Online (and Offline) Robust PCA: Novel Algorithms and Performance Guarantees

Jinchun Zhan **Brian Lois** **Han Guo** **Namrata Vaswani**
Iowa State University, Ames, IA, 50010, USA

Abstract

In this work we develop and study a novel online robust principal components' analysis (RPCA) algorithm based on the recently introduced ReProCS framework. Our algorithm significantly improves upon the original ReProCS algorithm and it also returns even more accurate offline estimates. The key contribution of this work is a correctness result for this algorithm under relatively mild assumptions. By using extra (but usually valid) assumptions we are able to remove one important limitation of batch RPCA results and two important limitations of a recent result for ReProCS for online RPCA. To the best of our knowledge, this work is among the first correctness results for online RPCA.

1 INTRODUCTION

Given a matrix of data, Principal Components Analysis (PCA) computes a small number of orthogonal directions that contain most of the variability of the data. PCA for relatively noise-free data is easily accomplished via singular value decomposition (SVD). The robust PCA (RPCA) problem, which is the problem of PCA in the presence of outliers, is much harder. In recent work, Candès et al. (2011) posed it as a problem of separating a low-rank matrix, $\boldsymbol{L}$, and a sparse matrix, $\boldsymbol{S}$, from their sum, $\boldsymbol{M} := \boldsymbol{L} + \boldsymbol{S}$. They proposed a convex program called principal components' pursuit (PCP) that provided a provably correct batch solution to this problem under mild assumptions. The same program was also analyzed in Chandrasekaran et al. (2011) and later in Hsu et al. (2011). Since these works, there has been a large amount of work on batch RPCA methods and performance guarantees.

Appearing in Proceedings of the 19^{th} International Conference on Artificial Intelligence and Statistics (AISTATS) 2016, Cadiz, Spain. JMLR: W&CP volume 41.

The noisy case, $\boldsymbol{M} := \boldsymbol{L} + \boldsymbol{S} + \boldsymbol{W}$, was studied in later works, e.g., Zhou et al. (2010).

When RPCA needs to be solved in a recursive fashion for sequentially arriving data vectors it is referred to as online RPCA. Online RPCA assumes that a short sequence of outlier-free (sparse component free) data vectors is available or that there is another way to get an estimate of the initial subspace of the true data (without outliers). A key application of RPCA is the problem of separating a video sequence into foreground and background layers Candès et al. (2011). We show an example in Fig. 1. Video layering is an important first step for automatic video surveillance and many other streaming video analytics tasks. Other applications include recommendation system design, anomaly detection in social network connectivity patterns and survey data analysis Candès et al. (2011).

Problem Definition. At time t we observe a data vector $\boldsymbol{m}_t \in \mathbb{R}^n$ that satisfies

$$\boldsymbol{m}_t = \boldsymbol{\ell}_t + \boldsymbol{x}_t + \boldsymbol{w}_t \tag{1}$$

for $t = t_{\text{train}} + 1, t_{\text{train}} + 2, \dots, t_{\max}$. For $t = 1, 2, \dots, t_{\text{train}}$, $\boldsymbol{x}_t = 0$, i.e., $\boldsymbol{m}_t = \boldsymbol{\ell}_t + \boldsymbol{w}_t$. Here $\boldsymbol{\ell}_t$ is a vector that lies in a low-dimensional subspace that is fixed or slowly changing in such a way that the matrix $\boldsymbol{L}_t := [\boldsymbol{\ell}_1, \boldsymbol{\ell}_2, \dots, \boldsymbol{\ell}_t]$ is a low-rank matrix (for all but very small values of t); $\boldsymbol{x}_t$ is a sparse (outlier) vector; and $\boldsymbol{w}_t$ is small modeling error or noise. We use $\mathcal{T}_t$ to denote the support set of $\boldsymbol{x}_t$ and we use $\boldsymbol{P}_t$ to denote a basis matrix for the subspace from which $\boldsymbol{\ell}_t$ is generated. For $t = 1, 2, \dots, t_{\text{train}}$, the measurements $\boldsymbol{m}_t = \boldsymbol{\ell}_t + \boldsymbol{w}_t$ are assumed to be outlier-free so that it is possible to accurately estimate the initial subspace via PCA. For video surveillance, this corresponds to having a short initial sequence of background only images, which can often be obtained. For $t > t_{\text{train}}$, the goal of online RPCA is to recursively estimate $\boldsymbol{\ell}_t$ and its subspace range($\boldsymbol{P}_t$), and $\boldsymbol{x}_t$ and its support, $\mathcal{T}_t$, as soon as a new data vector $\boldsymbol{m}_t$ arrives or within a short delay. Sometimes, e.g., in video analytics, it is often also desirable to get an improved offline estimate of $\boldsymbol{x}_t$ and $\boldsymbol{\ell}_t$ when possible. We show that this is an easy

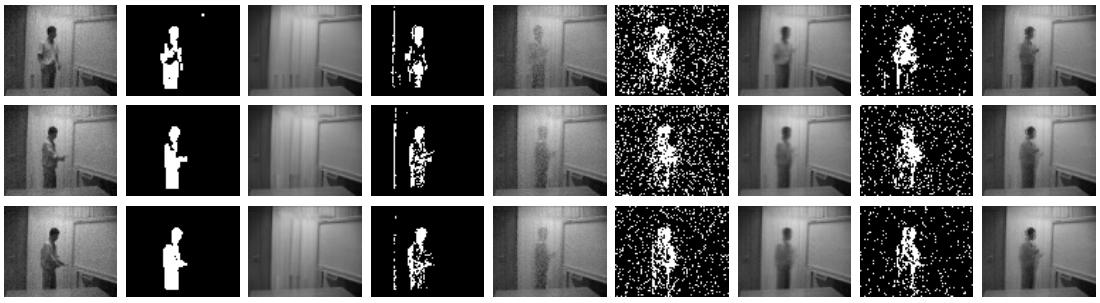

Figure 1: Columns from left to right: a noisy video, foreground and background recovered by automatic ReProCS, GRASTA, Stable PCP and robust subspace learning (RSL), respectively. The foreground in this video consisted of people walking somewhat slowly, stopping and writing on the white board and leaving. The background consisted of moving curtains. Extra i.i.d. Gaussian noise was added to make the separation problem even more difficult. Clearly ReProCS is able to obtain the best separation. Notice that the person or even his shadow are not visible in any of its background estimates. GRASTA is another popular online RPCA method that has the best empirical performance compared with the older online RPCA methods such as adapted iSVD or iRSL. RSL was the best batch RPCA technique before the work of Candes et al. Stable PCP is the noisy version of PCP.

by-product of our solution approach.

In many applications, such as in video, it is actually the sparse outlier $\boldsymbol{x}_t$ that is the quantity of interest. The above problem can thus also be interpreted as one of online sparse matrix recovery in large but structured noise $\boldsymbol{\ell}_t$ and unstructured small noise $\boldsymbol{w}_t$. The unstructured noise, $\boldsymbol{w}_t$, often models the modeling error. For example, when some nonzero entries of $\boldsymbol{x}_t$ are not large enough to be detected (by the proposed algorithm), these can be included into $\boldsymbol{w}_t$. Another example is when the $\boldsymbol{\ell}_t$'s form an approximately low-rank matrix.

Related Work. Solutions for online RPCA have been analyzed in recent works Qiu et al. (2014), Feng et al. (2013b). The work of Qiu et al. (2014) introduced the Recursive Projected Compressive Sensing (ReProCS) algorithmic framework and obtained a partial result for it. Another approach for online RPCA (defined differently from above) and a partial result for it were provided in Feng et al. (2013b). We use the term *partial result* to refer to a performance guarantee that depends on intermediate algorithm estimates satisfying certain properties. We will see examples of this in Sec. 2 when we discuss the above results. In very recent work Lois & Vaswani (2015a,b), a *correctness result* for ReProCS was obtained. The term *correctness result* refers to a complete performance guarantee, i.e., a guarantee that only puts assumptions on the input data (here $\boldsymbol{m}_t$) and/or on the algorithm initialization, but not on intermediate algorithm estimates. Other somewhat related work includes Feng et al. (2013a), Zhan & Vaswani (2015, to appear) and Qiu & Vaswani (2011b).

