Online (Recursive) Robust Principal Components Analysis

Namrata Vaswani, Chenlu Qiu, Brian Lois, Han Guo and Jinchun Zhan
Iowa State University, Ames IA, USA
Email: namrata@iastate.edu

Abstract

This work studies the problem of sequentially recovering a sparse vector S_t and a vector from a low-dimensional subspace L_t from knowledge of their sum $M_t := L_t + S_t$. If the primary goal is to recover the low-dimensional subspace in which the L_t's lie, then the problem is one of online or recursive robust principal components analysis (PCA). An example of where such a problem might arise is in separating a sparse foreground and a slowly changing dense background in a surveillance video. In this chapter, we describe our recently proposed algorithm, called Recursive Projected Compressed Sensing (ReProCS), to solve this problem and demonstrate its significant advantage over other robust PCA based methods for this video layering problem. We also summarize the performance guarantees for ReProCS. Lastly, we briefly describe our work on modified-PCP, which is a piecewise batch approach that removes a key limitation of ReProCS, however it retains a key limitation of existing work.

I. Introduction

Principal Components Analysis (PCA) is a widely used dimension reduction technique that finds a small number of orthogonal basis vectors, called principal components, along which most of the variability of the dataset lies. It is well known that PCA is sensitive to outliers. Accurately computing the principal subspace in the presence of outliers is called robust PCA [?], [?], [?], [?]. Often, for time series data, the principal subspace changes gradually over time. Updating its estimate on-the-fly (recursively) in the presence of outliers, as more data comes in is referred to as online or recursive robust PCA [?], [?], [?]. "Outlier" is a loosely defined term that refers to any corruption that is not small compared to the true data vector and that occurs occasionally. As suggested in [?], [?], an outlier can be nicely modeled as a sparse vector whose nonzero values can have any magnitude.

A key application where the robust PCA problem occurs is in video analysis where the goal is to separate a slowly changing background from moving foreground objects [?], [?]. If we stack each frame as a column vector, the background is well modeled as being dense and lying in a low dimensional subspace that may gradually change over time, while the moving foreground objects constitute the sparse outliers [?], [?]. Other applications include detection of brain activation patterns from functional MRI (fMRI) sequences (the "active" part of the brain can be interpreted as a sparse outlier), detection of anomalous behavior in dynamic social networks and sensor networks based detection and tracking of abnormal events such as forest fires or oil spills. In most of these applications, an online solution is desirable.

The moving objects or the active regions of the brain or the oil spill region may be "outliers" for the PCA problem, but in most cases, these are actually the signals-of-interest whereas the background image is the noise. Also, all the above signals-of-interest are sparse vectors. Thus, this problem can also be interpreted as one of recursively recovering a time sequence of sparse signals, S_t, from measurements $M_t := S_t + L_t$ that are corrupted by (potentially) large magnitude but dense and structured noise, L_t. The structure that we require is that L_t be dense and lie in a low dimensional subspace that is either fixed or changes "slowly enough".

A. Related Work

There has been a large amount of work on robust PCA, e.g. [?], [?], [?], [?], [?], [?], [?], and recursive or online or incremental robust PCA e.g. [?], [?], [?]. In most of these works, either the locations of the missing/corruped data points are assumed known [?] (not a practical assumption); or they first detect the corrupted data points and then replace their values using nearby values [?]; or weight each data point in proportion to its reliability (thus soft-detecting and down-weighting the likely outliers) [?], [?]; or just remove the entire outlier vector [?], [?]. Detecting or soft-detecting outliers (S_t) as in [?], [?], [?] is easy when the outlier magnitude is large, but not otherwise. When the signal of interest is S_t, the most difficult situation is when nonzero elements of S_t have small magnitude compared to those of L_t and in this case, these approaches do not work.

In recent works [?], [?], a new and elegant solution to robust PCA called Principal Components' Pursuit (PCP) has been proposed, that does not require a two step outlier location detection/correction process and also does not throw out the entire vector. It redefines batch robust PCA as a problem of separating a low rank matrix, $\mathcal{L}_t := [L_1, \dots, L_t]$, from a sparse matrix, $\mathcal{S}_t := [S_1, \dots, S_t]$, using the measurement matrix, $\mathcal{M}_t := [M_1, \dots, M_t] = \mathcal{L}_t + \mathcal{S}_t$.

Let $\|A\|_*$ be the nuclear norm of A (sum of singular values of A) while $\|A\|_1$ is the ℓ_1 norm of A seen as a long vector. It was shown in [?] that, with high probability (w.h.p.), one can recover $\mathcal{L}_t$ and $\mathcal{S}_t$ exactly by solving PCP:

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

provided that (a) the left and right singular vectors of $\mathcal{L}_t$ are dense; (b) any element of the matrix $\mathcal{S}_t$ is nonzero w.p. ϱ, and zero w.p. $1-\varrho$, independent of all others; and (c) the rank of $\mathcal{L}_t$ and the support size of $\mathcal{S}_t$ are bounded by a small enough values.

B. Our work: provably correct and practically usable recursive robust PCA solutions

As described earlier, many applications where robust PCA is required, such as video surveillance, require an online (recursive) solution. Even for offline applications, a recursive solution is typically faster than a batch one. Moreover, in many of these applications, the support of the sparse part changes in a correlated fashion over time, e.g., in video foreground objects do not move randomly, and may also be static for short periods of time. In recent work [?], [?], [?], [?], we introduced a novel solution approach, called Recursive Projected Compressive Sensing (ReProCS), that recursively recovered S_t and L_t at each time t. Moreover, as we showed later in [?], [?], it can also provably handle correlated support changes significantly better than batch approaches because it uses extra assumptions (accurate initial subspace knowledge and slow subspace change). In simulation and real data experiments (see [?] and http://www.ece.iastate.edu/~chenlu/ReProCS/ReProCS_main.htm), it was faster than batch methods such as PCP and also significantly outperformed them in situations where the support changes were correlated over time (as long as there was *some* support change every few frames). In [?], we studied a simple modification of the original ReProCS idea and obtained a performance guarantee for it. This result needed mild assumptions except one: it needed an assumption on intermediate algorithm estimates and hence it was not a correctness result. In very recent work [?], [?], we also have a complete correctness result for ReProCS that replaces this restrictive assumption by a simple and practical assumption on support change of S_t.

The ReProCS algorithm itself has a key limitation that is removed by our later work on modified-PCP [?], [?] (see Section **??**), however as explained in Sec **??**, modified-PCP does not handle correlated support change as well as ReProCS. In this bookchapter we provide an overview of both ReProCS and modified-PCP and a discussion of when which is better and why. We briefly also explain their correctness results and show simulation and real data experiments comparing them with various existing batch and online robust PCA solutions.

To the best of our knowledge, [?] provided the first performance guarantee for an online robust PCA approach or equivalently for an online sparse + low-rank matrix recovery approach. Of course it was only a partial result (depended on intermediate algorithm estimates satisfying a certain property). Another partial result for online robust PCA was given in more recent work by Feng et al [?]. In very recent work [?], [?], we provide the first complete correctness result for an online robust PCA / online sparse + low-rank matrix recovery approach. Another online algorithm that addresses the online robust PCA problem is GRASTA [?], however, it does not contain any performance analysis. We should mention here that our results directly apply to the recursive version of the matrix completion problem [?] as well since it is a simpler special case of the current problem (the support set of S_t is the set of indices of the missing entries and is thus known) [?]. We explicitly give the algorithm and result for online MC in [?].

The proof techniques used in our work are very different from those used to analyze other recent batch robust PCA works [?], [?], [?], [?], [?], [?], [?], [?], [?], [?], [?], [?]. The works of [?], [?] also study a different case: that where an entire vector is either an outlier or an inlier. Our proof utilizes (a) sparse recovery results [?]; (b) results from matrix perturbation theory that bound the estimation error in computing the eigenvectors of a perturbed Hermitian matrix with respect to eigenvectors of the original Hermitian matrix (the famous sin θ theorem of Davis and Kahan [?]) and (c) high probability bounds on eigenvalues of sums of independent random matrices (matrix Hoeffding inequality [?]).

A key difference of our approach to analyzing the subspace estimation step compared with most existing work analyzing finite sample PCA, e.g. [?] and references therein, is that it needs to provably work in the presence of error/noise that is correlated with L_t. Most existing works, including [?] and the references it discusses, assume that the noise is independent of (or at least uncorrelated with) the data. However, in our case, because of how the estimate $\hat{L}_t$ is computed, the error $e_t := L_t - \hat{L}_t$ is correlated with L_t. As a result, the tools developed in these earlier works cannot be used for our problem. This is also the reason why simple PCA cannot be used and we need to develop and analyze a projection-PCA algorithm for subspace tracking (this fact is explained in detail in the Appendix of [?]).

