothesis Testing

One limitation of the basic OMWRPCA algorithm is that it can only deal with slowly changing subspace. When the 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 subspace, the subspace may not be able to be updated in an online fashion or it may take a while to finish the updates. One example is that in the case the new subspace is in higher dimension than the original subspace, basic OMWRPCA algorithm can never be able to update the subspace correctly as the rank of the subspace is fixed to a constant. Similar drawback is shared by the majority of the other online RPCA algorithms which have been previously developed. Furthermore, an online RPCA algorithm which can identify the change point is very desirable since the change point detection is very crucial for subsequent identification, moreover, we sometimes are more interested to pinpoint the change points than to estimate the underlying subspace. Thus, we propose another variant

9

of our OMWRPCA algorithm to simultaneously detect change points and compute RPCA in an online fashion. The algorithm is robust to dramatic subspace change. We achieve this goal by embedding hypothesis testing into the original OMWRPCA algorithm. We call the new algorithm OMWRPCA-CP.

OMWRPCA-CP is based on a very simple yet important observation. That is when the new observation can not be modeled well with the current subspace, we will often result in an estimated sparse vector $\hat{s}_t$ with an abnormally large support size from the OMWR-PCA algorithm. So by monitoring the support size of the sparse vector computed from the OMWRPCA algorithm, we can identify the change point of subspace which is usually the start point of level increases in the time series of $\hat{c}_t$. Figure 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 index -200 and 0, and two change points are at $t = 1000$ and 2000. The time line is separated to three pieces with change points. In each piece, we have a constant underlying subspace, and the rank is given in r. The elements of the true sparse vector have a probability of ρ to be nonzero, and are independently generated. The dimension of the samples is 400. We choose window size 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 $\rho = 0.01$ case and $\rho = 0.1$ case, respectively. For the upper two cases, $\hat{c}_t$ blows up after the first change point and never return to the normal range. This is because after we fit RPCA-PCP on burn-in samples, the dimension of U_t is fixed to 10. Thus, the estimated subspace U_t can never approximate well the true subspace in later two pieces of the time line as the true subspaces have larger dimensions. On the other hand, for the lower two cases, $\hat{c}_t$ 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 give an informal proof of why the above phenomenon happens. Assume we are at time point t and the observation is $m_t = U_t v_t + s_t$. We estimate $(\hat{v}_t, \hat{s}_t)$ by computing

$$(v_t, s_t) = \arg\min \frac{1}{2} \|m_t - \hat{U}_{t-1} v - s\|_2^2 + \frac{\lambda_1}{2} \|v\|_2^2 + \lambda_2 \|s\|_1,$$

where $\hat{U}_{t-1}$ is the estimated subspace at time point $t-1$. The above optimization problem does not have an explicit solution and can be iteratively solved by fixing s_t, solving v_t and fixing v_t, solving s_t. We here approximate the solution with one step up-
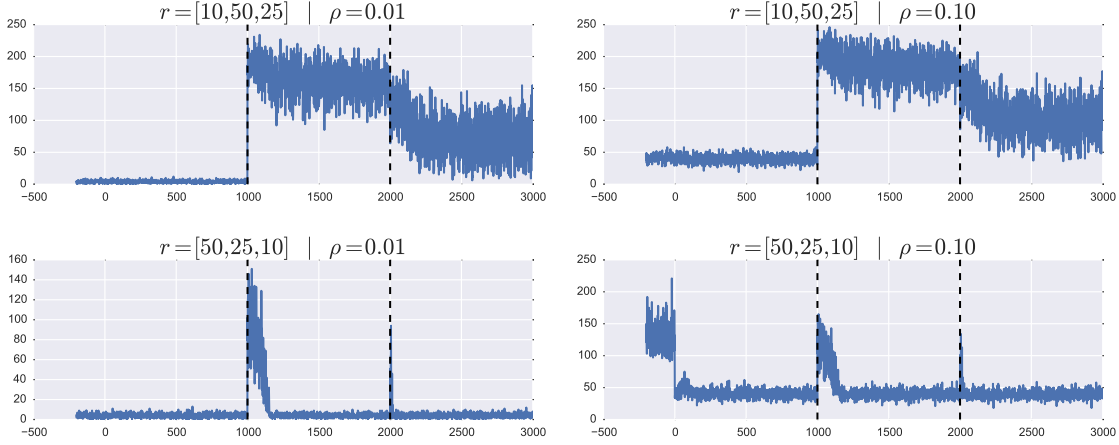
10

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

date from a good initial point $\hat{\boldsymbol{s}}_t = \boldsymbol{s}_t$. We have $\hat{\boldsymbol{v}}_t = (\hat{\boldsymbol{U}}_{t-1}^\mathsf{T}\hat{\boldsymbol{U}}_{t-1} + \lambda_1 I)^{-1}\hat{\boldsymbol{U}}_{t-1}^\mathsf{T}\boldsymbol{U}_t\boldsymbol{V}_t$, $\hat{\boldsymbol{s}}_t = \mathcal{S}_{\lambda_2}\left\{\boldsymbol{s}_t + \left[I - \hat{\boldsymbol{U}}_{t-1}(\hat{\boldsymbol{U}}_{t-1}^\mathsf{T}\hat{\boldsymbol{U}}_{t-1} + \lambda_1 I)^{-1}\hat{\boldsymbol{U}}_{t-1}\right]\boldsymbol{U}_t\boldsymbol{v}_t\right\}$. When λ_1 is small, we approximately have $\hat{\boldsymbol{v}}_t = (\hat{\boldsymbol{U}}_{t-1}^\mathsf{T}\hat{\boldsymbol{U}}_{t-1})^{-1}\hat{\boldsymbol{U}}_{t-1}^\mathsf{T}\boldsymbol{U}_t\boldsymbol{V}_t$ 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 in the order of $O\left(1/\sqrt{\max(m, n_{\text{win}})}\right)$, 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 ($\min_{i \in \{i: \boldsymbol{s}_t[i] \neq 0\}} \boldsymbol{s}_t[i] > \lambda_2$), 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 $\max_{i \in \{i: \boldsymbol{s}_t[i] = 0\}}\left(\mathcal{P}_{\hat{\boldsymbol{U}}_{t-1}^\perp}\boldsymbol{U}_t\boldsymbol{v}_t[i]\right) > \lambda_2$. 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.