Some other works, such as He et al. (2012)(GRASTA), Brand (2002) (recursive adaptive-iSVD), Li et al. (2003) (incremental Robust Subspace Learning) or Xu et al. (2013) (GOSUS), only provide an online RPCA algorithm without guarantees. We do not discuss these here. As demonstrated in Fig. 1 and in detailed experiments in Guo et al. (2014), when the outlier support is large and changes slowly over time, ReProCS-based algorithms significantly outperform most of these, besides also outperforming batch methods such as PCP and robust subspace learning (RSL) Candès et al. (2011); Torre & Black (2003).

Contributions. In this work we develop and study *Automatic ReProCS-cPCA* which is a significantly improved algorithm compared to the original one from Qiu & Vaswani (2010, 2011a); Qiu et al. (2014); Lois & Vaswani (2015a). It is able to automatically detect subspace changes within a short delay; is able to correctly estimate the number of directions added or deleted; and is also able to correctly estimate the clusters of eigenvalues along the existing directions. Moreover it is able to accurately estimate both the newly added subspace as well as the newly deleted subspace. The latter is done by re-estimating the current subspace using an approach called cluster-PCA whose basic idea was first introduced by Qiu et al. (2014).

The main contribution of this work is a correctness result for the proposed algorithm under relatively mild assumptions. To the best of our knowledge, this and Lois & Vaswani (2015a,b) are the first correctness results for online RPCA. The result obtained here removes two key limitations of Lois & Vaswani (2015a,b). (1) First, we obtain a result for the case where the $\boldsymbol{\ell}_t$'s can be correlated over time (follow an autoregressive (AR) model) where as the result of Lois & Vaswani (2015a,b) needed mutual independence of the $\boldsymbol{\ell}_t$'s. This models mostly static backgrounds in which changes are only due to independent variations at each time, e.g., light flickers. However, a large class of

background image sequences often change due to factors that are highly correlated over time, e.g., moving waters. This can be better modeled using an AR model. (2) Second, with one extra assumption – that the eigenvalues of the covariance matrix of $\boldsymbol{\ell}_t$ are "clustered" for a period of time after the subspace change has stabilized – we are able to remove another key limitation of Lois & Vaswani (2015a,b). That result needed the rank of $\boldsymbol{L}$ to grow as $\mathcal{O}(\log n)$ while our result allows it to grow as $\mathcal{O}(n)$. The reason for this significant relaxation is that, for ReProCS-cPCA, the dimension of the estimated subspace *does not* keep growing with time. Batch methods such as PCP also allow the rank to grow almost linearly with n. As explained later, the clustered eigenvalues assumption is valid for a large class of data that have multi-scale variations.

Because we use extra (but usually valid) assumptions – accurate initial subspace knowledge and slow subspace change – we are able to remove a key limitation of batch methods Candès et al. (2011); Chandrasekaran et al. (2011); Hsu et al. (2011). Our result requires an order-wise looser bound on the number of time instants for which a particular index i can be outlier-corrupted. In other words, it allows significantly more gradual changes of the outlier support over time. This is important in practice, e.g., in video, foreground objects do not randomly jump around; in social network data, once an anomalous pattern starts to occur, it remains on many of the same edges for a while.

Notation. We use the interval notation $[a, b]$ to mean all of the integers between a and b, inclusive, and similarly for $[a, b)$ etc. For a set $\mathcal{T}$, $|\mathcal{T}|$ denotes its cardinality and $\mathcal{T}^c$ denotes its complement set. We use $\emptyset$ to denote the empty set. Define $\boldsymbol{I}_{\mathcal{T}}$ to be an $n \times |\mathcal{T}|$ matrix of those columns of the identity matrix indexed by $\mathcal{T}$. For a matrix $\boldsymbol{A}$, define $\boldsymbol{A}_{\mathcal{T}} := \boldsymbol{A}\boldsymbol{I}_{\mathcal{T}}$. We use $'$ to denote transpose. The l_p-norm of a vector and the induced l_p-norm of a matrix are denoted by $\|\cdot\|_p$. We refer to a matrix with orthonormal columns as a *basis matrix*. For matrices $\boldsymbol{P}$, $\boldsymbol{Q}$ where the columns of $\boldsymbol{Q}$ are a subset of the columns of $\boldsymbol{P}$, $\boldsymbol{P} \setminus \boldsymbol{Q}$ refers to the matrix of columns in $\boldsymbol{P}$ and not in $\boldsymbol{Q}$. For a matrix $\boldsymbol{H}$, $\boldsymbol{H} \overset{\text{EVD}}{=} \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}'$ denotes its reduced eigenvalue decomposition. For basis matrices $\hat{\boldsymbol{P}}$ and $\boldsymbol{P}$, $\text{dif}(\hat{\boldsymbol{P}}, \boldsymbol{P}) := \|(\boldsymbol{I} - \hat{\boldsymbol{P}}\hat{\boldsymbol{P}}')\boldsymbol{P}\|_2$ quantifies error between their range spaces.

2 DATA MODELS, MAIN RESULT

In this section, we give the data model and correctness result for the proposed algorithm. The algorithm itself is explained in Sec 3 and summarized in Algorithm 1.

Model on the outlier support set, $\mathcal{T}_t$. We give here one practically relevant special case of the most general model on $\mathcal{T}_t$. It requires that the outlier support sets, $\mathcal{T}_t$, have *some* changes over time and have size $|\mathcal{T}_t| \le s$. An example of this is a video application consisting of a foreground with an object of length s or less that remains static for at most β frames at a time. When it moves, it moves *downwards (or upwards, but always in one direction)* by at least s/ρ pixels, and at most s/ρ_2 pixels. Once it reaches the bottom of the scene, it disappears. The maximum motion is such that, if the object were to move at each frame, it still does not go from the top to the bottom of the scene in a time interval of length α. Anytime after it has disappeared another object could appear. We have used this example only to explain the idea. One could also have multiple moving objects and arbitrary motion, as long as the union of their supports follows the model.

Model 2.1 (model on $\mathcal{T}_t$)**.** *Let t^k, with $t^k < t^{k+1}$, denote the times at which $\mathcal{T}_t$ changes and let $\mathcal{T}^{[k]}$ denote the distinct sets. For an integer α,*

1. *assume that $\mathcal{T}_t = \mathcal{T}^{[k]}$ for all times $t \in [t^k, t^{k+1})$ with $(t^{k+1} - t^k) < \beta$ and $|\mathcal{T}^{[k]}| \le s$;*
2. *let ρ be a positive integer so that for any k, $\mathcal{T}^{[k]} \cap \mathcal{T}^{[k+\rho]} = \emptyset$; assume that $\rho^2\beta \le 0.01\alpha$;*
3. *for any k, $\sum_{i=k+1}^{k+\alpha} \left|\mathcal{T}^{[i]} \setminus \mathcal{T}^{[i+1]}\right| \le n$ and for any $k < i \le k + \alpha$, $(\mathcal{T}^{[k]} \setminus \mathcal{T}^{[k+1]}) \cap (\mathcal{T}^{[i]} \setminus \mathcal{T}^{[i+1]}) = \emptyset$ (one way to ensure the first condition is to require that for all i, $|\mathcal{T}^{[i]} \setminus \mathcal{T}^{[i+1]}| \le \frac{s}{\rho_2}$ with $\frac{s}{\rho_2}\alpha \le n$).*

In this model, k takes values $1, 2, \ldots$; the largest value it can take is $t_{\max}$. We set α in Theorem 2.7.

Model on ℓ_t. A common model for data that lies in a low-dimensional subspace is to assume that, at all times, it is independent and identically distributed (iid) with zero mean and a fixed *low-rank* covariance matrix $\boldsymbol{\Sigma}$. However this can be restrictive since, in many applications, data statistics change with time, albeit slowly. To model this perfectly, one would need to assume that $\boldsymbol{\ell}_t$ is zero mean with covariance matrix $\boldsymbol{\Sigma}_t$ at time t. If $\boldsymbol{\Sigma}_t \overset{EVD}{=} \boldsymbol{P}_t\boldsymbol{\Lambda}_t\boldsymbol{P}_t'$, this means that both $\boldsymbol{P}_t$ and $\boldsymbol{\Lambda}_t$ can change at each time t, though slowly. This is the most general model but it has an identifiability problem if the goal is to estimate the subspace from which $\boldsymbol{\ell}_t$ was generated, range($\boldsymbol{P}_t$). The subspace cannot be estimated with one data point. So, if $\boldsymbol{P}_t$ changes at each time, it is not clear how one can estimate all the subspaces. To resolve this issue, a general enough but tractable option is to assume that $\boldsymbol{P}_t$ is piecewise constant with time and $\boldsymbol{\Lambda}_t$ can change at each time. To ensure that $\boldsymbol{\Sigma}_t$ changes "slowly", we assume that, when $\boldsymbol{P}_t$ changes, the eigenvalues along the newly added directions are small initially for the first d frames, and after that they can increase gradually or suddenly to any large value. One precise model for this is specified next.