The ReProCS approach is related to that of [?], [?], [?] in that all of these first try to nullify the low dimensional signal by projecting the measurement vector into a subspace perpendicular to that of the low dimensional signal, and then solve for the sparse "error" vector (outlier). However, the big difference is that in all of these works the basis for the subspace of the low dimensional signal is *perfectly known.* Our work studies *the case where the subspace is not known*. We have an initial approximate estimate of the subspace, but over time it can change significantly. In this work, to keep things simple, we mostly use ℓ_1 minimization done separately for each time instant (also referred to as basis pursuit denoising (BPDN)) [**?**], [**?**]. However, this can be replaced by any other sparse recovery algorithm, either recursive or batch, as long as the batch algorithm is applied to α frames at a time, e.g. one can replace BPDN by modified-CS or support-predicted modified-CS [**?**]. We have used a combination of simple ℓ_1 minimization and modified-CS / weighted ℓ_1 in the practical version of the ReProCS algorithm that was used for our video experiments (see Section **??**).

C. Chapter Organization

This chapter is organized as follows. In Section **??**, we give the problem definition and the key assumptions required. We explain the main idea of the ReProCS algorithm and develop its practically usable version in Section **??**. We develop the modified-PCP solution in Section **??**. The performance guarantees for both ReProCS and modified-PCP are summarized in Section **??**. Numerical experiments, both for simulated data and for real video sequences, are shown in Section **??**. Conclusions and future work are discussed in Section **??**.

D. Notation

For a set $\mathcal{T} \subset \{1, 2, \dots, n\}$, we use $|\mathcal{T}|$ to denote its cardinality, i.e., the number of elements in $\mathcal{T}$. We use $\mathcal{T}^c$ to denote its complement w.r.t. $\{1, 2, \dots n\}$, i.e. $\mathcal{T}^c := \{i \in \{1, 2, \dots n\} : i \notin \mathcal{T}\}$.

We use the interval notation, $[t_1, t_2]$, to denote the set of all integers between and including t_1 to t_2, i.e. $[t_1, t_2] := \{t_1, t_1+1, \dots, t_2\}$. For a vector v, v_i denotes the ith entry of v and $v_\mathcal{T}$ denotes a vector consisting of the entries of v indexed by $\mathcal{T}$. We use $\|v\|_p$ to denote the ℓ_p norm of v. The support of v, supp(v), is the set of indices at which v is nonzero, supp$(v) := \{i : v_i \neq 0\}$. We say that v is s-sparse if $|\text{supp}(v)| \leq s$.

For a matrix B, B' denotes its transpose, and $B^\dagger$ its pseudo-inverse. For a matrix with linearly independent columns, $B^\dagger = (B'B)^{-1}B'$. We use $\|B\|_2 := \max_{x \neq 0} \|Bx\|_2 / \|x\|_2$ to denote the induced 2-norm of the matrix. Also, $\|B\|_*$ is the nuclear norm (sum of singular values) and $\|B\|_{\max}$ denotes the maximum over the absolute values of all its entries. We let $\sigma_i(B)$ denotes the ith largest singular value of B. For a Hermitian matrix, B, we use the notation $B \overset{EVD}{=} U \Lambda U'$ to denote the eigenvalue decomposition of B. Here U is an orthonormal matrix and Λ is a diagonal matrix with entries arranged in decreasing order. Also, we use $\lambda_i(B)$ to denote the ith largest eigenvalue of a Hermitian matrix B and we use $\lambda_{\max}(B)$ and $\lambda_{\min}(B)$ denote its maximum and minimum eigenvalues. If B is Hermitian positive semi-definite (p.s.d.), then $\lambda_i(B) = \sigma_i(B)$. For Hermitian matrices B_1 and B_2, the notation $B_1 \preceq B_2$ means that $B_2 - B_1$ is p.s.d. Similarly, $B_1 \succeq B_2$ means that $B_1 - B_2$ is p.s.d. For a Hermitian matrix B, $\|B\|_2 = \sqrt{\max(\lambda_{\max}^2(B), \lambda_{\min}^2(B))}$ and thus, $\|B\|_2 \leq b$ implies that $-b \leq \lambda_{\min}(B) \leq \lambda_{\max}(B) \leq b$.

We use the notation $B \overset{SVD}{=} U \Sigma V'$ to denote the singular value decomposition (SVD) of B with the diagonal entries of Σ being arranged in non-decreasing order.

We use I to denote an identity matrix of appropriate size. For an index set $\mathcal{T}$ and a matrix B, $B_\mathcal{T}$ is the sub-matrix of B containing columns with indices in the set $\mathcal{T}$. Notice that $B_\mathcal{T} = B I_\mathcal{T}$. Given a matrix B of size $m \times n$ and B_2 of size $m \times n_2$, $[B \ B_2]$ constructs a new matrix by concatenating matrices B and B_2 in the horizontal direction. Let B_{rem} be a matrix formed by some columns of B. Then $B \setminus B_{\text{rem}}$ is the matrix B with columns in B_{rem} removed.

The notation [.] denotes an empty matrix.

For a matrix M,

- $\text{span}(M)$ denotes the subspace spanned by the columns of M.
- M is a *basis matrix* if $M'M = I$.
- The *notation* $Q =$ *basis*$(\text{span}(M))$*, or* $Q =$ *basis*(M) *for short,* means that Q is a *basis matrix* for $\text{span}(M)$ i.e. Q satisfies $Q'Q = I$ and $\text{span}(Q) = \text{span}(M)$.
- The $b\%$ *left singular values' set* of a matrix M is the smallest set of indices of its singular values that contains at least $b\%$ of the total singular values' energy. In other words, if $M \overset{SVD}{=} U \Sigma V'$, it is the smallest set T such that $\sum_{i \in T} (\Sigma)_{i,i}^2 \geq \frac{b}{100} \sum_{i=1}^n (\Sigma)_{i,i}^2$.
- The corresponding matrix of left singular vectors, U_T, is referred to as the $b\%$ *left singular vectors' matrix.*
- The notation $[Q, \Sigma] =$ *approx-basis*$(M, b\%)$ means that Q is the $b\%$ left singular vectors' matrix for M and Σ is the diagonal matrix with diagonal entries equal to the b% left singular values' set.
- The notation $Q =$ *approx-basis*(M, r) means that Q contains the left singular vectors of M corresponding to its r largest singular values. This is also sometimes referred to as: Q *contains the* r *top singular vectors of* M.

Definition 1.1. *The* s-restricted isometry constant (RIC) *[**?**],* δ_s*, for an* $n \times m$ *matrix* Ψ *is the smallest real number satisfying* $(1-\delta_s)\|x\|_2^2 \leq \|\Psi_T x\|_2^2 \leq (1+\delta_s)\|x\|_2^2$ *for all sets* T *with* $|T| \leq s$ *and all real vectors* x *of length* $|T|$.

II. Problem Definition and Model Assumptions

We give the problem definition below followed by the model and then describe the key assumptions needed.

A. Problem Definition

The measurement vector at time t, M_t, is an n dimensional vector which can be decomposed as

$$M_t = L_t + S_t \tag{2}$$

Here S_t is a sparse vector with support set size at most s and minimum magnitude of nonzero values at least $S_{\min}$. L_t is a dense but low dimensional vector, i.e. $L_t = P_{(t)} a_t$ where $P_{(t)}$ is an $n \times r_{(t)}$ basis matrix with $r_{(t)} < n$, that changes every so often according to the model given below. We are given an accurate estimate of the subspace in which the initial t_{train} L_t's lie, i.e. we are given a basis matrix $\hat{P}_0$ so that $\|(I - \hat{P}_0 \hat{P}_0')P_0\|_2$ is small. Here P_0 is a basis matrix for $\text{span}(\mathcal{L}_{t_{\text{train}}})$, i.e. $\text{span}(P_0) = \text{span}(\mathcal{L}_{t_{\text{train}}})$. Also, for the first t_{train} time instants, S_t is zero. The goal is

1) to estimate both S_t and L_t at each time $t > t_{\text{train}}$, and
2) to estimate $\text{span}(\mathcal{L}_t)$ every so often, i.e. compute $\hat{P}_{(t)}$ so that the subspace estimation error, $\text{SE}_{(t)} := \|(I - \hat{P}_{(t)} \hat{P}_{(t)}')P_{(t)}\|_2$ is small.

We assume a subspace change model that allows the subspace to change at certain change times t_j rather than continuously at each time. It should be noted that this is only a model for reality. In practice there will typically be some changes at every time t; however this is difficult to model in a simple fashion. Moreover the analysis for such a model will be a lot more complicated. However, we do allow the variance of the projection of L_t along the subspace directions to change continuously. The projection along the new directions is assumed to be small initially and allowed to gradually increase to a large value (see Sec **??**).