We develop a change point detection algorithm by monitoring $\hat{c}_t$. The algorithm determines that a change point exists when it finds $\hat{c}_t$ is abnormally high for a while. Specifically, the users can provide two parameters N_{check} and α_{prop}. When in N_{check} consecutive observations, the algorithm finds more than α_{prop} observations with abnormally high $\hat{c}_t$, the algorithm decides that a change point exists in these N_{check} observations and traces back to find the change point. We refer the cases when the underlying estimated subspace approximates the true subspace well as "normal", and "abnormal" otherwise. All the collected information of $\{\hat{c}_j\}_{j=1}^{t-1}$ from the normal period is stored in $H_c \in \mathbb{R}^{m+1}$, where the $(i+1)$-th

11

element of H_c denotes the number of times that $\{\hat{c}_j\}$ equals i. When we compute $\hat{c}_t$ from a new observation $\boldsymbol{m}_t$, based on H_c we can flag this observation as a normal one or an abnormal one via hypothesis testing. We compute p-value $p = \sum_{i=\hat{c}_t+1}^{m+1} H_t[i] / \sum_{i=0}^{m+1} H_t[i]$, which is the probability that we observe a sparse vector with at least as many nonzero elements as the current observation under the hypothesis that this observation is normal. If $p \leq \alpha$, we flag this observation as abnormal. Otherwise, we flag this observation as normal. Here α is a user specified threshold. We find $\alpha = 0.01$ works very well. Algorithm 5 displays OMWRPCA-CP in pseudocode. The key steps are

1. Initialize H_c as a zero vector $\boldsymbol{0}^{m+1}$. Initialize the buffer list for flag B_f and the buffer list for count B_c as empty lists. In practice, B_f and B_c can be implemented as queue structures (first in first out).

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, we wait for the subspace of OMWRPCA algorithm to become stable. H_c, B_f and B_c remain unchanged.

4. For the next n_{test} observations of OMWRPCA, we calculate $\hat{c}_t$ from $\hat{s}_t$, and update H_c ($H_c[\hat{c}_t + 1] \leftarrow H_c[\hat{c}_t + 1] + 1$). B_f and B_c remain unchanged. Hypothesis testing is not done in this stage, as the sample size of the hypothesis test is small ($\sum_{i=0}^{m+1} H_c[i] \leq n_{\text{test}}$) and the test will not be accurate.

5. Now we can do hypothesis testing on $\hat{c}_t$ with H_c. We get the flag f_t for the tth observation, where $f_t = 1$ if the tth observation is abnormal, and $f_t = 0$ if the tth 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. Assuming the number popped from B_c is c, we then update H_c with c ($H_c[c + 1] \leftarrow H_c[c + 1] + 1$). If $n_b < N_{\text{check}} + 1$, we do nothing. Thus, the buffer size n_b is always kept within N_{check}, and when the buffer size reaches N_{check}, it will remain in N_{check}. Buffers B_f contains the flag information of most recent N_{check} observations and can be used to detect 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 $n_{\text{abnormal}} \geq \alpha_{\text{prop}} N_{\text{check}}$, we know a change point exists in the most recent N_{check} observations. We then do a simple loop over the list B_f to find the change point, which is detected as the first instance of n_{positive} consecutive abnormal cases. For example, suppose we find that $B_f[i]$ which corresponding to time points $t_0, t_0+1, \ldots, t_0+n_{\text{positive}}-1$ are the first instance of n_{positive} consecutive abnormal cases. The change point is then determined as t_0. Here n_{positive} is another parameter specified by the user. In practice, we find $n_{\text{positive}} = 3$ works well. After the change point is identified, OMWRPCA-CP can restart from the change point.

Good choice of tuning parameters is the key to the success of the proposed algorithms. (λ_1, λ_2) can be chosen based on cross-validation on the order of $O\left(1/\sqrt{\max(m, n_{\text{win}})}\right)$. 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, and not too small to avoid generating false alarms. α_{prop} can be chosen based on user's prior-knowledge. Sometimes the assumption that the support size of sparse vector s_t remains stable and much smaller than m is violated in real world data. For example, in video surveillance data, the foreground may contain significant variations over time. In this case, we can add one additional positive tuning parameter n_{tol} and change the formula of p-value to $p = \sum_{i=\hat{c}_t-n_{\text{tol}}+1}^{m+1} H_t[i] / \sum_{i=0}^{m+1} H_t[i]$. This change can make the hypothesis test more conservative, and force the algorithm not detecting too many change points (false alarms).

It is easy to prove that OMWRPCA and OMWRPCA-CP (ignore 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 [17]. In contrast, RPCA-PCP computes an SVD and a thresholding operation in each iteration with the computational complexity $O(nm^2)$. Based on the experiment of [17], the proposed method are also more efficient than other online RPCA algorithm such as GRASTA. The memory costs of OMWRPCA and OMWRPCA-CP are $O(mr)$ and $O(mr + N_{\text{check}})$ respectively. In contrast, the memory cost of RPCA-PCP is $O(mn)$, where $n \gg r$. Thus, the proposed methods are well suitable to process big data.