The model given below assumes boundedness of $\boldsymbol{\ell}_t$. This is more practically valid than the usual Gaussian assumption since most sensor data or noise is bounded. We also replace independence of $\boldsymbol{\ell}_t$'s by an AR model with independent perturbations $\boldsymbol{\nu}_t$ and we place all the above assumptions on $\boldsymbol{\nu}_t$. This is a more practical model and includes independence as a special case.

Model 2.2 (Model on $\boldsymbol{\ell}_t$)**.** *Assume the following.*

1. *Let $\boldsymbol{\ell}_0 = \mathbf{0}$ and for $t = 1, 2, \ldots t_{\max}$, assume that*

$$\boldsymbol{\ell}_t = b\boldsymbol{\ell}_{t-1} + \boldsymbol{\nu}_t$$

for a $b < 1$. Assume that the $\boldsymbol{\nu}_t$ are zero mean, mutually independent and bounded random vectors with covariance matrix

$$\mathrm{Cov}(\boldsymbol{\nu}_t) = \boldsymbol{\Sigma}_t \overset{\text{EVD}}{=} \boldsymbol{P}_t \boldsymbol{\Lambda}_t \boldsymbol{P}_t{}'.$$

Define $\lambda^- := \lambda_{\min}\left(\frac{1}{t_{\text{train}}}\sum_{t=1}^{t_{\text{train}}} \boldsymbol{\Lambda}_t\right)$, and $\lambda^+ := \lambda_{\max}\left(\frac{1}{t_{\text{train}}}\sum_{t=1}^{t_{\text{train}}} \boldsymbol{\Lambda}_t\right)$.

2. *Let $t_1, t_2, \ldots t_J$ denote the subspace change times. The basis matrices $\boldsymbol{P}_t$ change as*

$$\boldsymbol{P}_t = \begin{cases} [(\boldsymbol{P}_{t-1}\boldsymbol{R}_t \setminus \boldsymbol{P}_{t,\text{old}})\ \boldsymbol{P}_{t,\text{new}}] & \text{if } t = t_1, t_2, \ldots\ t_J \\ \boldsymbol{P}_{t-1} & \text{otherwise.} \end{cases}$$

Here $\boldsymbol{R}_t$ is a rotation matrix and $\boldsymbol{P}_{t_j,\text{new}}$ and $\boldsymbol{P}_{t_j,\text{old}}$ are basis matrices of size $n \times r_{j,\text{new}}$ and $n \times r_{j,\text{old}}$ respectively.

3. *The eigenvalues' matrices $\boldsymbol{\Lambda}_t$ are such that (i) $\lambda_{\max}(\boldsymbol{\Lambda}_t) \le \lambda^+$ and (ii) the following holds. Let $\boldsymbol{\Lambda}_{t,\text{new}} := \boldsymbol{P}_{t_j,\text{new}}{}'\boldsymbol{\Sigma}_t\boldsymbol{P}_{t_j,\text{new}}$ for $t \in [t_j, t_{j+1})$. Assume that*

$$0 < \lambda^- \le \lambda^-_{\text{new}} \le \lambda^+_{\text{new}} \le 3\lambda^- \tag{2}$$

where

$$\lambda^-_{\text{new}} := \min_j \min_{t\in[t_j,t_j+d]} \lambda_{\min}\left(\boldsymbol{\Lambda}_{t,\text{new}}\right),$$
$$\lambda^+_{\text{new}} := \max_j \max_{t\in[t_j,t_j+d]} \lambda_{\max}\left(\boldsymbol{\Lambda}_{t,\text{new}}\right).$$

4. *Assume that $d \ge (K+2)\alpha$ and $t_{j+1} - t_j \ge d$. Here K and α are algorithm parameters that are set in Theorem 2.7. This lower bound on $t_{j+1} - t_j$ along with (2) quantifies "slow subspace change".*

5. *Other assumptions: (i) define $t_0 := 1$ and assume that $t_{\text{train}} \in [t_0, t_1)$; (ii) for $j = 0, 1, 2, \ldots, J$, define $r_j := \mathrm{rank}(\boldsymbol{P}_{t_j})$, $r_{j,\text{new}} := \mathrm{rank}(\boldsymbol{P}_{t_j,\text{new}})$, $r_{j,\text{old}} := \mathrm{rank}(\boldsymbol{P}_{t_j,\text{old}})$ Clearly, $r_j = r_{j-1} + r_{j,\text{new}} - r_{j,\text{old}}$. Assume that $r_{j,\text{new}}$ is small enough compared to $r_{j,\text{old}}$ so that $r_j \le r$ and $r_{j,\text{new}} \le r_{\text{new}}$ for all j for constants r and r_{new}. Assume that $r + r_{\text{new}} < \min(n, t_{j+1} - t_j)$.*

6. *Since the $\boldsymbol{\nu}_t$'s are bounded random variables, there exists a $\gamma < \infty$ and a $\gamma_{\text{new}} \le \gamma$ such that*

$$\max_t \|\boldsymbol{P}_t{}'\boldsymbol{\nu}_t\|_2 \le \gamma,\ \max_j \max_{t\in[t_j,t_j+d]} \|\boldsymbol{P}_{t_j,\text{new}}{}'\boldsymbol{\nu}_t\|_\infty \le \gamma_{\text{new}}.$$

We assume an upper bound on γ_{new} in the Thoerem.

Various low-rank and "slow changing" $\boldsymbol{\Sigma}_t$ models are special cases of this model. A particularly relevant special case is one that allows the variance along new directions to increase slowly: $(\boldsymbol{\Lambda}_{t,\text{new}})_{i,i} = (v_i)^{t-t_j} q_i \lambda^-$ for $i = 1, \ldots, r_{j,\text{new}}$ where $q_i \ge 1$ and $v_i > 1$ but not too large. An upper bound on v_i of the form $q_i(v_i)^d \le 3$ ensures that (2) holds.

The above model requires the directions to get deleted and added at the same set of times $t = t_j$. This is done for simplicity. In general directions could get deleted at any other time as well. Moreover, we can significantly relax the lower bound in (2) to the following: we can let λ^-_{new} be the minimum eigenvalue along the new directions of any α-frame *average* covariance matrix over the period $[t_j, t_j + d]$ and we can require this to be larger than λ^-. For video analytics, this translates to requiring that, after a subspace change, *enough (but not necessarily all)* background frames have "detectable" energy along the new directions, so that the minimum eigenvalue of the average covariance along the new directions is above a threshold. For the recommendation systems' application, the initial set of users may only be influenced by a few, say five, factors, but as more users come in to the system, *some (not necessarily all)* of them may also get influenced by a sixth factor (newly added direction).

Eigenvalues' clustering. In order to be able to design an accurate algorithm to delete the old directions by re-estimating the current subspace, we need one of the following for a period of d_2 frames within the interval $[t_j, t_{j+1})$. We either need the condition number of $\boldsymbol{\Lambda}_t$ (or equivalently of $\boldsymbol{\Sigma}_t$) to be small, or we need a generalization of it: we need its eigenvalues to be "clustered" into a few (at most ϑ) clusters in such a way that the condition number within each cluster is small and the distance between consecutive clusters is large. The problem with requiring a tight upper bound on the condition number of $\boldsymbol{\Sigma}_t$ is that it disallows situations where the $\boldsymbol{\ell}_t$'s constitute large but structured noise. This is why the "clustered" generalization is needed.

Validity of clustered eigenvalues assumption. This is valid for data that has variations at different scales. For example, for data that has variations at two scales, there would be two clusters, the large scale variations would form the first cluster and the small scale ones

the second cluster. These clusters would naturally be well separated.

Let ϑ denote the maximum number of clusters. As we will explain in Sec. 3, the subspace deletion via re-estimation step is done after the new directions are accurately estimated. As we will prove, with high probability (whp), this will not happen until $t_j + K\alpha$. Thus, we assume that the above clustering assumption holds for the period $[t_j + K\alpha + 1, t_j + K\alpha + d_2]$ with $d_2 > (\vartheta + 3)\alpha$. This implicitly also assumes $t_{j+1} - t_j > K\alpha + d_2$.