Model 2.1 (Model on L_t)**.**

1) *We assume that* $L_t = P_{(t)} a_t$ *with* $P_{(t)} = P_j$ *for all* $t_j \leq t < t_{j+1}$*,* $j = 0, 1, 2 \cdots J$*. Here* P_j *is an* $n \times r_j$ *basis matrix with* $r_j < \min(n, (t_{j+1} - t_j))$ *that changes as*

$$P_j = [P_{j-1} \ P_{j,\text{new}}]$$

where $P_{j,\text{new}}$ *is a* $n \times c_{j,\text{new}}$ *basis matrix with* $P_{j,\text{new}}' P_{j-1} = 0$*. Thus*

$$r_j = \text{rank}(P_j) = r_{j-1} + c_{j,\text{new}}.$$

We let $t_0 = 0$*. Also* $t_{J+1} = t_{\max}$ *which is the length of the sequence. This can be infinite too. This model is illustrated in Figure* **??**.

2) *The vector of coefficients,* $a_t := {P_{(t)}}' L_t$*, is a zero mean random variable (r.v.) with mutually uncorrelated entries, i.e.* $\mathbf{E}[a_t] = 0$ *and* $\Lambda_t := \text{Cov}[a_t] = \mathbf{E}(a_t a_t')$ *is a diagonal matrix. This assumption follows automatically if we let* $P_{(t)} \Lambda_t {P_{(t)}}'$ *be the eigenvalue decomposition (EVD) of Cov*(L_t)*. .*

3) *Assume that* $0 < \lambda^- \leq \lambda^+ < \infty$ *where*

$$\lambda^- := \inf_t \lambda_{\min}(\Lambda_t), \quad \lambda^+ := \sup_t \lambda_{\max}(\Lambda_t)$$

Notice that, for $t_j \leq t < t_{j+1}$, a_t is an r_j length vector which can be split as

$$a_t = {P_j}' L_t = \begin{bmatrix} a_{t,*} \\ a_{t,\text{new}} \end{bmatrix}$$

where $a_{t,*} := {P_{j-1}}' L_t$ and $a_{t,\text{new}} := {P_{j,\text{new}}}' L_t$. Thus, for this interval, L_t can be rewritten as

$$L_t = [P_{j-1} \ P_{j,\text{new}}] \begin{bmatrix} a_{t,*} \\ a_{t,\text{new}} \end{bmatrix} = P_{j-1} a_{t,*} + P_{j,\text{new}} a_{t,\text{new}}$$

Also, Λ_t can be split as

$$\Lambda_t = \begin{bmatrix} \Lambda_{t,*} & 0 \\ 0 & \Lambda_{t,\text{new}} \end{bmatrix}$$

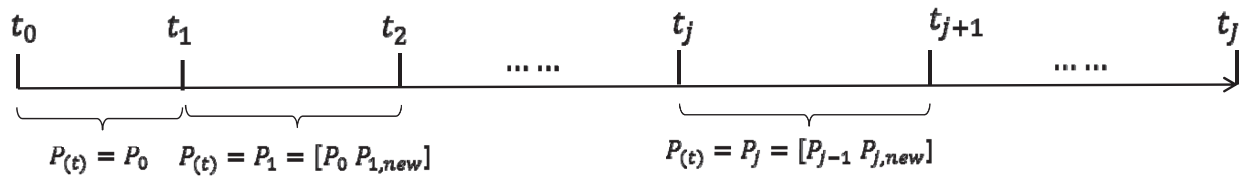

t_0
t_1
t_2
t_j
t_{j+1}
t_J
$P_{(t)} = P_0$
$P_{(t)} = P_1 = [P_0 \; P_{1,new}]$
$P_{(t)} = P_j = [P_{j-1} \; P_{j,new}]$

Fig. 1. The subspace change model explained in Sec **??**. Here $t_0 = 0$ and $0 < t_{\text{train}} < t_1$.

where $\Lambda_{t,*} = \text{Cov}[a_{t,*}]$ and $\Lambda_{t,\text{new}} = \text{Cov}[a_{t,\text{new}}]$ are diagonal matrices.

Definition 2.2. *Define*

$$f := \frac{\lambda^+}{\lambda^-}$$

Define

$$\lambda_{\text{new}}^- := \min_t \lambda_{\min}(\Lambda_{t,\text{new}}), \quad \lambda_{\text{new}}^+ := \max_t \lambda_{\max}(\Lambda_{t,\text{new}}), \quad g := \frac{\lambda_{\text{new}}^+}{\lambda_{\text{new}}^-}$$

Notice that the set of new *directions changes with each subspace change interval and hence at any time g is an upper bound on the condition number of $\Lambda_{t,\text{new}}$ for the currrent set of* new *directions.*

The above simple model only allows new additions to the subspace and hence the rank of P_j can only grow over time. The ReProCS algorithm designed for this model can be interpreted as a recursive algorithm for solving the robust PCA problem studied in [**?**] and other batch robust PCA works. At time t we estimate the subspace spanned by $L_1, L_2, \dots L_t$. We give this model first for ease of understanding. Both our algorithm and its guarantee will apply without any change even if one or more directions were deleted from the subspace with the following more general model.

Model 2.3. *Assume Model* **??** *with the difference that P_j now changes as*

$$P_j = [(P_{j-1}R_j \setminus P_{j,\text{old}}) \; P_{j,\text{new}}]$$

where $P_{j,\text{new}}$ is a $n \times c_{j,\text{new}}$ basis matrix with $P_{j,\text{new}}' P_{j-1} = 0$, R_j is a rotation matrix and $P_{j,\text{old}}$ is a $n \times c_{j,\text{old}}$ matrix containing columns of $(P_{j-1}R_j)$.

B. Slow subspace change

By slow subspace change we mean all of the following.

First, the delay between consecutive subspace change times is large enough, i.e., for a d large enough,

$$t_{j+1} - t_j \geq d \tag{3}$$

Second, the magnitude of the projection of L_t along the newly added directions, $a_{t,\text{new}}$, is initially small but can increase gradually. We model this as follows. Assume that

$$\lambda_{\text{new}}^+ \leq g^+ \lambda_{\text{new}}^- \tag{4}$$

for a $g^+ \geq 1$ but not too large; and for an $\alpha > 0$ [1] the following holds

$$\|a_{t,\text{new}}\|_\infty \leq \min\left(v^{\frac{t-t_j}{\alpha}-1}\gamma_{\text{new}}, \gamma_*\right) \tag{5}$$

when $t \in [t_j, t_{j+1} - 1]$ for a $v \geq 1$ but not too large and with $\gamma_{\text{new}} < \gamma_*$ and $\gamma_{\text{new}} < S_{\min}$. Clearly, the above assumption implies that

$$\|a_{t,\text{new}}\|_\infty \leq \gamma_{\text{new},k} := \min(v^{k-1}\gamma_{\text{new}}, \gamma_*)$$

for all $t \in [t_j + (k-1)\alpha, t_j + k\alpha - 1]$.

Third, the number of newly added directions is small, i.e. $c_{j,\text{new}} \leq c_{\max} \ll r_0$.

The last two assumptions are verified for real video data in [**?**, Section IX-A].

[1]As we will see in the algorithm α is the number of previous frames used to get a new estimate of $P_{j,\text{new}}$.

Remark 2.4 (Large f). *Since our problem definition allows large noise, L_t, but assumes slow subspace change, thus $f = \lambda^+/\lambda^-$ cannot be bounded by a small value. The reason is as follows. Slow subspace change implies that the projection of L_t along the new directions is initially small, i.e. γ_{new} is small. Since $\lambda^- \leq \gamma_{\text{new}}$, this means that λ^- is small. Since $\mathbf{E}[\|L_t\|^2] \leq r_{\max}\lambda^+$ and $r_{\max}$ is small (low-dimensional), thus, large L_t means that λ^+ needs to be large. As a result f cannot be upper bounded by a small value.*

C. Measuring denseness of a matrix and its relation with RIC

Before we can state the denseness assumption, we need to define the denseness coefficient.

Definition 2.5 (denseness coefficient). *For a matrix or a vector B, define*

$$\kappa_s(B) = \kappa_s(\text{span}(B)) := \max_{|T| \leq s} \|I_{\mathcal{T}}{}' \text{basis}(B)\|_2 \tag{6}$$

where $\|.\|_2$ is the vector or matrix ℓ_2-norm.

Clearly, $\kappa_s(B) \leq 1$. First consider an n-length vector B. Then κ_s measures the denseness (non-compressibility) of B. A small value indicates that the entries in B are spread out, i.e. it is a dense vector. A large value indicates that it is compressible (approximately or exactly sparse). The worst case (largest possible value) is $\kappa_s(B) = 1$ which indicates that B is an s-sparse vector. The best case is $\kappa_s(B) = \sqrt{s/n}$ and this will occur if each entry of B has the same magnitude. Similarly, for an $n \times r$ matrix B, a small κ_s means that most (or all) of its columns are dense vectors.