Last, current version of OMWRPCA-CP does hypothesis tests based on all historical information of $\hat{c}_t$. We can easily change H_c to a queue structure and store only recent history of $\hat{c}_t$, where the user can specify how long of the history they would like to trace back. This

change makes the algorithm detect a change point only by comparing with recent history. We do not pursue this variation in this paper.

4   Numerical Experiments and Applications

In this section, we compare OMWRPCA with RPCA-STOC via extensive numerial 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. We implement all algorithms in python and the code is available at Github address `https://github.com/wxiao0421/onlineRPCA.git`.

4.1   Simulation Study 1: stable subspace

The simulation study has a setting similar to [17]. 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. 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 $\boldsymbol{L} = \boldsymbol{UV}$, 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 the $\mathcal{N}(0, 1)$ distribution. Here $\boldsymbol{U}$ is the basis of the constant subspace with dimension r. We fix $T = 5000$ and $m = 400$. A burn-in samples $\boldsymbol{M}^b$ with the size 400×200 is also generated. We have four settings $(r, \rho) = (10, 0.01)$, $(10, 0.1)$, $(50, 0.01)$, and $(50, 0.1)$. For each setting, we run 50 replications. We choose the following parameters in all simulation studies, $\lambda_1 = 1/\sqrt{400}$, $\lambda_2 = 100/\sqrt{400}$, $n_{\text{win}} = 200$, $n_{\text{burnin}} = 200$, $n_{\text{cp-burnin}} = 200$, $n_{\text{test}} = 100$, $N_{\text{check}} = 20$, $\alpha_{\text{prop}} = 0.5$, $\alpha = 0.01$, $n_{\text{positive}} = 3$ and $n_{\text{tol}} = 0$.

We compare three different methods, STOC-RPCA, OMWRPCA and OMWRPCA-CP. Three criteria are applied:

$$\text{ERR}_{\text{L}} = \|\hat{\boldsymbol{L}} - \boldsymbol{L}\|/\|\boldsymbol{L}\|_F;$$
$$\text{ERR}_{\text{S}} = \|\hat{\boldsymbol{S}} - \boldsymbol{S}\|/\|\boldsymbol{S}\|_F;$$
$$\text{F}_{\text{S}} = \#\{(\hat{\boldsymbol{S}} \neq \boldsymbol{0}) \neq (\boldsymbol{S} \neq \boldsymbol{0})\}/(mT).$$

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 correctly identified elements in $\boldsymbol{S}$.

14

Box plots of $\mathrm{ERR_L}$, $\mathrm{ERR_S}$, $\mathrm{F_S}$ and running times are shown in Figure 2. OMWRPCA-CP has the same result as OMWRPCA, and no change point is detected with OMWRPCA-CP in all replications. All three methods STOC-RPCA, OMWRPCA and OMWRPCA-CP has comparable performance on $\mathrm{ERR_S}$ and running times. STOC-RPCA has slightly better performance on $\mathrm{ERR_L}$ when $\rho = 0.1$. OMWRPCA and OMWRPCA-CP has slightly better performance on $\mathrm{F_S}$. All methods take approximately the same amount of time to run, and are all very fast (around 1 minute per replication).
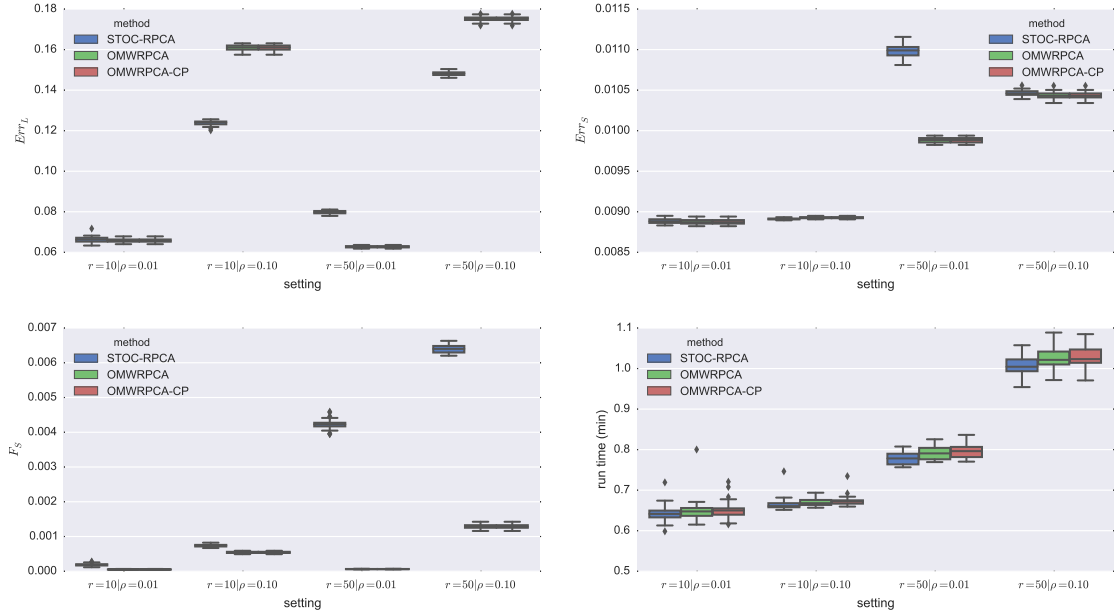
Figure 2: Box plots of $\mathrm{ERR_L}$, $\mathrm{ERR_S}$, $\mathrm{F_S}$ and running times under the simulation study of stable subspace. For each setting, the box plots of STOC-RPCA, OMWRPCA and OMWRPCA-CP are arranged from left to right.