Model 2.3. *Assume the following.*

1. *Assume that $t_{j+1} - t_j > K\alpha + d_2$ for an integer $d_2 \geq (\vartheta+3)\alpha$ (where ϑ is defined below). Assume that for all $t \in [t_j + K\alpha, t_j + K\alpha + d_2]$, $\mathbf{\Lambda}_t$ is constant; let $\mathbf{\Lambda}_{(j)}$ be this constant matrix; assume that $\lambda_{\min}(\mathbf{\Lambda}_{(j)}) \geq \lambda^-$.*
2. *Define a partition of the index set $\{1, 2, \dots r_j\}$ into sets $\mathcal{G}_{j,1}, \mathcal{G}_{j,2}, \dots, \mathcal{G}_{j,\vartheta_j}$ as follows. Sort the eigenvalues of $\mathbf{\Lambda}_{(j)}$ in decreasing order of magnitude. To define $\mathcal{G}_{j,1}$, start with the first (largest) eigenvalue and keep adding smaller eigenvalues to the set and stop just when the ratio of the maximum to the minimum eigenvalue first exceeds $g^+ = 1.5$. Suppose this happens for the i-th eigenvalue. Then, define $\mathcal{G}_{j,1} = \{1, 2, \dots i-1\}$. For $\mathcal{G}_{j,2}$, start with the i-th eigenvalue and repeat the same procedure. Stop when there are no more nonzero eigenvalues. Let ϑ_j denote the number of clusters and let $\vartheta := \max_j \vartheta_j$. Observe that the above way of defining the partition is one way to ensure that the condition number of the eigenvalues in each set of the partition is below $g^+ = 1.5$. Assume that the eigenvalues are "clustered" (distance between the sets $\mathcal{G}_{j,k}$ is large enough), i.e.,*

$$\chi_{j,k} := \frac{\max\limits_{i \in \mathcal{G}_{j,k+1}} \lambda_i\left(\mathbf{\Lambda}_{(j)}\right)}{\min\limits_{i \in \mathcal{G}_{j,k}} \lambda_i\left(\mathbf{\Lambda}_{(j)}\right)} \leq \chi^+ = 0.2 \qquad (3)$$

Remark 2.4. *Model 2.3 requires $\mathbf{\Lambda}_t$ to be constant for $t \in [t_j + K\alpha, t_j + K\alpha + d_2]$ while Model 2.3 requires the eigenvalues along $\boldsymbol{P}_{t_j,\text{new}}$ to be small for $t \in [t_j, t_j + d]$ with $d \geq (K+2)\alpha$. Taken together, this means that for all $t \in [t_j, t_j + K\alpha + d_2]$, we are requiring that the eigenvalues along $\boldsymbol{P}_{t_j,\text{new}}$ be small. However after this time, there is no constraint on its eigenvalues. By $t = t_{j+1} + K\alpha - 1$ some or all eigenvalues along it could have increased to λ^+ or decreased to zero or be anything in between.*

Denseness. To separate sparse $\boldsymbol{x}_t$'s from the $\boldsymbol{\ell}_t$'s, the basis vectors for the subspace from which the $\boldsymbol{\ell}_t$'s are generated cannot be sparse. We quantify this using an incoherence condition similar to Candès et al. (2011).

Model 2.5 (Denseness)**.** *Let μ be the smallest real number such that $\max_i \|\boldsymbol{P}_{t_j}{}' \boldsymbol{I}_i\|_2^2 \leq \frac{\mu r_j}{n}$ and $\max_i \|\boldsymbol{P}_{t_j,\text{new}}{}' \boldsymbol{I}_i\|_2^2 \leq \frac{\mu r_{j,\text{new}}}{n}$ for all j. Assume that*

$$2sr\mu \leq 0.09n \text{ and } 2sr_{\text{new}}\mu \leq 0.0004n$$

Assumption on $\boldsymbol{w}_t$. Assume the following.

Model 2.6. *$\boldsymbol{w}_t$'s are zero mean, mutually independent over time, and bounded with $\|\boldsymbol{w}_t\|_2 \leq \epsilon_w$.*

Main result. We give below a correctness result for Automatic ReproCS-cPCA (Algorithm 1). It has five parameters - α, K, ξ, ω, $\hat{g}^+$ - that need to be set appropriately. The parameter α is the number of consecutive time instants that are used to obtain *an* estimate of the new subspace, and K is the total number of times the new subspace is estimated before we get an accurate enough estimate of it. The parameter ξ is the bound on the l_2 norm of the noise seen by the projected sparse recovery step of the algorithm, and ω is the threshold used to recover the support of $\boldsymbol{x}_t$. The parameter $\hat{g}^+$ is used to estimate the eigenvalue clusters automatically from an empirical covariance matrix computed using an appropriate set of $\hat{\boldsymbol{\ell}}_t$'s.

Theorem 2.7. *Consider Algorithm 1. Assume that $\boldsymbol{m}_t$ satisfies (1) with $\boldsymbol{x}_t = 0$ for $t \leq t_{\text{train}}$. Pick a ζ that satisfies $\zeta \leq \min\left\{\frac{10^{-4}}{(r+r_{\text{new}})^2}, \frac{0.03\lambda^-}{(r+r_{\text{new}})^2\lambda^+}, \frac{1}{(r+r_{\text{new}})^3\gamma^2}, \frac{0.05\lambda^-}{(r+r_{\text{new}})^3\gamma^2}\right\}$. Suppose that the following hold.*

1. *enough initial training data is available: $t_{\text{train}} \geq \frac{32(2r\gamma^2)}{(1-b^2)(1-b)(0.01r_{\text{new}}\zeta\lambda^-)^2}(11\log n + \log 4)$*
2. *algorithm parameters are set as: $\xi = \xi_{\text{cor}} := \epsilon_w + \frac{2\sqrt{\zeta}+\sqrt{r_{\text{new}}}\gamma_{\text{new}}}{1-b}$; $\omega = 7\xi$; $\hat{g}^+ := \frac{g^+ + 0.25}{1-0.25} = 2.33$; $K = \left\lceil \frac{\log(0.32 r_{\text{new}}\zeta)}{\log(0.83)} \right\rceil$; $\alpha = \max\{\alpha_{\text{add}}, \alpha_{\text{del}}\}$ where $\alpha_{\text{add}} \geq 32\frac{1.2^2(2\sqrt{\zeta}+\sqrt{r_{\text{new}}}\gamma_{\text{new}}+2\epsilon_w)^4}{(1-b)^6}\frac{(1-b^2)^2}{(0.01r_{\text{new}}\zeta\lambda^-)^2}(11\log n + \log(40(K+1)J))$ and $\alpha_{\text{del}} \geq \frac{8\cdot 3r\gamma^2}{(0.01r_{\text{new}}\zeta\lambda^-)^2}(\log(6\max_j \vartheta_j J) + 11\log n)$;*
3. *model on $\mathcal{T}_t$: Model 2.1 holds;*
4. *model on $\boldsymbol{\ell}_t$:*
   Model 2.2 holds with $b \leq b_0 = 0.1$ and with $\sqrt{r_{\text{new}}}\gamma_{\text{new}}$ small enough so that $14\xi \leq \min_t \min_{i \in \mathcal{T}_t} |(x_t)_i|$;
   Model 2.3 holds with $|\mathcal{G}_{j,k}| \geq 0.27(r + r_{\text{new}})$;
   Model 2.5 holds.
5. *model on $\boldsymbol{w}_t$: Model 2.6 holds with $\epsilon_w^2 \leq 0.03\zeta\lambda^-$*
6. *independence: Let $\mathcal{T} := \{\mathcal{T}_t\}_{t=1,2,\dots,t_{\max}}$. Assume that $\mathcal{T}, \boldsymbol{w}_1, \boldsymbol{w}_2, \dots, \boldsymbol{w}_{t_{\max}}, \boldsymbol{\nu}_1, \boldsymbol{\nu}_2, \dots, \boldsymbol{\nu}_{t_{\max}}$ are mutually independent random variables.*

Then, with probability $\geq 1 - 3n^{-10}$*, at all times* t*,*

1. $\mathcal{T}_t$ *is exactly recovered, i.e.* $\hat{\mathcal{T}}_t = \mathcal{T}_t$ *for all* t*;*
2. $\|\boldsymbol{x}_t - \hat{\boldsymbol{x}}_t\|_2 \leq 1.2\left(\sqrt{\zeta} + \sqrt{r_{\text{new}}}\gamma_{\text{new}} + 2\epsilon_w\right)$ *and* $\|\hat{\boldsymbol{\ell}}_t - \boldsymbol{\ell}_t\|_2 \leq \|\boldsymbol{x}_t - \hat{\boldsymbol{x}}_t\|_2 + \epsilon_w$*;*
3. *the subspace error* $\text{SE}_t := \|(\boldsymbol{I} - \hat{\boldsymbol{P}}_t\hat{\boldsymbol{P}}_t{}')\boldsymbol{P}_t\|_2 \leq 10^{-2}\sqrt{\zeta}$ *for all* $t \in [t_j + d, t_{j+1})$*;*
4. *the subspace change time estimates given by Algorithm 1 satisfy* $t_j \leq \hat{t}_j \leq t_j + 2\alpha$*; and its estimates of the number of new directions are correct:* $\hat{r}_{j,\text{new},k} = r_{j,\text{new}}$ *for* $j = 1, \dots, J$*.*
5. *eigenvalue clusters are recovered exactly:* $\hat{\mathcal{G}}_{j,k} = \mathcal{G}_{j,k}$ *for all* j *and* k*; thus the its estimate of number of deleted directions is also correct.*

Proof: See supplementary document.