Remark 2.6. *The following facts should be noted about $\kappa_s(.)$:*

1) *For a given matrix B, $\kappa_s(B)$ is an non-decreasing function of s.*
2) *$\kappa_s([B_1]) \leq \kappa_s([B_1\ B_2])$ i.e. adding columns cannot decrease κ_s.*
3) *A bound on $\kappa_s(B)$ is $\kappa_s(B) \leq \sqrt{s}\kappa_1(B)$. This follows because $\|B\|_2 \leq \left\| \left[\|b_1\|_2 \dots \|b_r\|_2 \right] \right\|_2$ where b_i is the i^{th} column of B.*
4) *For a basis matrix $P = [P_1, P_2]$, $\kappa_s([P_1, P_2])^2 \leq \kappa_s(P_1)^2 + \kappa_s(P_2)^2$*

The lemma below relates the denseness coefficient of a basis matrix P to the RIC of $I - PP'$. The proof is in the Appendix.

Lemma 2.7. *For an $n \times r$ basis matrix P (i.e P satisfying $P'P = I$),*

$$\delta_s(I - PP') = \kappa_s^2(P).$$

In other words, if the columns of P are dense enough (small κ_s), then the RIC of $I - PP'$ is small.

In this work, we assume an upper bound on $\kappa_{2s}(P_j)$ for all j, and a tighter upper bound on $\kappa_{2s}(P_{j,\text{new}})$, i.e., there exist $\kappa_{2s,*}^+ < 1$ and a $\kappa_{2s,\text{new}}^+ < \kappa_{2s,*}^+$ such that

$$\max_j \kappa_{2s}(P_{j-1}) \leq \kappa_{2s,*}^+ \text{ and } \max_j \kappa_{2s}(P_{j,\text{new}}) \leq \kappa_{2s,\text{new}}^+ \tag{7}$$

The denseness coefficient $\kappa_s(B)$ is related to the denseness assumption required by PCP [**?**]. That work uses $\kappa_1(B)$ to quantify denseness. To relate to their condition, define the incoherence parameter μ as in [**?**] to be the smallest real number such that

$$\kappa_1(P_0)^2 \leq \frac{\mu r_0}{n} \quad \text{and} \quad \kappa_1(P_{t_j,\text{new}})^2 \leq \frac{\mu c_{j,\text{new}}}{n} \text{ for all } j = 1, 2, \dots J \tag{8}$$

Then, using Remark **??**, it is easy to see that (**??**) holds if

$$\frac{2s(r_0 + Jc_{\max})\mu}{n} \leq {\kappa_{2s,*}^+}^2 \text{ and } \frac{2sc_{\max}\mu}{n} \leq {\kappa_{2s,\text{new}}^+}^2 \tag{9}$$

where

$$c_{\max} := \max_j c_{j,\text{new}}.$$

D. Definitions and assumptions on the sparse outliers

Define the following quantities for the sparse outliers.

Definition 2.8. *Let $\mathcal{T}_t := \{i : (S_t)_i \neq 0\}$ denote the support set of S_t. Define*

$$S_{\min} := \min_{t > t_{train}} \min_{i \in \mathcal{T}_t} |(S_t)_i|, \text{ and } s := \max_t |\mathcal{T}_t|$$

In words, $S_{\min}$ is the magnitude of smallest nonzero entry of S_t for all t, and s is an upper bound on the support size of S_t for all t.

An upper bound on s is imposed by the denseness assumption given in Sec **??**. Moreover, we either need a denseness assumption on the basis matrix for the the currently unestimated part of $\operatorname{span}(P_{j,\text{new}})$ (a quantity that depends on intermediate algorithm estimates) or we need an assumption on how slowly the support can change (the support should change at least once every β time instants and when it changes, the change should be by at least s/ρ indices and be such that the sets of changed indices are mutually disjoint for a period of α frames). The former is needed in the partial result from our recently published work [**?**]. The latter is needed in our complete correctness result that will appear in the proceedings of ICASSP and ISIT 2015 [**?**], [**?**].

III. Recursive Projected Compressed Sensing (ReProCS) algorithm

In Sec **??**, we first explain the projection-PCA algorithm which is used for the subspace update step in the ReProCS algorithm. This is followed by a brief explanation of the main idea of the simplest possible ReProCS algorithm in Sec **??**. This algorithm is impractical though since it assumes knowledge of model parameters including the subspace change times. In Sec **??**, we develop a practical modification of the basic ReProCS idea that does not need model knowledge and can be directly used for experiments with real data. We further improve the sparse recovery and support estimation steps of this algorithm in Sec **??**.

A. Projection-PCA algorithm for ReProCS

Given a data matrix $\mathcal{D}$, a basis matrix P and an integer r, projection-PCA (proj-PCA) applies PCA on $\mathcal{D}_{\text{proj}} := (I - PP')\mathcal{D}$, i.e., it computes the top r eigenvectors (the eigenvectors with the largest r eigenvalues) of $\frac{1}{\alpha}\mathcal{D}_{\text{proj}}{\mathcal{D}_{\text{proj}}}'$. Here α is the number of column vectors in $\mathcal{D}$. This is summarized in Algorithm **??**.

If $P = [.]$, then projection-PCA reduces to standard PCA, i.e. it computes the top r eigenvectors of $\frac{1}{\alpha}\mathcal{D}\mathcal{D}'$.

The reason we need projection PCA algorithm in step 3 of Algorithm **??** is because the error $e_t = \hat{L}_t - L_t = S_t - \hat{S}_t$ is correlated with L_t; and the maximum condition number of $\operatorname{Cov}(L_t)$, which is bounded by f, cannot be bounded by a small value (see Remark **??**). This issue is explained in detail in the Appendix of [**?**]. Most other works that analyze standard PCA, e.g. [**?**] and references therein, do not face this issue because they assume uncorrelated-ness of the noise/error and the true data vector. With this assumption, one only needs to increase the PCA data length α to deal with the larger condition number.

We should mention that the idea of projecting perpendicular to a partly estimated subspace has been used in other different contexts in past work [**?**], [**?**].

Algorithm 1 projection-PCA: $Q \leftarrow \text{proj-PCA}(\mathcal{D}, P, r)$

```
1) Projection: compute $\mathcal{D}_{\text{proj}} \leftarrow (I - PP')\mathcal{D}$
2) PCA: compute $\frac{1}{\alpha}\mathcal{D}_{\text{proj}}{\mathcal{D}_{\text{proj}}}' \overset{EVD}{=} \begin{bmatrix} Q \; Q_\perp \end{bmatrix} \begin{bmatrix} \Lambda & 0 \\ 0 & \Lambda_\perp \end{bmatrix} \begin{bmatrix} Q' \\ {Q_\perp}' \end{bmatrix}$ where $Q$ is an $n \times r$ basis matrix and $\alpha$ is the number of columns in $\mathcal{D}$.
```

B. Recursive Projected CS (ReProCS)

We summarize the Recursive Projected CS (ReProCS) algorithm in Algorithm **??**. It uses the following definition.

Definition 3.1. *Define the time interval $\mathcal{I}_{j,k} := [t_j + (k-1)\alpha, t_j + k\alpha - 1]$ for $k = 1, \dots K$ and $\mathcal{I}_{j,K+1} := [t_j + K\alpha, t_{j+1} - 1]$.*

The key idea of ReProCS is as follows. First, consider a time t when the current basis matrix $P_{(t)} = P_{(t-1)}$ and this has been accurately predicted using past estimates of L_t, i.e. we have $\hat{P}_{(t-1)}$ with $\|(I - \hat{P}_{(t-1)}\hat{P}'_{(t-1)})P_{(t)}\|_2$ small. We project the measurement vector, M_t, into the space perpendicular to $\hat{P}_{(t-1)}$ to get the projected measurement vector $y_t := \Phi_{(t)} M_t$

where $\Phi_{(t)} = I - \hat{P}_{(t-1)}\hat{P}'_{(t-1)}$ (step 1a). Since the $n \times n$ projection matrix, $\Phi_{(t)}$ has rank $n - r_*$ where $r_* = \text{rank}(\hat{P}_{(t-1)})$, therefore y_t has only $n - r_*$ "effective" measurements[2], even though its length is n. Notice that y_t can be rewritten as

$$y_t = \Phi_{(t)} S_t + \boldsymbol{b}_t, \text{ where } \boldsymbol{b}_t := \Phi_{(t)} L_t.$$