4.2   Simulation Study 2: slowly changing subspace

We adopt almost all setting of Simulation Study 1 except that we make the underlying subspace U change linearly over time. We first generate $U_0 \in \mathbb{R}^{m \times r}$ with i.i.d. samples from the $\mathcal{N}(0,1)$ distribution. We generate burnin samples M^b based on U_0. We make U slowly change over time by adding new matrics $\{\tilde{U}_k\}_{k=1}^{K}$ generated independently with i.i.d. $\mathcal{N}(0,1)$ elements to the first r_0 columns of U, where $\tilde{U}_k \in \mathbb{R}^{m \times r_0}$, $k = 1, \ldots, K$ and $K = T/T_p$. We choose $T_p = 250$. Specifically, for $t = T_p * i + j$, where $i = 0, \ldots, K-1$, $j = 0, \ldots, T_p - 1$, we

have

$$\boldsymbol{U}(t)[:,1:r_0] = \boldsymbol{U}_0[:,1:r_0] + \sum_{1 \leq k \leq i} \tilde{\boldsymbol{U}}_k + \frac{j}{n_p}\tilde{\boldsymbol{U}}_{i+1}, \quad \boldsymbol{U}(t)[:,(r_0+1):r] = \boldsymbol{U}_0[:,(r_0+1):r].$$

We set $r_0 = 5$ in this simulation.

Box plots of $\mathrm{ERR_L}$, $\mathrm{ERR_S}$, $\mathrm{F_S}$ and running times are shown in Figure 3. OMWRPCA-CP has the same result as OMWRPCA, and no change point is detected with OMWRPCA-CP in all replications. OMWRPCA and OMWRPCA-CP have better performance on all three criteria $\mathrm{ERR_L}$, $\mathrm{ERR_S}$, $\mathrm{F_S}$ comparing with STOC-RPCA as we pointed out before that STOC-RPCA is not able to efficiently track changing subspace. In Figure 4, we plot the average of $\mathrm{ERR_L}$ across all replications as a function of number of observations to investigate the progress of performance across different methods. It shows that the performance of STOC-RPCA deteriorates over time, while OMWRPCA and OMWRPCA-CP's performance is quite stable.
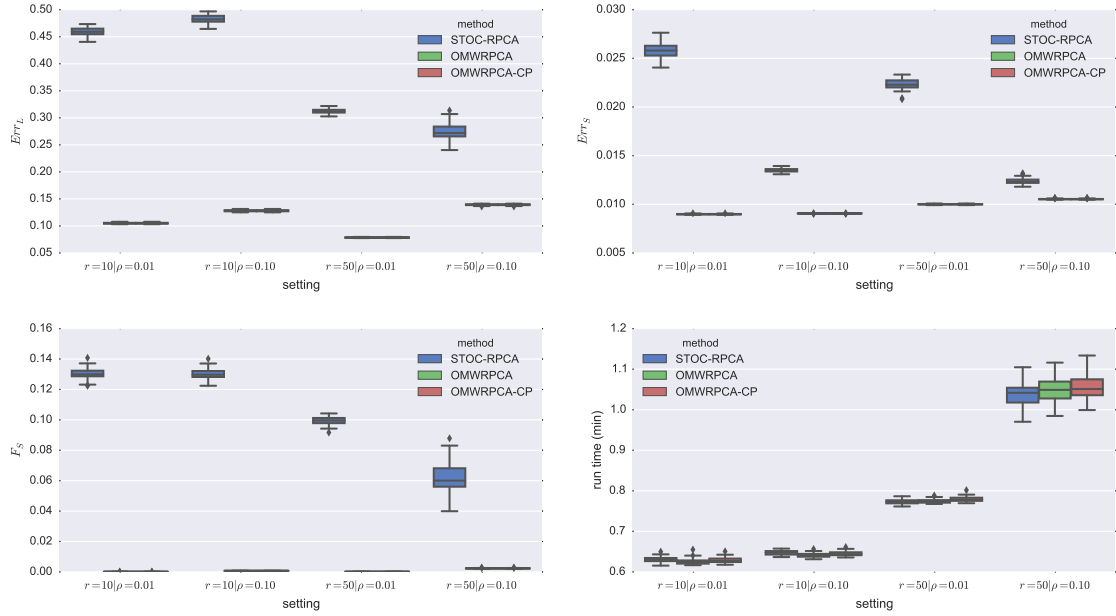

Figure 3: Box plots of $\mathrm{ERR_L}$, $\mathrm{ERR_S}$, $\mathrm{F_S}$ and running times under the simulation study of slowly changing subspace. For each setting, the box plots of STOC-RPCA, OMWRPCA and OMWRPCA-CP are arranged from left to right.
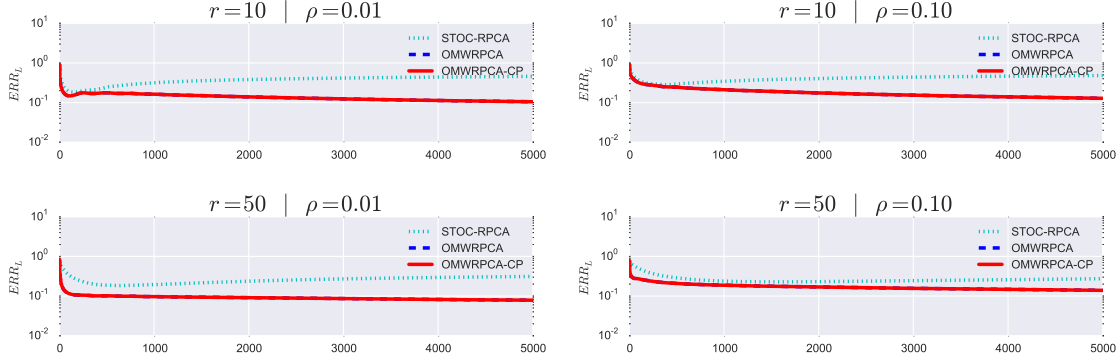
16

Figure 4: Line plots of average $\mathrm{ERR_L}$ over time t under the simulation study of slowly changing subspace.