Remark 2.8. *Notice that the lower bound* $|\mathcal{G}_{j,k}| \geq 0.27(r + r_{\text{new}})$ *can only hold if the number of clusters* ϑ_j *is at most 3. This is one choice that works along with the given bounds on other quantities such as* $\rho^2\beta$*. It can be made larger if we assume a tighter bound on* $\rho^2\beta$ *for example. But what will remain true is that our result requires the number of clusters to be* $\mathcal{O}(1)$*.*

Offline RPCA. In certain situations, an improved offline estimate is desirable. In many applications, there is no need for an online solution. We show here that, with a delay of at most $(K+2)\alpha$ frames, it is possible to recover $\boldsymbol{x}_t$ and $\boldsymbol{\ell}_t$ with close to zero error.

Corollary 2.9 (Offline RPCA)**.** *Consider the estimates given in the last two lines of Algorithm 1. Under the assumptions of Theorem 2.7, with probability at least* $1 - 3n^{-10}$*, at all times* t*,* $\|\boldsymbol{x}_t - \hat{\boldsymbol{x}}_t^{\text{offline}}\|_2 \leq 1.2(\sqrt{\zeta} + 2\epsilon_w)$*,* $\|\hat{\boldsymbol{\ell}}_t^{\text{offline}} - \boldsymbol{\ell}_t\|_2 \leq 1.2(\sqrt{\zeta} + 3\epsilon_w)$*, and all its other conclusions hold.*

The offline recovery error can be made smaller and smaller by reducing ζ (this, in turn, will result in an increased delay between subspace change times).

Discussion.

To our knowledge, our work and Lois & Vaswani (2015a,b) are the only correctness results for an online RPCA method. All other results are for batch techniques. As we explain, our work significantly improves upon Lois & Vaswani (2015a,b). We allow the $\boldsymbol{\ell}_t$'s to be correlated over time and model them using a first order AR model. This is significantly more practically valid than the independence assumption used in Lois & Vaswani (2015a,b).

Consider the bounds on rank and sparsity. Let $\boldsymbol{L} := [\ell_1, \ell_2 \dots \ell_{t_{\max}}]$, $\boldsymbol{S} := [x_1, x_2 \dots x_{t_{\max}}]$, $r_{\text{mat}} := \text{rank}(\boldsymbol{L})$ and let s_{mat} be the number of nonzero entries in $\boldsymbol{S}$.

Algorithm 1 Automatic ReProCS-cPCA

Compute $\hat{\lambda}^-_{\text{train}}$ as the r_0-th eigenvalue of $\frac{1}{t_{\text{train}}}\sum_{t=1}^{t_{\text{train}}} \boldsymbol{m}_t\boldsymbol{m}_t'$ and $\hat{\boldsymbol{P}}_{t_{\text{train}}}$ as its top r_0 eigenvectors.
Set thresh $= \frac{\hat{\lambda}^-_{\text{train}}}{2}$. Set $\hat{\boldsymbol{P}}_{t,*} \leftarrow \hat{\boldsymbol{P}}_{t_{\text{train}}}$, $\hat{\boldsymbol{P}}_{t,\text{new}} \leftarrow [.]$, $\hat{j} \leftarrow 0$, phase $\leftarrow$ detect.
For every $t > t_{\text{train}}$, do

- Estimate $\mathcal{T}_t$ and $\boldsymbol{x}_t$.
  1. compute $\boldsymbol{\Phi}_t \leftarrow \boldsymbol{I} - \hat{\boldsymbol{P}}_{t-1}\hat{\boldsymbol{P}}_{t-1}{}'$ and $\boldsymbol{y}_t \leftarrow \boldsymbol{\Phi}_t\boldsymbol{m}_t$
  2. solve $\min_{\boldsymbol{x}} \|\boldsymbol{x}\|_1$ *s.t.* $\|\boldsymbol{y}_t - \boldsymbol{\Phi}_t\boldsymbol{x}\|_2 \leq \xi$ and let $\hat{\boldsymbol{x}}_{t,\text{cs}}$ denote its solution
  3. compute $\hat{\mathcal{T}}_t = \{i : |(\hat{\boldsymbol{x}}_{t,\text{cs}})_i| > \omega\}$
  4. LS: compute $\hat{\boldsymbol{x}}_t = \boldsymbol{I}_{\hat{\mathcal{T}}_t}((\boldsymbol{\Phi}_t)_{\hat{\mathcal{T}}_t})^\dagger\boldsymbol{y}_t$
- Estimate $\boldsymbol{\ell}_t$: $\hat{\boldsymbol{\ell}}_t \leftarrow \boldsymbol{m}_t - \hat{\boldsymbol{x}}_t$
- If $t \mod \alpha \neq 0$ then $\hat{\boldsymbol{P}}_{t,*} \leftarrow \hat{\boldsymbol{P}}_{t-1,*}$, $\hat{\boldsymbol{P}}_{t,\text{new}} \leftarrow \hat{\boldsymbol{P}}_{t-1,\text{new}}$, $\hat{\boldsymbol{P}}_t \leftarrow [\hat{\boldsymbol{P}}_{t,*}\ \hat{\boldsymbol{P}}_{t,\text{new}}]$
- If $t \mod \alpha = 0$ then
  if phase = detect **then**
  1. Set $u = \frac{t}{\alpha}$ and compute $\boldsymbol{\mathcal{D}}_u = (\boldsymbol{I} - \hat{\boldsymbol{P}}_{u\alpha-1,*}\hat{\boldsymbol{P}}_{u\alpha-1,*}{}')[\hat{\boldsymbol{\ell}}_{(u-1)\alpha+1}, \dots \hat{\boldsymbol{\ell}}_{u\alpha}]$
  2. $\hat{\boldsymbol{P}}_{t,*} \leftarrow \hat{\boldsymbol{P}}_{t-1,*}$, $\hat{\boldsymbol{P}}_{t,\text{new}} \leftarrow \hat{\boldsymbol{P}}_{t-1,\text{new}}$, $\hat{\boldsymbol{P}}_t \leftarrow [\hat{\boldsymbol{P}}_{t,*}\ \hat{\boldsymbol{P}}_{t,\text{new}}]$
  3. If $\lambda_{\max}(\frac{1}{\alpha}\boldsymbol{\mathcal{D}}_u\boldsymbol{\mathcal{D}}_u{}') \geq$ thresh then
     (a) phase $\leftarrow$ ppca, $\hat{j} \leftarrow \hat{j} + 1$, $k \leftarrow 0$, $\hat{t}_{\hat{j}} = t$

  else if phase = ppca **then**
  1. Set $u = \frac{t}{\alpha}$ and compute $\boldsymbol{\mathcal{D}}_u = (\boldsymbol{I} - \hat{\boldsymbol{P}}_{u\alpha-1,*}\hat{\boldsymbol{P}}_{u\alpha-1,*}{}')[\hat{\boldsymbol{\ell}}_{(u-1)\alpha+1}, \dots \hat{\boldsymbol{\ell}}_{u\alpha}]$
  2. $\hat{\boldsymbol{P}}_{t,\text{new}} \leftarrow$ eigenvectors $\left(\frac{1}{\alpha}\boldsymbol{\mathcal{D}}_u\boldsymbol{\mathcal{D}}_u{}', \text{thresh}\right)$, $\hat{\boldsymbol{P}}_{t,*} \leftarrow \hat{\boldsymbol{P}}_{t-1,*}$, $\hat{\boldsymbol{P}}_t \leftarrow [\hat{\boldsymbol{P}}_{t,*}\ \hat{\boldsymbol{P}}_{t,\text{new}}]$
  3. $k \leftarrow k + 1$, set $\hat{r}_{j,\text{new},k} = \text{rank}(\hat{\boldsymbol{P}}_{t,\text{new}})$
  4. If $k == K$, then
     (a) phase $\leftarrow$ cPCA, reset $k \leftarrow 0$