Since $\|(I - \hat{P}_{(t-1)}\hat{P}'_{(t-1)})P_{(t-1)}\|_2$ is small, the projection nullifies most of the contribution of L_t and so the projected noise $\boldsymbol{b}_t$ is small. Recovering the n dimensional sparse vector S_t from y_t now becomes a traditional sparse recovery or CS problem in small noise [**?**], [**?**], [**?**], [**?**], [**?**], [**?**]. We use ℓ_1 minimization to recover it (step 1b). If the current basis matrix $P_{(t)}$, and hence its estimate, $\hat{P}_{(t-1)}$, is dense enough, then, by Lemma **??**, the RIC of $\Phi_{(t)}$ is small enough. Using [**?**, Theorem 1], this ensures that S_t can be accurately recovered from y_t. By thresholding on the recovered S_t, one gets an estimate of its support (step 1c). By computing a least squares (LS) estimate of S_t on the estimated support and setting it to zero everywhere else (step 1d), we can get a more accurate final estimate, $\hat{S}_t$, as first suggested in [**?**]. This $\hat{S}_t$ is used to estimate L_t as $\hat{L}_t = M_t - \hat{S}_t$. It is easy to see that if γ_{new} is small enough compared to $S_{\min}$ and the support estimation threshold, ω, is chosen appropriately, we can get exact support recovery, i.e. $\hat{\mathcal{T}}_t = \mathcal{T}_t$. In this case, the error $e_t := \hat{S}_t - S_t = L_t - \hat{L}_t$ has the following simple expression:

$$e_t = I_{\mathcal{T}_t} {(\Phi_{(t)})_{\mathcal{T}_t}}^{\dagger} \boldsymbol{b}_t = I_{\mathcal{T}_t} [(\Phi_{(t)})'_{\mathcal{T}_t} (\Phi_{(t)})_{\mathcal{T}_t}]^{-1} {I_{\mathcal{T}_t}}' \Phi_{(t)} L_t \tag{10}$$

The second equality follows because ${(\Phi_{(t)})_{\mathcal{T}}}' \Phi_{(t)} = (\Phi_{(t)} I_{\mathcal{T}})' \Phi_{(t)} = {I_{\mathcal{T}}}' \Phi_{(t)}$ for any set $\mathcal{T}$.

Now consider a time t when $P_{(t)} = P_j = [P_{j-1}, P_{j,\text{new}}]$ and P_{j-1} has been accurately estimated but $P_{j,\text{new}}$ has not been estimated, i.e. consider a $t \in \mathcal{I}_{j,1}$. At this time, $\hat{P}_{(t-1)} = \hat{P}_{j-1}$ and so $\Phi_{(t)} = \Phi_{j,0} := I - \hat{P}_{j-1}\hat{P}'_{j-1}$. Let $r_* := r_0 + (j-1)c_{\max}$ (we remove subscript j for ease of notation) and $c := c_{\max}$. Assume that the delay between change times is large enough so that by $t = t_j$, $\hat{P}_{j-1}$ is an accurate enough estimate of P_{j-1}, i.e. $\|\Phi_{j,0} P_{j-1}\|_2 \le r_* \zeta \ll 1$. Clearly, $\kappa_s(\Phi_0 P_{\text{new}}) \le \kappa_s(P_{\text{new}}) + r_* \zeta$, i.e. $\Phi_0 P_{\text{new}}$ is dense because P_{new} is dense and because $\hat{P}_{j-1}$ is an accurate estimate of P_{j-1} (which is perpendicular to P_{new}). Moreover, using Lemma **??**, it can be shown that $\phi_0 := \max_{|T| \le s} \|[(\Phi_0)'_T (\Phi_0)_T]^{-1}\|_2 \le \frac{1}{1-\delta_s(\Phi_0)} \le \frac{1}{1-(\kappa_s(P_{j-1}) + r_*\zeta)^2}$. The error e_t still satisfies (**??**) although its magnitude is not as small. Using the above facts in (**??**), we get that

$$\|e_t\|_2 \le \frac{\kappa_s(P_{\text{new}})\sqrt{c}\gamma_{\text{new}} + r_*\zeta(\sqrt{r_*}\gamma_* + \sqrt{c}\gamma_{\text{new}})}{1 - (\kappa_s(P_{j-1}) + r\zeta)^2}$$

If $\sqrt{\zeta} < 1/\gamma_*$, all terms containing ζ can be ignored and we get that the above is approximately upper bounded by $\frac{\kappa_s(P_{\text{new}})}{1-\kappa_s^2(P_{j-1})}\sqrt{c}\gamma_{\text{new}}$. Using the denseness assumption, this quantity is a small constant times $\sqrt{c}\gamma_{\text{new}}$, e.g. with the numbers assumed in Theorem **??** given later, we get a bound of $0.18\sqrt{c}\gamma_{\text{new}}$. Because of slow subspace change, c and γ_{new} are small and hence the error e_t is small, i.e. S_t is recovered accurately. With each projection PCA step, as we explain below, the error e_t becomes even smaller.

Since $\hat{L}_t = M_t - \hat{S}_t$ (step 2), e_t also satisfies $e_t = L_t - \hat{L}_t$. Thus, a small e_t means that L_t is also recovered accurately. The estimated $\hat{L}_t$'s are used to obtain new estimates of $P_{j,\text{new}}$ every α frames for a total of $K\alpha$ frames by projection PCA (step 3). We illustrate the projection PCA algorithm in Figure **??**. In the first projection PCA step, we get the first estimate of $P_{j,\text{new}}$, $\hat{P}_{j,\text{new},1}$. For the next α frame interval, $\hat{P}_{(t-1)} = [\hat{P}_{j-1}, \hat{P}_{j,\text{new},1}]$ and so $\Phi_{(t)} = \Phi_{j,1} = I - \hat{P}_{j-1}\hat{P}'_{j-1} - \hat{P}_{\text{new},1}\hat{P}'_{\text{new},1}$. Using this in the projected CS step reduces the projection noise, $\boldsymbol{b}_t$, and hence the reconstruction error, e_t, for this interval, as long as $\gamma_{\text{new},k}$ increases slowly enough. Smaller e_t makes the perturbation seen by the second projection PCA step even smaller, thus resulting in an improved second estimate $\hat{P}_{j,\text{new},2}$. Within K updates (K chosen as given in Theorem **??**), it can be shown that both $\|e_t\|_2$ and the subspace error drop down to a constant times $\sqrt{\zeta}$. At this time, we update $\hat{P}_j$ as $\hat{P}_j = [\hat{P}_{j-1}, \hat{P}_{j,\text{new},K}]$.

C. Practical ReProCS

The algorithm described in the previous subsection and in Algorithm **??** cannot be used in practice because it assumes knowledge of the subspace change times t_j and of the exact number of new directions added at each change time, $c_{j,\text{new}}$. Moreover, the choices of the algorithm parameters are also impractically large. In this section, we develop a practical modification of the ReProCS algorithm from the previous section that does not assume any model knowledge and that also directly works for various video sequences. The algorithm is summarized in Algorithm **??**. We use the YALL-1 solver for solving all ℓ_1 norm minimization problems. Code for Algorithm **??** is available at http://www.ece.iastate.edu/~hanguo/PracReProCS.html#Code_.

Given the initial training sequence which does not contain the sparse components, $\mathcal{M}_{\text{train}} = [L_1, L_2, \ldots L_{t_{\text{train}}}]$ we compute $\hat{P}_0$ as an approximate basis for $\mathcal{M}_{\text{train}}$, i.e. $\hat{P}_0 = \text{approx-basis}(\mathcal{M}_{\text{train}}, b\%)$. Let $\hat{r} = \text{rank}(\hat{P}_0)$. We need to compute an approximate basis because for real data, the L_t's are only approximately low-dimensional. We use $b\% = 95\%$ or $b\% = 99.99\%$ depending

[2]i.e. some r_* entries of y_t are linear combinations of the other $n - r_*$ entries

Algorithm 2 Recursive Projected CS (ReProCS)

```
*Parameters:* algorithm parameters: $\xi$, $\omega$, $\alpha$, $K$, model parameters: $t_j$, $c_{j,\text{new}}$
(set as in Theorem **??** )
*Input:* $M_t$, *Output:* $\hat{S}_t$, $\hat{L}_t$, $\hat{P}_{(t)}$
Initialization: Compute $\hat{P}_0 \leftarrow$ proj-PCA$([L_1, L_2, \cdots, L_{t_{\text{train}}}], [.], r_0)$ where $r_0 = \text{rank}([L_1, L_2, \cdots, L_{t_{\text{train}}}])$.
Set $\hat{P}_{(t)} \leftarrow \hat{P}_0$, $j \leftarrow 1$, $k \leftarrow 1$.
For $t > t_{\text{train}}$, do the following:
  1) Estimate $\mathcal{T}_t$ and $S_t$ via Projected CS:
     a) Nullify most of $L_t$: compute $\Phi_{(t)} \leftarrow I - \hat{P}_{(t-1)}\hat{P}'_{(t-1)}$, compute $y_t \leftarrow \Phi_{(t)} M_t$
     b) Sparse Recovery: compute $\hat{S}_{t,\text{cs}}$ as the solution of $\min_x \|x\|_1 \ s.t. \ \|y_t - \Phi_{(t)}x\|_2 \leq \xi$
     c) Support Estimate: compute $\hat{\mathcal{T}}_t = \{i : \ |(\hat{S}_{t,\text{cs}})_i| > \omega\}$
     d) LS Estimate of $S_t$: compute $(\hat{S}_t)_{\hat{\mathcal{T}}_t} = ((\Phi_{(t)})_{\hat{\mathcal{T}}_t})^{\dagger} y_t, \ (\hat{S}_t)_{\hat{\mathcal{T}}_t^c} = 0$
  2) Estimate $L_t$: $\hat{L}_t = M_t - \hat{S}_t$.
  3) Update $\hat{P}_{(t)}$: K Projection PCA steps.
     a) If $t = t_j + k\alpha - 1$,
        i) $\hat{P}_{j,\text{new},k} \leftarrow$ proj-PCA$\left(\left[\hat{L}_{t_j+(k-1)\alpha}, \ldots, \hat{L}_{t_j+k\alpha-1}\right], \hat{P}_{j-1}, c_{j,\text{new}}\right)$.
        ii) set $\hat{P}_{(t)} \leftarrow [\hat{P}_{j-1} \ \hat{P}_{j,\text{new},k}]$; increment $k \leftarrow k+1$.
        Else
        i) set $\hat{P}_{(t)} \leftarrow \hat{P}_{(t-1)}$.
     b) If $t = t_j + K\alpha - 1$, then set $\hat{P}_j \leftarrow [\hat{P}_{j-1} \ \hat{P}_{j,\text{new},K}]$. Increment $j \leftarrow j+1$. Reset $k \leftarrow 1$.
  4) Increment $t \leftarrow t+1$ and go to step 1.
```