4.3 Simulation Study 3: slowly changing subspace with change points

We adopt almost all setting of Simulation Study 2 except that we add two change points at time point 1000 and 2000, where the underlying subspace U is changed and generated completely independently. These two change points cut the time line into three pieces, and the start of the subspace U for each piece has rank $r = (r_1, r_2, r_3)^\intercal$, where r_i is the rank of U for ith piece. We consider three settings of r, $(10, 10, 10)^\intercal$, $(50, 50, 50)^\intercal$ and $(10, 50, 25)^\intercal$. We let the subspace U slowly changing over time in each piece as we assumed in Simulation Study 2, where $T_p = 250$. We set $T = 3000$.

Box plots of $\mathrm{ERR_L}$, $\mathrm{ERR_S}$ and $\mathrm{F_S}$ at time point T are shown in Figure 5. Under almost all settings, OMWRPCA outperforms STOC-RPCA, and OMWRPCA-CP has the best performance of all three methods. In Figure 6, we plot 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 $t = 1000$, while the performance of OMWRPCA-CP is stable over time. This indicates that only OMWRPCA-CP can track subspace correctly under the scenario of suddenly changed subspace. Furthermore we find OMWRPCA-CP correctly identify two change points for all replications over all settings. The distribution of the difference between the detected change points and the true change points ($\delta_{cp} = \hat{t}_{cp} - t_{cp}$) is shown in Figure 7. We find δ_{cp} is almost always 0 when the subspaces before and after the change point are highly distinguishable, which represents the cases $r = (50, 50, 50)^\intercal$ and $(10, 50, 25)^\intercal$. For the cases

17

when the change of subspace is not dramatic ($\boldsymbol{r} = (10, 10, 10)^{\mathsf{T}}$), we also have reasonably good result.
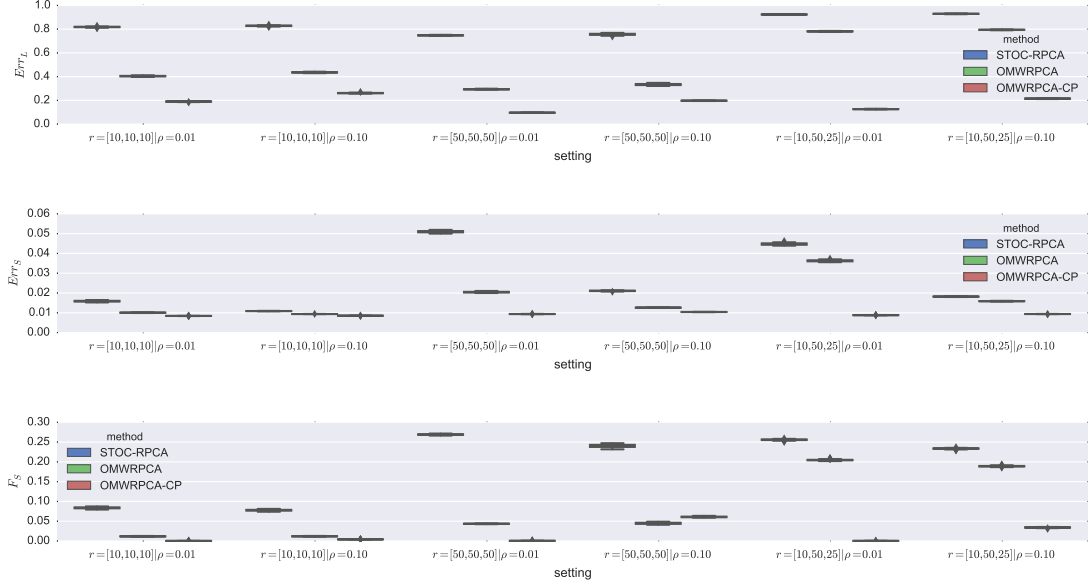
Figure 5: Box plots of $\mathrm{ERR_L}$, $\mathrm{ERR_S}$ and $\mathrm{F_S}$ under the simulation study of slowly changing subspace with change points. For each setting, the box plots of STOC-RPCA, OMWRPCA and OMWRPCA-CP are arranged from left to right.