  else if phase = cPCA**) then**
  1. cluster-PCA (summarized in Algorithm 2)

  end-if

eigenvectors$(\boldsymbol{\mathcal{M}}, \text{thresh})$ returns a basis matrix for the span of eigenvectors with eigenvalue above thresh.
Offline RPCA: at $t = \hat{t}_j + K\alpha$, for all $t \in [\hat{t}_{j-1} + K\alpha + 1, \hat{t}_j + K\alpha]$, compute $\hat{\boldsymbol{x}}_t^{\text{offline}} \leftarrow \boldsymbol{I}_{\hat{\mathcal{T}}_t}((\boldsymbol{\Phi}_{\hat{t}_j+K\alpha})_{\hat{\mathcal{T}}_t})^\dagger\boldsymbol{\Phi}_{\hat{t}_j+K\alpha}\boldsymbol{m}_t$ and $\hat{\boldsymbol{\ell}}_t^{\text{offline}} \leftarrow \boldsymbol{m}_t - \boldsymbol{x}_t$

With our models, $s_{\text{mat}} \leq st_{\max}$ and $r_{\text{mat}} \leq r_0 + Jr_{\text{new}} \leq r + Jr_{\text{new}}$ with both bounds being tight. Models 2.1 and 2.5 constrain s and s, r, r_{new} respectively. Using the expression for α, it can be argued that both definitely hold in two regimes of interest. The first is $r \in \mathcal{O}(\log n)$, $s_{\text{mat}} \in \mathcal{O}(\frac{nt_{\max}}{(\log n)^8})$ and $r_{\text{mat}} \in \mathcal{O}(n)$. The second is $r \in \mathcal{O}(1)$, $s_{\text{mat}} \in \mathcal{O}(\frac{nt_{\max}}{\log n})$ and $r_{\text{mat}} \in \mathcal{O}(n)$. These requirements are significantly weaker than what

was needed in Lois & Vaswani (2015a,b) that analyzed ReProCS without the cluster-PCA based subspace deletion step: their result needed $r_{\text{mat}} \in \mathcal{O}(\log n)$. In either regime, our requirements are weaker than those of the PCP results from Chandrasekaran et al. (2011); Hsu et al. (2011): they need the product $r_{\text{mat}} s_{\text{mat}} \in \mathcal{O}(nt_{\max})$; thus if $s_{\text{mat}} \in \mathcal{O}(nt_{\max})$, they would require r_{mat} to be $\mathcal{O}(1)$. In the first regime, our conditions are slightly stronger than those of the PCP result from Candès et al. (2011) while in the second, they are comparable: it needs $r_{\text{mat}} \in \mathcal{O}(\frac{n}{(\log n)^2})$ and $s_{\text{mat}} \in \mathcal{O}(nt_{\max})$.

An important advantage of our work over PCP and other batch methods is that we allow much more gradual changes of the set of outliers over time. From the assumption on $\mathcal{T}_t$, it is easy to see that we allow the number of outliers per row of $\boldsymbol{L}$ to be $\mathcal{O}(t_{\max})$, as long as the sets follow Model 2.1. The PCP results from Chandrasekaran et al. (2011); Hsu et al. (2011) need this number to be $\mathcal{O}(\frac{t_{\max}}{r_{\text{mat}}})$ which is stronger. The PCP result from Candès et al. (2011) needs an even stronger condition – the set $\cup_{t=1}^{t_{\max}} \mathcal{T}_t$ should be generated uniformly at random. Moreover, we do not need any assumption on the right singular vectors of $\boldsymbol{L}$ while all results for PCP do. Finally, we analyze an *online* algorithm that is faster and needs less storage.

We get the above advantages over PCP because we use extra assumptions - accurate initial subspace knowledge, slow subspace change and and clustered eigenvalues after a subspace change has stabilized. Also our result needs five algorithm parameters to be appropriately set. The actual values used for these parameters in experiments is much smaller than what our theorem requires. The PCP results need this for none Candès et al. (2011) or at most one Chandrasekaran et al. (2011); Hsu et al. (2011) algorithm parameter.

3 AUTOMATIC REPROCS-CPCA

The Automatic ReProCS-cPCA algorithm is given in Algorithm 1. Its main idea is as follows. It begins by estimating the initial subspace as the top r_0 left singular vectors of $[\boldsymbol{m}_1, \boldsymbol{m}_2, \dots, \boldsymbol{m}_{t_{\text{train}}}]$. At time t, if the previous subspace estimate, $\hat{\boldsymbol{P}}_{t-1}$, is accurate enough, because of the "slow subspace change" assumption, projecting $\boldsymbol{m}_t = \boldsymbol{x}_t + \boldsymbol{\ell}_t + \boldsymbol{w}_t$ onto its orthogonal complement nullifies most of $\boldsymbol{\ell}_t$. Specifically, we compute $\boldsymbol{y}_t := \boldsymbol{\Phi}_t \boldsymbol{m}_t$ where $\boldsymbol{\Phi}_t := \boldsymbol{I} - \hat{\boldsymbol{P}}_{t-1}\hat{\boldsymbol{P}}_{t-1}'$. Clearly, $\boldsymbol{y}_t = \boldsymbol{\Phi}_t \boldsymbol{x}_t + \boldsymbol{b}_t$ where $\boldsymbol{b}_t := \boldsymbol{\Phi}_t \boldsymbol{\ell}_t + \boldsymbol{\Phi}_t \boldsymbol{w}_t$. $\|\boldsymbol{\Phi}_t \boldsymbol{\ell}_t\|_2$ is small due to slow subspace change and $\boldsymbol{w}_t$ is small by assumption. Thus recovering $\boldsymbol{x}_t$ from $\boldsymbol{y}_t$ then becomes a traditional sparse recovery problem in small noise Candes (2008). We recover $\boldsymbol{x}_t$ by l_1 minimization and estimate its support by thresholding. We use the estimated support, $\hat{\mathcal{T}}_t$, to get an improved debiased estimate of $\boldsymbol{x}_t$, denoted $\hat{\boldsymbol{x}}_t$, by least squares (LS) estimation on $\hat{\mathcal{T}}_t$. We then estimate of $\boldsymbol{\ell}_t$ as $\hat{\boldsymbol{\ell}}_t = \boldsymbol{m}_t - \hat{\boldsymbol{x}}_t$. By the denseness assumption given in Model 2.5, it can be argued that the restricted isometry constant (RIC) of $\boldsymbol{\Phi}_t$ will be small. Under the theorem's assumptions, we can bound it by 0.14. This ensures that a sparse $\boldsymbol{x}_t$ is indeed accurately recoverable from $\boldsymbol{y}_t$. With the support estimation threshold ω set as in Theorem 2.7, it can be argued that the support will be exactly recovered, i.e., $\hat{\mathcal{T}}_t = \mathcal{T}_t$. Let $\boldsymbol{e}_t := \boldsymbol{\ell}_t - \hat{\boldsymbol{\ell}}_t$. It can be shown that $\boldsymbol{e}_t = (\hat{\boldsymbol{x}}_t - \boldsymbol{x}_t) - \boldsymbol{w}_t$ satisfies

$$\boldsymbol{e}_t = \boldsymbol{I}_{\mathcal{T}_t}[(\boldsymbol{\Phi}_t)_{\mathcal{T}_t}'(\boldsymbol{\Phi}_t)_{\mathcal{T}_t}]^{-1}\boldsymbol{I}_{\mathcal{T}_t}'\boldsymbol{\Phi}_t(\boldsymbol{\ell}_t + \boldsymbol{w}_t) - \boldsymbol{w}_t.$$

Using the bound on RIC of $\boldsymbol{\Phi}_t$, $\|(\boldsymbol{\Phi}_t)_{\mathcal{T}_t}'(\boldsymbol{\Phi}_t)_{\mathcal{T}_t}^{-1}\|_2 \le (1-0.14)^{-1} < 1.2$. Thus $\|\boldsymbol{e}_t\|_2 \le 1.2\|\boldsymbol{\Phi}_t\boldsymbol{\ell}_t\|_2 + 2.2\|\boldsymbol{w}_t\|_2$ (i.e., $\boldsymbol{\ell}_t$ is accurately recovered).