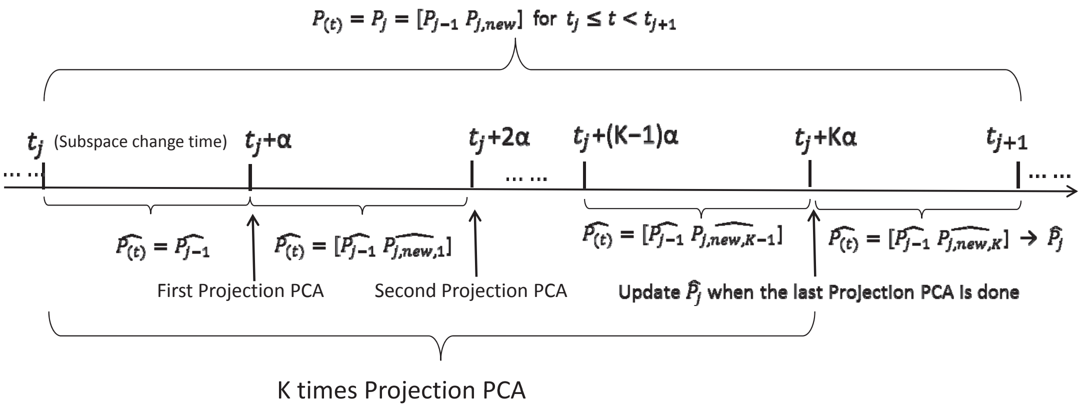

$P_{(t)} = P_j = [P_{j-1} \ P_{j,new}]$ for $t_j \leq t < t_{j+1}$
t_j (Subspace change time)
$t_j+\alpha$
$t_j+2\alpha$
$t_j+(K-1)\alpha$
$t_j+K\alpha$
t_{j+1}
$\widehat{P_{(t)}} = \widehat{P_{j-1}}$
$\widehat{P_{(t)}} = [\widehat{P_{j-1}} \ \widehat{P_{j,new,1}}]$
$\widehat{P_{(t)}} = [\widehat{P_{j-1}} \ \widehat{P_{j,new,K-1}}]$
$\widehat{P_{(t)}} = [\widehat{P_{j-1}} \ \widehat{P_{j,new,K}}] \rightarrow \hat{P}_j$
First Projection PCA
Second Projection PCA
Update $\hat{P}_j$ when the last Projection PCA is done
K times Projection PCA

Fig. 2. The K projection PCA steps.

on whether the low-rank part is approximately low-rank or almost exactly low-rank. After this, at each time t, ReProCS proceeds as follows.

Perpendicular Projection. In the first step, at time t, we project the measurement vector, M_t, into the space orthogonal to $\text{span}(\hat{P}_{(t-1)})$ to get the projected measurement vector,

$$y_t := \Phi_{(t)} M_t. \tag{11}$$

Sparse Recovery (Recover $\mathcal{T}_t$ and S_t). With the above projection, y_t can be rewritten as

$$y_t = \Phi_{(t)} S_t + \boldsymbol{b}_t \text{ where } \boldsymbol{b}_t := \Phi_{(t)} L_t \tag{12}$$

As explained earlier, because of the slow subspace change assumption, projecting orthogonal to $\text{span}(\hat{P}_{(t-1)})$ nullifies most of the contribution of L_t and hence $\boldsymbol{b}_t$ can be interpreted as small "noise". Thus, the problem of recovering S_t from y_t becomes a traditional noisy sparse recovery / CS problem. To recover S_t from y_t, one can use ℓ_1 minimization [**?**], [**?**], or any of the greedy or iterative thresholding algorithms from literature. In this work we use ℓ_1 minimization: we solve

$$\min_x \|x\|_1 \text{ s.t. } \|y_t - \Phi_{(t)} x\|_2 \leq \xi \tag{13}$$

and denote its solution by $\hat{S}_{t,cs}$. By the denseness assumption, P_{t-1} is dense. Since $\hat{P}_{(t-1)}$ approximates it, this is true for $\hat{P}_{(t-1)}$ as well [?, Lemma 6.6]. Thus, by Lemma ??, the RIC of $\Phi_{(t)}$ is small enough. Using [?, Theorem 1], this and the fact that $\boldsymbol{b}_t$ is small ensures that S_t can be accurately recovered from y_t. The constraint ξ used in the minimization should equal $\|\boldsymbol{b}_t\|_2$ or its upper bound. Since $\boldsymbol{b}_t$ is unknown we set $\xi = \|\hat{\beta}_t\|_2$ where $\hat{\beta}_t := \Phi_{(t)} \hat{L}_{t-1}$.

By thresholding on $\hat{S}_{t,cs}$ to get an estimate of its support followed by computing a least squares (LS) estimate of S_t on the estimated support and setting it to zero everywhere else, we can get a more accurate estimate, $\hat{S}_t$, as suggested in [?]. We discuss better support estimation and its parameter setting in Sec **??**.

Recover L_t. The estimate $\hat{S}_t$ is used to estimate L_t as $\hat{L}_t = M_t - \hat{S}_t$. Thus, if S_t is recovered accurately, so will L_t.

Subspace Update (Update $\hat{P}_{(t)}$). Within a short delay after every subspace change time, one needs to update the subspace estimate, $\hat{P}_{(t)}$. To do this in a provably reliable fashion, we introduced the projection PCA (p-PCA) algorithm [?] in the previous two subsections. In this section, we design a practical version of projection-PCA (p-PCA) which does not need knowledge of t_j or $c_{j,\text{new}}$. This is summarized in Algorithm **??**. The key idea is as follows. We let $\hat{\sigma}_{\min}$ be the $\hat{r}^{th}$ largest singular value of the training dataset. This serves as the noise threshold for approximately low rank data. We split projection PCA into two phases: "detect subspace change" and "p-PCA". We are in the detect phase when the previous subspace has been accurately estimated. Denote the basis matrix for this subspace by $\hat{P}_{j-1}$. We detect the subspace change as follows. Every α frames, we project the last α $\hat{L}_t$'s perpendicular to $\hat{P}_{j-1}$ and compute the SVD of the resulting matrix. If there are any singular values above $\hat{\sigma}_{\min}$, this means that the subspace has changed. At this point, we enter the "p-PCA" phase. In this phase, we repeat the K p-PCA steps described earlier with the following change: we estimate $c_{j,\text{new}}$ as the number of singular values above $\hat{\sigma}_{\min}$, but clipped at $\lceil \alpha/3 \rceil$ (i.e. if the number is more than $\lceil \alpha/3 \rceil$ then we clip it to $\lceil \alpha/3 \rceil$). We stop either when the stopping criterion given in step **??** is achieved ($k \geq K_{\min}$ and the projection of $\hat{L}_t$ along $\hat{P}_{\text{new},k}$ is not too different from that along $\hat{P}_{\text{new},k}$) or when $k \geq K_{\max}$.