4.4 Application: background subtraction from surveillance video

Video is a good candidate for low-rank subspace tracking due to the correlation between frames [6]. In surveillance video, background is generally stable and may change very slowly due to varying illumination. We experiment on both the airport and lobby surveillance video data which has been previously studied in [22, 6, 7]. To demonstrate that the OMWRPCA-CP algorithm can effectively track slowly changing subspace with change points, we consider panning a "virtual camera" moving from left to right and 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 and 160×128 for the airport and the lobby video data respectively. The virtual camera has the same hight and half the width. We stack each frame to a column and feed it to the algorithms. To make the background subtraction task even more difficult, we add one change point to both video where the "virtual camera" jumps instantly from the most
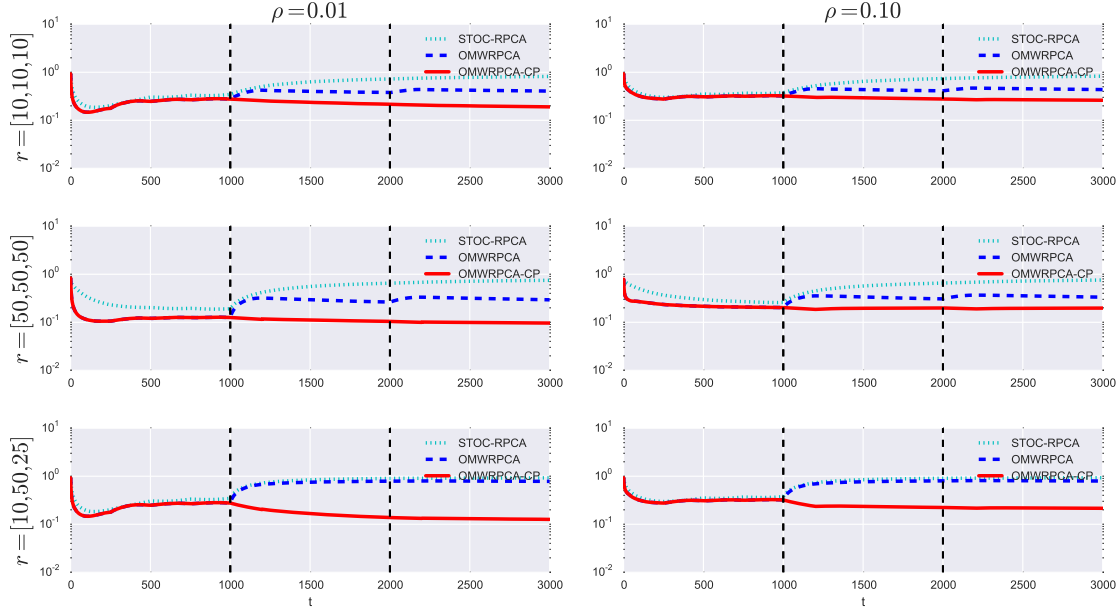
18

Figure 6: Line plots of average ERR_L over time t under the simulation study of suddenly changed subspace. Two change points are at time point 1000 and 2000 are marked by vertical lines.

right-hand side to the most left-hand side.

We choose the following parameters in the algorithm, $\lambda_1 = 1/\sqrt{200}$, $\lambda_2 = 100/\sqrt{200}$, $n_{\text{win}} = 100$, $n_{\text{burnin}} = 300$, $n_{\text{cp-burnin}} = 100$, $n_{\text{test}} = 100$, $N_{\text{check}} = 3$, $\alpha_{\text{prop}} = 1$, $\alpha = 0.01$, $n_{\text{positive}} = 3$ and $n_{\text{tol}} = 2000$. OMWRPCA-CP catches the true change points exactly in both experiments. We show the recovered low-rank $\hat{L}$ and sparse $\hat{S}$ at two frames before and after the change points in Figure 8 and Figure 9 for airport and lobby video data, respectively. OMWRPCA-CP has much sharper recovered low-rank $\hat{L}$ comparing with STOC_RPCA. OMWRPCA-CP also has comparable performance in recovering sparse $\hat{S}$.

5   Conclusion and Discussion

In this paper we have proposed an online robust PCA algorithm. The algorithm can track both slowly and abruptly changed subspaces. By embedding hypothesis tests in the algorithm, the algorithm can discover the exact locations of change points for the underlying low-rank subspaces. Though in this work we have only applied the algorithm for real-time video layering where we separate the video sequence into a slowly changing background and
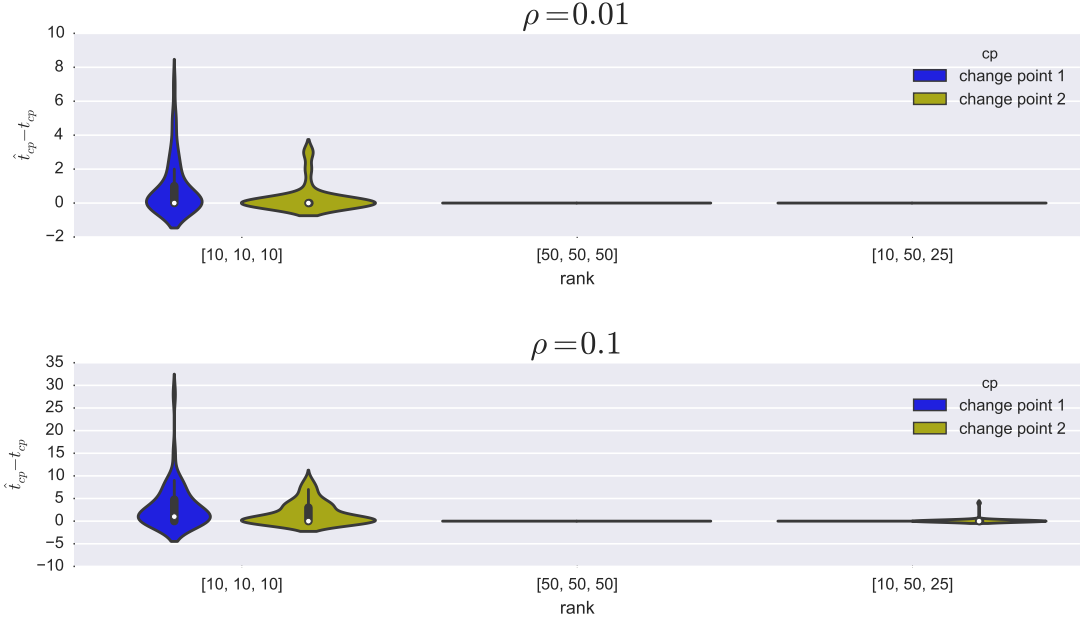
19

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

a sparse foreground, we believe the algorithm can also be used for applications such as failure detection in mechanical systems, intrusion detection in computer networks and human activity recognition based on sensor data. These are left for future work. Another important direction is to develop an automatic (data-driven) method to choose tuning parameters in the algorithm.