The estimates $\hat{\boldsymbol{\ell}}_t$ are used in the subspace estimation step which involves (i) detecting subspace change; (ii) K steps of projection-PCA, each done with a new set

Algorithm 2 cluster PCA (called from Algorithm 1)

1. If $k == 0$, estimate the clusters
   (a) Set $u = \frac{t}{\alpha}$ and compute $\hat{\boldsymbol{\Sigma}}_{\text{sample}} = \frac{1}{\alpha}\sum_{t=(u-1)\alpha+1}^{u\alpha} \hat{\boldsymbol{\ell}}_t \hat{\boldsymbol{\ell}}_t'$ and sort its eigenvalues in decreasing order.
   (b) For $\hat{\mathcal{G}}_{j,1}$, start with the first (largest) eigenvalue and keep adding smaller eigenvalues to the set until either the ratio of the maximum to the minimum eigenvalue first exceeds $\hat{g}^+$ or the next eigenvalue is below $0.25\hat{\lambda}_{\text{train}}^-$. Suppose this happens for the (k_1+1)-th eigenvalue. Then, define $\hat{\mathcal{G}}_{j,1} = \{1, 2, \dots, k_1\}$.
   For $\hat{\mathcal{G}}_{j,2}$, start with the (k_1+1)-th eigenvalue and repeat the same procedure.
   Repeat the above for each new cluster until the next eigenvalue is below $0.25\hat{\lambda}_{\text{train}}^-$.
   (c) $k \leftarrow k+1$, $\hat{\boldsymbol{P}}_{t,*} \leftarrow \hat{\boldsymbol{P}}_{t-1,*}$, $\hat{\boldsymbol{P}}_{t,\text{new}} \leftarrow \hat{\boldsymbol{P}}_{t-1,\text{new}}$, $\hat{\boldsymbol{P}}_t \leftarrow [\hat{\boldsymbol{P}}_{t,*}\ \hat{\boldsymbol{P}}_{t,\text{new}}]$
2. If $1 \le k \le \vartheta$, estimate the k-th cluster's subspace by cluster-PCA
   (a) Set $u = \frac{t}{\alpha}$, set $\hat{\boldsymbol{G}}_{j,0} \leftarrow [.]$.
   - let $\hat{\boldsymbol{G}}_{j,\text{det},k} := [\hat{\boldsymbol{G}}_{j,0}, \hat{\boldsymbol{G}}_{j,1}, \dots \hat{\boldsymbol{G}}_{j,k-1}]$ and let $\Psi_k := (\boldsymbol{I} - \hat{\boldsymbol{G}}_{j,\text{det},k}\hat{\boldsymbol{G}}_{j,\text{det},k}')$; compute $\boldsymbol{\mathcal{M}}_{\text{cpca}} = \Psi_k \left(\frac{1}{\alpha}\sum_{t\in(u-1)\alpha+1}^{u\alpha} \hat{\boldsymbol{\ell}}_t \hat{\boldsymbol{\ell}}_t'\right) \Psi_k$
   - compute $\hat{\boldsymbol{G}}_{j,k} \leftarrow$ eigenvectors$(\boldsymbol{\mathcal{M}}_{\text{cpca}}, , |\hat{\mathcal{G}}_{j,k}|)$
   (b) $k \leftarrow k+1$, $\hat{\boldsymbol{P}}_{t,*} \leftarrow \hat{\boldsymbol{P}}_{t-1,*}$, $\hat{\boldsymbol{P}}_{t,\text{new}} \leftarrow \hat{\boldsymbol{P}}_{t-1,\text{new}}$, $\hat{\boldsymbol{P}}_t \leftarrow [\hat{\boldsymbol{P}}_{t,*}\ \hat{\boldsymbol{P}}_{t,\text{new}}]$
3. If $k == \vartheta$,
   - set $\hat{\boldsymbol{P}}_t \leftarrow [\hat{\boldsymbol{G}}_{j,1} \cdots \hat{\boldsymbol{G}}_{j,\vartheta}]$, $\hat{\boldsymbol{P}}_{t,*} \leftarrow \hat{\boldsymbol{P}}_t$, reset $\hat{\boldsymbol{P}}_{t,\text{new}} \leftarrow [.]$
   - phase $\leftarrow$ detect, reset $k \leftarrow 0$.

of α frames of $\hat{\boldsymbol{\ell}}_t$, to get an accurate enough estimate of the newly added subspace; and (iii) cluster-PCA to delete the old subspace by re-estimating the current subspace. At the end of the projection PCA step, the estimated subspace dimension is at most $r + r_{\text{new}}$, and after cluster-PCA, it comes down to at most r.

Automatically setting algorithm parameters. The algorithm has five parameters. As explained in Guo et al. (2014), one can set $\xi_t = \|\boldsymbol{\Phi}_t \hat{\boldsymbol{\ell}}_{t-1}\|_2$. One can either set $\omega_t = 7\xi_t$ or, in the video application, one can use the average image pixel intensity to set it. In Guo et al. (2014), they used $\omega = q\sqrt{\|\boldsymbol{m}_t\|_2^2/n}$ with $q = 1$ when it was known that $\|\boldsymbol{x}_t\|_2$ is of the same order as $\|\boldsymbol{\ell}_t\|_2$; and $q = 0.25$ when $\|\boldsymbol{x}_t\|_2$ was known to be much smaller (the case of foreground moving objects whose intensity is very similar to that of background objects). There is no good heuristic to pick α except that α_{add} should be large enough compared to r_{new} and α_{del} should be large enough compared to r. We used $\alpha = 100$ and $K = 12$ in our experiments. We need K to be large enough so that the new subspace is accurately recovered at the end of K projection-PCA iterations. Thus, one way to set K indirectly is as follows: do projection-PCA for at least $K_{\min}$ times, but after that stop when there is not much difference between $\hat{\boldsymbol{P}}_{j,\text{new},k}{}'\hat{\boldsymbol{\ell}}_t$ and $\hat{\boldsymbol{P}}_{j,\text{new},k+1}{}'\hat{\boldsymbol{\ell}}_t$ Qiu et al. (2014); Guo et al. (2014). This, along with imposing an upper bound on K works well in practice Guo et al. (2014). We can set $\hat{g}^+$ as suggested in Qiu et al. (2014); by applying any clustering algorithm from literature, e.g., k-means, to the empirical covariance matrix used in the clustering step of cluster-PCA.

4 PROOF OUTLINE: THEOREM 2.7

Consider the j-the subspace change interval. Let $\boldsymbol{P}_* := \boldsymbol{P}_{t_{j-1}}$, $\boldsymbol{P}_{\text{new}} := \boldsymbol{P}_{t_j,\text{new}}$ and let $\hat{\boldsymbol{P}}_* := \hat{\boldsymbol{P}}_{t_{j-1}}$. Assume that at $t = t_j - 1$, the subspace, range$(\boldsymbol{P}_*)$, is accurately recovered, i.e., $\text{SE}_t \le r\zeta$. Conditioned on this, we use the following steps to show that the same bound holds for SE_t at $t = t_{j+1} - 1$ as well.