Remark 3.2. *The p-PCA algorithm only allows addition of new directions. If the goal is to estimate the span of $[L_1, \ldots L_t]$, then this is what is needed. If the goal is sparse recovery, then one can get a smaller rank estimate of $\hat{P}_{(t)}$ by also including a step to delete the span of the removed directions, $P_{(j),\text{old}}$. This will result in more "effective" measurements available for the sparse recovery step and hence possibly in improved performance. The simplest way to do this is to by simple PCA done at the end of the K p-PCA steps. A provably accurate solution is the cluster-PCA approach described in [?, Sec VII].*

Remark 3.3. *The p-PCA algorithm works on small batches of α frames. This can be made fully recursive if we compute the SVD needed in projection-PCA using the incremental SVD [?] procedure summarized in Algorithm ?? (inc-SVD) for one frame at a time. As explained in [?] and references therein, we can get the left singular vectors and singular values of any matrix $M = [M_1, M_2, \ldots M_\alpha]$ recursively by starting with $\hat{P} = [.], \hat{\Sigma} = [.]$ and calling [$\hat{P}$, $\hat{\Sigma}$] = inc-SVD($\hat{P}$, $\hat{\Sigma}$, M_i) for every column i or for short batches of columns of size of α/k. Since we use $\alpha = 20$ which is a small value, the use of incremental SVD does not speed up the algorithm in practice.*

D. Exploiting slow support change when valid and improved two-step support estimation

In [?], [?], we always used ℓ_1 minimization followed by thresholding and LS for sparse recovery. However if slow support change holds, one can replace simple ℓ_1 minimization by modified-CS [?] which requires fewer measurements for exact/accurate recovery as long as the previous support estimate, $\hat{\mathcal{T}}_{t-1}$, is an accurate enough predictor of the current support, $\mathcal{T}_t$. In our application, $\hat{\mathcal{T}}_{t-1}$ is likely to contain a significant number of extras and in this case, a better idea is to solve the following weighted ℓ_1 problem [?], [?]

$$\min_x \lambda \|x_{\mathcal{T}}\|_1 + \|x_{T^c}\|_1 \text{ s.t. } \|y_t - \Phi_{(t)} x\|_2 \leq \xi, \quad T := \hat{\mathcal{T}}_{t-1} \tag{14}$$

with $\lambda < 1$ (modified-CS solves the above with $\lambda = 0$). Denote its solution by $\hat{S}_{t,cs}$. One way to pick λ is to let it be proportional to the estimate of the percentage of extras in $\hat{\mathcal{T}}_{t-1}$. If slow support change does not hold, the previous support estimate is not a good predictor of the current support. In this case, doing the above is a bad idea and one should instead solve simple ℓ_1, i.e. solve (**??**) with $\lambda = 1$. As explained in [?], if the support estimate contains at least 50% correct entries, then weighted ℓ_1 is better than simple ℓ_1. We use the above criteria with true values replaced by estimates. Thus, if $\frac{|\hat{\mathcal{T}}_{t-2} \cap \hat{\mathcal{T}}_{t-1}|}{|\hat{\mathcal{T}}_{t-2}|} > 0.5$, then we solve (**??**) with $\lambda = \frac{|\hat{\mathcal{T}}_{t-2} \setminus \hat{\mathcal{T}}_{t-1}|}{|\hat{\mathcal{T}}_{t-1}|}$, else we solve it with $\lambda = 1$.

1) Improved support estimation: A simple way to estimate the support is by thresholding the solution of (**??**). This can be improved by using the Add-LS-Del procedure for support and signal value estimation [**?**]. We proceed as follows. First we compute the set $\hat{\mathcal{T}}_{t,\text{add}}$ by thresholding on $\hat{S}_{t,cs}$ in order to retain its k largest magnitude entries. We then compute a LS estimate of S_t on $\hat{\mathcal{T}}_{t,\text{add}}$ while setting it to zero everywhere else. As explained earlier, because of the LS step, $\hat{S}_{\text{t,add}}$ is a less biased estimate of S_t than $\hat{S}_{t,cs}$. We let $k = 1.4|\hat{\mathcal{T}}_{t-1}|$ to allow for a small increase in the support size from $t-1$ to t. A larger value of k also makes it more likely that elements of the set $(\mathcal{T}_t \setminus \hat{\mathcal{T}}_{t-1})$ are detected into the support estimate[3].

The final estimate of the support, $\hat{\mathcal{T}}_t$, is obtained by thresholding on $\hat{S}_{\text{t,add}}$ using a threshold ω. If ω is appropriately chosen, this step helps to delete some of the extra elements from $\hat{\mathcal{T}}_{\text{add}}$ and this ensures that the size of $\hat{\mathcal{T}}_t$ does not keep increasing (unless the object's size is increasing). An LS estimate computed on $\hat{\mathcal{T}}_t$ gives us the final estimate of S_t, i.e. $\hat{S}_t = \text{LS}(y_t, A, \hat{\mathcal{T}}_t)$. We use $\omega = \sqrt{\|M_t\|^2/n}$ except in situations where $\|S_t\| \ll \|L_t\|$ - in this case we use $\omega = 0.25\sqrt{\|M_t\|^2/n}$. An alternate approach is to let ω be proportional to the noise magnitude seen by the ℓ_1 step, i.e. to let $\omega = q\|\hat{\beta}_t\|_\infty$, however this approach required different values of q for different experiments (it is not possible to specify one q that works for all experiments).

The complete algorithm with all the above steps is summarized in Algorithm **??**.

IV. Modified-PCP: a piecewise batch online robust PCA solution

A key limitation of ReProCS is that it does not use the fact that the noise seen by the sparse recovery step of ReProCS, $\boldsymbol{b}_t$, approximately lies in a very low dimensional subspace. To understand this better, consider a time t just after a subspace change time, t_j. Assume that before t_j the previous subspace P_{j-1} has been accurately estimated. For $t \in [t_j, t_j+\alpha)$, $\boldsymbol{b}_t$ can be written as

$$\boldsymbol{b}_t = P_{j,\text{new}} a_{t,\text{new}} + w_t, \text{ where } w_t := (I - \hat{P}_{j-1}\hat{P}_{j-1}{}')P_{j-1}a_{t,*} + \hat{P}_{j-1}\hat{P}_{j-1}' P_{j,\text{new}} a_{t,\text{new}}$$

Conditioned on accurate recovery so far, w_t is small noise because $\|(I - \hat{P}_{j-1}\hat{P}_{j-1}{}')P_{j-1}\|_2 \le r\zeta$ and $\|\hat{P}_{j-1}\hat{P}_{j-1}' P_{j,\text{new}}\|_2 \le r\zeta$. Thus, $\boldsymbol{b}_t$ approximately lies in a $c_{j,\text{new}} \le c_{\max}$ dimensional subspace. Here $c_{\max}$ is the maximum number of newly added directions and by assumption $c_{\max} \ll r \ll n$.

The only way to use the above fact is in a piecewise batch fashion (one cannot impose a low-dimensional subspace assumption on a single vector). The resulting approach can be understood as a modification of the PCP idea that uses an idea similar to our older work on modified-CS (for sparse recovery with partial support knowledge) [**?**] and hence we call it *modified-PCP*. We introduced and studied modified-PCP in [**?**], [**?**]

A. Problem re-formulation: robust PCA with partial subspace knowledge

To understand our solution, one can reformulate the online robust PCA problem as a problem of robust PCA with partial subspace knowledge $\hat{P}_{j-1}$. To keep notation simple, we use $\mathbf{G}$ to denote the partial subspace knowledge. With this we have the following problem.

We are given a data matrix $\mathbf{M} \in \mathbb{R}^{n_1 \times n_2}$ that satisfies

$$\mathbf{M} = \mathbf{L} + \mathbf{S} \tag{15}$$

where $\mathbf{S}$ is a sparse matrix with support set Ω and $\mathbf{L}$ is a low rank matrix with reduced singular value decomposition (SVD)

$$\mathbf{L} \overset{\text{SVD}}{=} \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}' \tag{16}$$

Let $r := \operatorname{rank}(\mathbf{L})$. We assume that we are given a basis matrix $\mathbf{G}$ so that $(\mathbf{I} - \mathbf{G}\mathbf{G}')\mathbf{L}$ has rank smaller than r. The goal is to recover $\mathbf{L}$ and $\mathbf{S}$ from $\mathbf{M}$ using $\mathbf{G}$. Let $r_G := \operatorname{rank}(\mathbf{G})$.

Define $\mathbf{L}_{\text{new}} := (\mathbf{I} - \mathbf{G}\mathbf{G}')\mathbf{L}$ with $r_{\text{new}} := \operatorname{rank}(\mathbf{L}_{\text{new}})$ and reduced SVD given by

$$\mathbf{L}_{\text{new}} := (\mathbf{I} - \mathbf{G}\mathbf{G}')\mathbf{L} \overset{\text{SVD}}{=} \mathbf{U}_{\text{new}}\boldsymbol{\Sigma}_{\text{new}}\mathbf{V}_{\text{new}}' \tag{17}$$

We explain this a little more. With the above, it is easy to show that there exist rotation matrices $\mathbf{R}_U, \mathbf{R}_G$, and basis matrices $\mathbf{G}_{\text{extra}}$ and $\mathbf{U}_{\text{new}}$ with $\mathbf{G}_{\text{extra}}{}'\mathbf{U}_{\text{new}} = 0$, such that

$$\mathbf{U} = [\underbrace{(\mathbf{G}\mathbf{R}_G \setminus \mathbf{G}_{\text{extra}})}_{\mathbf{U}_0}\ \mathbf{U}_{\text{new}}]\mathbf{R}_U'. \tag{18}$$

Define $r_0 := \operatorname{rank}(\mathbf{U}_0)$ and $r_{\text{extra}} := \operatorname{rank}(\mathbf{G}_{\text{extra}})$. Clearly, $r_G = r_0 + r_{\text{extra}}$ and $r = r_0 + r_{\text{new}} = (r_G - r_{\text{extra}}) + r_{\text{new}}$.