References

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

[2] R. Basri and D. W. Jacobs, "Lambertian reflectance and linear subspaces," *IEEE transactions on pattern analysis and machine intelligence*, vol. 25, no. 2, pp. 218–233, 2003.

[3] E. J. Candès and B. Recht, "Exact matrix completion via convex optimization," *Foundations of Computational mathematics*, vol. 9, no. 6, pp. 717–772, 2009.
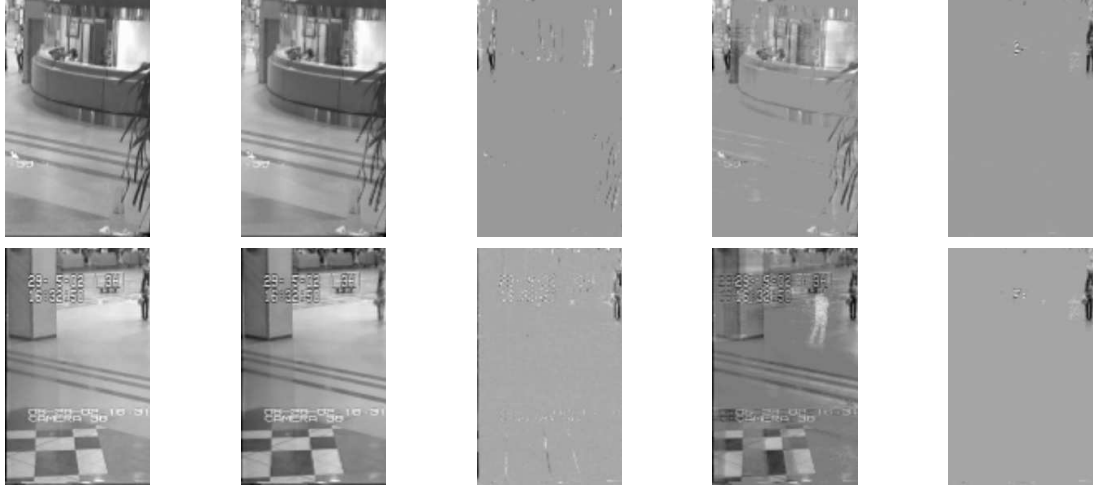
(a) Original frames (b) Low-rank $\hat{L}$ (c) Sparse $\hat{S}$ (d) Low-rank $\hat{L}$ (e) Sparse $\hat{S}$

Figure 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 $\hat{L}$ and Sparse $\hat{S}$ obtained from OMWRPCA-CP. (d)-(e) Low-rank $\hat{L}$ and Sparse $\hat{S}$ obtained from STOC-RPCA.

[4] A. Aravkin, S. Becker, V. Cevher, and P. Olsen, "A variational approach to stable principal component pursuit," *arXiv preprint arXiv:1406.1089*, 2014.

[5] X. Jiang and R. Willett, "Online data thinning via multi-subspace tracking," *arXiv preprint arXiv:1609.03544*, 2016.

[6] E. J. Candès, X. Li, Y. Ma, and J. Wright, "Robust principal component analysis?," *Journal of the ACM (JACM)*, vol. 58, no. 3, p. 11, 2011.

[7] J. He, L. Balzano, and A. Szlam, "Incremental gradient on the grassmannian for online foreground and background separation in subsampled video," in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pp. 1568–1575, IEEE, 2012.

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

[9] F. De La Torre and M. J. Black, "A framework for robust subspace learning," *International Journal of Computer Vision*, vol. 54, no. 1-3, pp. 117–142, 2003.
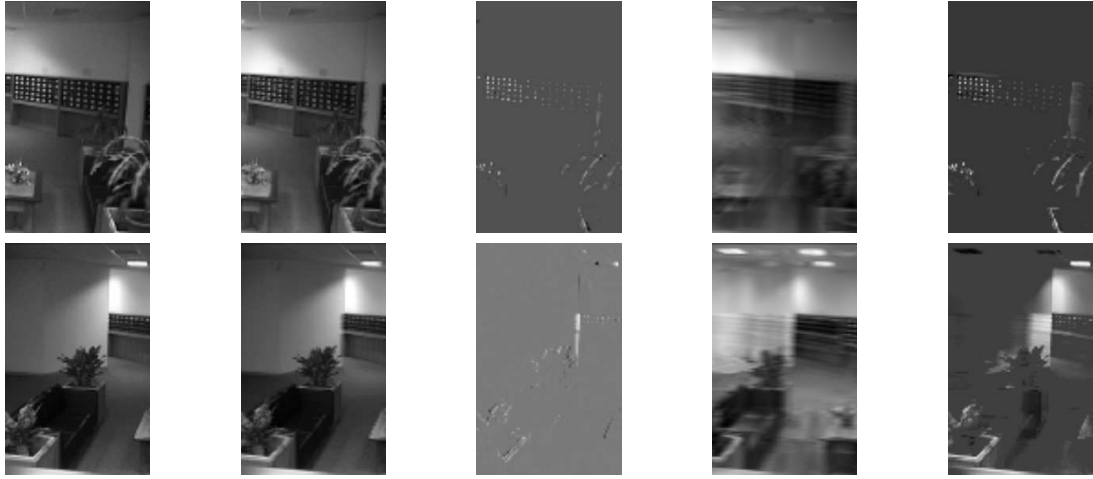
(a) Original frames   (b) Low-rank $\hat{\boldsymbol{L}}$ (c) Sparse $\hat{\boldsymbol{S}}$ (d) Low-rank $\hat{\boldsymbol{L}}$ (e) Sparse $\hat{\boldsymbol{S}}$

Figure 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.