1. First, we show that the subspace change is detected within a short delay of t_j. We show that $t_j \le \hat{t}_j \le t_j + 2\alpha$ whp.
2. At $t = \hat{t}_j + \alpha$, the first projection-PCA step is done to get the first estimate, $\hat{\boldsymbol{P}}_{\text{new},1}$, of range$(\boldsymbol{P}_{\text{new}})$. This computes the top singular vectors of $[\hat{\boldsymbol{\ell}}_{\hat{t}_j+1}, \hat{\boldsymbol{\ell}}_{\hat{t}_j+2}, \dots, \hat{\boldsymbol{\ell}}_{\hat{t}_j+\alpha}]$ projected orthogonal to range$(\hat{\boldsymbol{P}}_*)$. Before $\hat{t}_j + \alpha$, the noise seen by the projected sparse recovery step, $\boldsymbol{b}_t$, is the largest. Hence the error $\boldsymbol{e}_t$ is also the largest for the $\hat{\boldsymbol{\ell}}_t$'s used in this step. However due to slow subspace change, even this error is not too large. Because of this and because $\boldsymbol{e}_t$ is approximately sparse with support $\mathcal{T}_t$ and $\mathcal{T}_t$ follows Model 2.1, we can argue that $\hat{\boldsymbol{P}}_{\text{new},1}$ is a good estimate. We show that $\text{dif}([\hat{\boldsymbol{P}}_* \ \hat{\boldsymbol{P}}_{\text{new},1}], \boldsymbol{P}_{\text{new}}) \le 0.6 < 1$. Thus, at this time the subspace error $\text{SE}_t \le r\zeta + 0.6$.
3. At $t = \hat{t}_j + k\alpha$, for $k = 1, 2, \dots, K$, the k-th projection-PCA (p-PCA) step is done to get $\hat{\boldsymbol{P}}_{\text{new},k}$. After the first p-PCA step, $\hat{\boldsymbol{P}}_t = [\hat{\boldsymbol{P}}_* \ \hat{\boldsymbol{P}}_{\text{new},1}]$ and this reduces $\boldsymbol{b}_t$ and hence $\boldsymbol{e}_t$ for the $\hat{\boldsymbol{\ell}}_t$'s in the next α frames. This fact and the approximate sparseness of $\boldsymbol{e}_t$ and Model 2.1 on $\mathcal{T}_t$, in turn, imply that the perturbation seen by the second p-PCA step will be even smaller. So $\hat{\boldsymbol{P}}_{\text{new},2}$ will be a more accurate estimate of range$(\boldsymbol{P}_{\text{new}})$ than $\hat{\boldsymbol{P}}_{\text{new},1}$. Repeating the same argument, the third estimate will be even better and so on. We can show that $\text{dif}([\hat{\boldsymbol{P}}_* \ \hat{\boldsymbol{P}}_{\text{new},k}], \boldsymbol{P}_{\text{new}}) \le 0.83^k + 0.84 r_{\text{new}}\zeta$ and so $\text{SE}_t \le r\zeta + 0.83^k + 0.84 r_{\text{new}}\zeta$.
4. We set the value of K in the theorem to ensure that by $t = \hat{t}_j + K\alpha$, $\text{SE}_t \le (r + r_{\text{new}})\zeta$.
5. In the interval $[\hat{t}_j + K\alpha + 1, \hat{t}_j + K\alpha + (\vartheta + 1)\alpha]$, cluster-PCA is done to delete range$(P_{t_j,\text{old}})$. At the end of this step, we can show the bound on SE_t has reduces from $(r + r_{\text{new}})\zeta$ to $r\zeta$.
6. Finally, we also argue that there are no false subspace change detects for any $t \in (\hat{t}_j + K\alpha + (\vartheta + 1)\alpha + 1, t_{j+1})$. This ensures $\hat{t}_{j+1} \ge t_{j+1}$.

To prove the theorem, we first show that the initial subspace is recovered accurately enough, i.e., $\text{SE}_t \le r\zeta$ at $t = t_{\text{train}} + 1$, whp. Then, repeating the above argument for each subspace change period, we can obtain the subspace error bounds of the theorem. The sparse recovery error bounds can be obtained by using these and quantifying the discussion of Sec. 3.

The main part of the proof is the analysis of the projection-PCA steps (for subspace addition) and the cluster-PCA steps (for subspace deletion). To analyze the k-th projection-PCA step, we first use a lemma based on the $\sin\theta$ theorem Davis & Kahan (1970) to bound the subspace error in recovering range$(P_{t_j,\text{new}})$. This upper bound consists of three terms. We then bound these terms using the matrix Azuma inequality from Tropp (2012). The matrix Azuma is significantly harder to apply than matrix Hoeffding used in Lois & Vaswani (2015a,b). This is because we need to get the sums of conditional expectations of the quantities in each term in a form that their spectral norm can be bounded easily. The obvious way of doing this can lead to very loose bounds. To get the desired bounds, one need to rewrite $\boldsymbol{\ell}_t$ in terms of past $\boldsymbol{\nu}_t$'s and use the fact that the contribution of very old $\boldsymbol{\nu}_t$'s is negligible and the contribution due to the last α $\boldsymbol{\nu}_t$'s is only slightly larger than that of one $\boldsymbol{\nu}_t$ (because $b \le b_0 = 0.1$).

The analysis of cluster-PCA is a significant generalization of the above ideas. The slow subspace change assumption is replaced by the clustering assumption at various places in its proof.

References

Brand, M. Incremental singular value decomposition of uncertain data with missing values. In *Eur. Conf. on Comp. Vis. (ECCV)*, 2002.

Candes, E. The restricted isometry property and its implications for compressed sensing. *Compte Rendus de l'Academie des Sciences, Paris, Serie I*, pp. 589–592, 2008.

Candès, E. J., Li, X., Ma, Y., and Wright, J. Robust principal component analysis? *Journal of ACM*, 58 (3), 2011.

Chandrasekaran, V., Sanghavi, S., Parrilo, P. A., and Willsky, A. S. Rank-sparsity incoherence for matrix decomposition. *SIAM Journal on Optimization*, 21, 2011.

Davis, C. and Kahan, W. M. The rotation of eigenvectors by a perturbation. iii. *SIAM Journal on Numerical Analysis*, 7:1–46, Mar. 1970.

Feng, J., Xu, H., Mannor, S., and Yan, S. Online pca for contaminated data. In *Adv. Neural Info. Proc. Sys. (NIPS)*, 2013a.

Feng, J., Xu, H., and Yan, S. Online robust pca via stochastic optimization. In *Adv. Neural Info. Proc. Sys. (NIPS)*, 2013b.

Guo, H., Qiu, C., and Vaswani, N. An online algorithm for separating sparse and low-dimensional signal sequences from their sum. *IEEE Trans. Sig. Proc.*, 62 (16):4284–4297, 2014.

He, J., Balzano, L., and Szlam, A. Incremental gradient on the grassmannian for online foreground and background separation in subsampled video. In *IEEE Conf. on Comp. Vis. Pat. Rec. (CVPR)*, 2012.

Hsu, D., Kakade, S.M., and Zhang, T. Robust matrix decomposition with sparse corruptions. *IEEE Trans. Info. Th.*, Nov. 2011.

Li, Y., Xu, L., Morphett, J., and Jacobs, R. An integrated algorithm of incremental and robust pca. In *IEEE Intl. Conf. Image Proc. (ICIP)*, pp. 245–248, 2003.

Lois, B. and Vaswani, N. A correctness result for online robust pca. In *IEEE Intl. Conf. Acoustics, Speech, Sig. Proc. (ICASSP)*, 2015a.

Lois, B. and Vaswani, N. Online matrix completion and online robust pca. In *IEEE Intl. Symp. Info. Th. (ISIT)*, 2015b.

Qiu, C. and Vaswani, N. Real-time robust principal components' pursuit. In *Allerton Conference on Communication, Control, and Computing*, pp. 591–598. IEEE, 2010.

Qiu, C. and Vaswani, N. Recursive sparse recovery in large but correlated noise. In *Allerton Conf. on Communication, Control, and Computing*, 2011a.

Qiu, C. and Vaswani, N. Support predicted modified-cs for recursive robust principal components' pursuit. In *IEEE Intl. Symp. Info. Th. (ISIT)*, 2011b.

Qiu, C., Vaswani, N., Lois, B., and Hogben, L. Recursive robust pca or recursive sparse recovery in large but structured noise. *IEEE Trans. Info. Th.*, 60(8): 5007–5039, August 2014.

Torre, F. De La and Black, M. J. A framework for robust subspace learning. *International Journal of Computer Vision*, 54:117–142, 2003.

Tropp, J. A. User-friendly tail bounds for sums of random matrices. *Foundations of Computational Mathematics*, 12(4), 2012.

Xu, J., Ithapu, V. K, Mukherjee, L., Rehg, J. M, and Singh, V. Gosus: Grassmannian online subspace updates with structured-sparsity. In *Computer Vision (ICCV), 2013 IEEE International Conference on*, pp. 3376–3383. IEEE, 2013.

Zhan, J. and Vaswani, N. Robust pca with partial subspace knowledge. *IEEE Trans. Sig. Proc.*, 2015, to appear.

Zhou, Zihan, Li, Xiaodong, Wright, John, Candes, Emmanuel, and Ma, Yi. Stable principal component pursuit. In *Information Theory Proceedings (ISIT), 2010 IEEE International Symposium on*, pp. 1518–1522. IEEE, 2010.