[10] S. Roweis, "Em algorithms for pca and spca," *Advances in neural information processing systems*, pp. 626–632, 1998.

[11] T. Zhang and G. Lerman, "A novel m-estimator for robust pca.," *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 749–808, 2014.

[12] M. McCoy and J. A. Tropp, "Two proposals for robust pca using semidefinite programming," *Electron. J. Statist.*, vol. 5, pp. 1123–1160, 2011.

[13] 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 *Advances in neural information processing systems*, pp. 2080–2088, 2009.

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

[15] C. Qiu, N. Vaswani, B. Lois, and L. Hogben, "Recursive robust pca or recursive sparse recovery in large but structured noise," *IEEE Transactions on Information Theory*, vol. 60, no. 8, pp. 5007–5039, 2014.

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

[17] J. Feng, H. Xu, and S. Yan, "Online robust pca via stochastic optimization," in *Advances in Neural Information Processing Systems 26* (C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, eds.), pp. 404–412, Curran Associates, Inc., 2013.

[18] L. Balzano, R. Nowak, and B. Recht, "Online identification and tracking of subspaces from highly incomplete information," in *Communication, Control, and Computing (Allerton), 2010 48th Annual Allerton Conference on*, pp. 704–711, IEEE, 2010.

[19] Z. Lin, A. Ganesh, J. Wright, L. Wu, M. Chen, and Y. Ma, "Fast convex optimization algorithms for exact recovery of a corrupted low-rank matrix," *Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP)*, vol. 61, 2009.

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

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

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

Algorithm 5: Online Moving Window RPCA with Change Point Detection

---

1 Input: $\{\boldsymbol{m}_1, \ldots, \boldsymbol{m}_T\}$(observed data which are revealed sequentially),
  $\lambda_1, \lambda_2 \in \mathbb{R}$(regularization parameters)

2 $t = 1$; c_p is initialized as an empty list.

3 $t^* \leftarrow \min(t + n_{\text{burnin}} - 1, T)$; 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^*}; $t \leftarrow t^* + 1$.

4 $t_{\text{start}} \leftarrow t$; $H_c \leftarrow \boldsymbol{0}^{m+1}$; B_f and B_c are initialized as empty lists.

5 while $t \leq T$ **do**

6 $\quad$ 1) Reveal the sample $\boldsymbol{m}_t$

7 $\quad$ 2) Project the new sample:

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

8 $\quad$ 3) $A_t \leftarrow A_{t-1} + \boldsymbol{v}_t\boldsymbol{v}_t^\mathsf{T} - \boldsymbol{v}_{t-n_{\text{win}}}\boldsymbol{v}_{t-n_{\text{win}}}$,
  $\quad B_t \leftarrow B_{t-1} + (\boldsymbol{m}_t - \boldsymbol{s}_t)\boldsymbol{v}_t^\mathsf{T} - (\boldsymbol{m}_{t-n_{\text{win}}} - \boldsymbol{s}_{t-n_{\text{win}}})\boldsymbol{v}_{t-n_{\text{win}}}^\mathsf{T}$.

9 $\quad$ 4) Compute $\boldsymbol{U}_t$ with $\boldsymbol{U}_{t-1}$ as warm restart using Algorithm 3:

$$\boldsymbol{U}_t \triangleq \operatorname{argmin} \frac{1}{2} \operatorname{Tr}[\boldsymbol{U}^\mathsf{T}(A_t + \lambda_1 I)\boldsymbol{U}] - \operatorname{Tr}(\boldsymbol{U}^\mathsf{T}B_t).$$

10 $\quad$ 5) Compute $c_t \leftarrow \sum_{i=1}^m \boldsymbol{s}_t[i]$.

11 $\quad$ 6) **if** $t < t_{\text{start}} + n_{\text{cp-burnin}}$ **then**

12 $\quad\quad$ Go to next loop;

13 $\quad$ **else if** $t_{\text{start}} + n_{\text{cp-burnin}} \leq t < t_{\text{start}} + n_{\text{cp-burnin}} + n_{\text{test}}$ **then**

14 $\quad\quad$ $H_c[c_t + 1] \leftarrow H_c[c_t + 1] + 1$; Go to next loop;

15 $\quad$ **else**

16 $\quad\quad$ Do hypothesis testing on c_t with H_c, and get p-value p; $f_t \leftarrow I_{p \leq \alpha}$.

17 $\quad\quad$ Append c_t and f_t to B_c and B_f, respectively.

18 $\quad\quad$ **if** *size of $B_f = N_{\text{check}} + 1$* **then**

19 $\quad\quad\quad$ Pop one element from both B_c and B_f at left-hand side ($c \leftarrow \operatorname{Pop}(\mathrm{B_c})$,
  $\quad\quad\quad f \leftarrow \operatorname{Pop}(\mathrm{B_f})$); Update H_c ($H_c[c + 1] \leftarrow H_c[c + 1] + 1$).

20 $\quad\quad$ **if** *size of $B_f = N_{\text{check}}$* **then**

21 $\quad\quad\quad$ Compute $n_{\text{abnormal}} \leftarrow \sum_{i=1}^{N_{\text{check}}} B_f[i]$

22 $\quad\quad\quad$ **if** $n_{\text{abnormal}} \geq \alpha_{\text{prop}} N_{\text{check}}$ **then**

23 $\quad\quad\quad\quad$ Find change point t_0 by looping over B_f; Append t_0 to c_p.

24 $\quad\quad\quad\quad$ $t \leftarrow t_0$; Jump to step 3.

25 $\quad\quad\quad$ **else**

26 $\quad\quad\quad\quad$ Go to next loop;

27 return c_p *(list of all change points)*