[3]Due to the larger weight on the $\|x_{(\hat{\mathcal{T}}_{t-1}^c)}\|_1$ term as compared to that on the $\|x_{(\hat{\mathcal{T}}_{t-1})}\|_1$ term, the solution of (**??**) is biased towards zero on $\hat{\mathcal{T}}_{t-1}^c$ and thus the solution values along $(\mathcal{T}_t \setminus \hat{\mathcal{T}}_{t-1})$ are smaller than the true ones.

Algorithm 3 Practical ReProCS

```
Input: $M_t$; Output: $\hat{\mathcal{T}}_t$, $\hat{S}_t$, $\hat{L}_t$; Parameters: $q, b, \alpha, K_{\min}, K_{\max}$. We used $\alpha = 20, K_{\min} = 3, K_{\max} = 10$ in all experiments ($\alpha$ needs to only be large compared to $c_{\max}$); we used $b = 95$ for approximately low-rank data (all real videos and the lake video with simulated foreground) and used $b = 99.99$ for almost exactly low rank data (simulated data); we used $q = 1$ whenever $\|S_t\|_2$ was of the same order or larger than $\|L_t\|_2$ (all real videos and the lake video) and used $q = 0.25$ when it was much smaller (simulated data with small magnitude $S_t$).
Initialization
  • $[\hat{P}_0, \hat{\Sigma}_0] \leftarrow$ approx-basis($\frac{1}{\sqrt{t_{\text{train}}}}[M_1, \dots M_{t_{\text{train}}}], b\%$).
  • Set $\hat{r} \leftarrow \text{rank}(\hat{P}_0)$, $\hat{\sigma}_{\min} \leftarrow ((\hat{\Sigma}_0)_{\hat{r},\hat{r}})$, $\hat{t}_0 = t_{\text{train}}$, flag = detect
  • Initialize $\hat{P}_{(t_{\text{train}})} \leftarrow \hat{P}_0$ and $\hat{\mathcal{T}}_t \leftarrow [.]$.
For $t > t_{\text{train}}$ do
  1) Perpendicular Projection: compute $y_t \leftarrow \Phi_{(t)} M_t$ with $\Phi_{(t)} \leftarrow I - \hat{P}_{(t-1)} \hat{P}'_{(t-1)}$
  2) Sparse Recovery (Recover $S_t$ and $\mathcal{T}_t$)
     a) If $\frac{|\hat{\mathcal{T}}_{t-2} \cap \hat{\mathcal{T}}_{t-1}|}{|\hat{\mathcal{T}}_{t-2}|} < 0.5$
          i) Compute $\hat{S}_{t,\text{cs}}$ as the solution of (??) with $\xi = \|\Phi_{(t)} \hat{L}_{t-1}\|_2$.
         ii) $\hat{\mathcal{T}}_t \leftarrow \text{Thresh}(\hat{S}_{t,\text{cs}}, \omega)$ with $\omega = q\sqrt{\|M_t\|^2/n}$. Here $\mathcal{T} \leftarrow \text{Thresh}(x, \omega)$ means that $\mathcal{T} = \{i : \ |(x)_i| \geq \omega\}$.
        Else
          i) Compute $\hat{S}_{t,\text{cs}}$ as the solution of (??) with $\mathcal{T} = \hat{\mathcal{T}}_{t-1}$, $\lambda = \frac{|\hat{\mathcal{T}}_{t-2} \setminus \hat{\mathcal{T}}_{t-1}|}{|\hat{\mathcal{T}}_{t-1}|}$, $\xi = \|\Phi_{(t)} \hat{L}_{t-1}\|_2$.
         ii) $\hat{\mathcal{T}}_{\text{add}} \leftarrow \text{Prune}(\hat{S}_{t,\text{cs}}, 1.4|\hat{\mathcal{T}}_{t-1}|)$. Here $\mathcal{T} \leftarrow \text{Prune}(x, k)$ returns indices of the $k$ largest magnitude elements of $x$.
        iii) $\hat{S}_{t,\text{add}} \leftarrow \text{LS}(y_t, \Phi_{(t)}, \hat{\mathcal{T}}_{\text{add}})$. Here $\hat{x} \leftarrow \text{LS}(y, A, T)$ means that $\hat{x}_{\mathcal{T}} = (A_{\mathcal{T}}{}' A_{\mathcal{T}})^{-1} A_{\mathcal{T}}{}' y$ and $\hat{x}_{\mathcal{T}^c} = 0$.
         iv) $\hat{\mathcal{T}}_t \leftarrow \text{Thresh}(\hat{S}_{t,\text{add}}, \omega)$ with $\omega = q\sqrt{\|M_t\|^2/n}$.
     b) $\hat{S}_t \leftarrow \text{LS}(y_t, \Phi_{(t)}, \hat{\mathcal{T}}_t)$
  3) Estimate $L_t$: $\hat{L}_t \leftarrow M_t - \hat{S}_t$
  4) Update $\hat{P}_{(t)}$: projection PCA
     a) If flag = detect and $\text{mod}(t - \hat{t}_j + 1, \alpha) = 0$, (here $\text{mod}(t, \alpha)$ is the remainder when $t$ is divided by $\alpha$)
          i) compute the SVD of $\frac{1}{\sqrt{\alpha}}(I - \hat{P}_{j-1} \hat{P}'_{j-1})[\hat{L}_{t-\alpha+1}, \dots \hat{L}_t]$ and check if any singular values are above $\hat{\sigma}_{\min}$
         ii) if the above number is more than zero then set flag $\leftarrow$ pPCA, increment $j \leftarrow j + 1$, set $\hat{t}_j \leftarrow t - \alpha + 1$, reset $k \leftarrow 1$
        Else $\hat{P}_{(t)} \leftarrow \hat{P}_{(t-1)}$.
     b) If flag = pPCA and $\text{mod}(t - \hat{t}_j + 1, \alpha) = 0$,
          i) compute the SVD of $\frac{1}{\sqrt{\alpha}}(I - \hat{P}_{j-1} \hat{P}'_{j-1})[\hat{L}_{t-\alpha+1}, \dots \hat{L}_t]$,
         ii) let $\hat{P}_{j,\text{new},k}$ retain all its left singular vectors with singular values above $\hat{\sigma}_{\min}$ or all $\alpha/3$ top left singular vectors whichever is smaller,
        iii) update $\hat{P}_{(t)} \leftarrow [\hat{P}_{j-1} \ \hat{P}_{j,\text{new},k}]$, increment $k \leftarrow k + 1$
         iv) If $k \geq K_{\min}$ and $\frac{\|\sum_{t-\alpha+1}^{t} (\hat{P}_{j,\text{new},i-1} \hat{P}'_{j,\text{new},i-1} - \hat{P}_{j,\text{new},i} \hat{P}'_{j,\text{new},i}) L_t\|_2}{\|\sum_{t-\alpha+1}^{t} \hat{P}_{j,\text{new},i-1} \hat{P}'_{j,\text{new},i-1} L_t\|_2} < 0.01$ for $i = k-2, k-1, k$; or $k = K_{\max}$, then $K \leftarrow k$, $\hat{P}_j \leftarrow [\hat{P}_{j-1} \ \hat{P}_{j,\text{new},K}]$ and reset flag $\leftarrow$ detect.
        Else $\hat{P}_{(t)} \leftarrow \hat{P}_{(t-1)}$.
```

B. Modified-PCP

From the above model, it is clear that

$$\mathbf{L}_{\text{new}} + \mathbf{G}\mathbf{X}' + \mathbf{S} = \mathbf{M} \tag{19}$$

for $\mathbf{X} = \mathbf{L}'\mathbf{G}$. We recover $\mathbf{L}$ and $\mathbf{S}$ using $\mathbf{G}$ by solving the following **Modified PCP** (mod-PCP) program

$$\begin{array}{ll} \text{minimize}_{\tilde{\mathbf{L}}_{\text{new}}, \tilde{\mathbf{S}}, \tilde{\mathbf{X}}} & \|\tilde{\mathbf{L}}_{\text{new}}\|_* + \lambda \|\tilde{\mathbf{S}}\|_1 \\ \text{subject to} & \tilde{\mathbf{L}}_{\text{new}} + \mathbf{G}\tilde{\mathbf{X}}' + \tilde{\mathbf{S}} = \mathbf{M} \end{array} \tag{20}$$

Denote a solution to the above by $\hat{\mathbf{L}}_{\text{new}}, \hat{\mathbf{S}}, \hat{\mathbf{X}}$. Then, $\mathbf{L}$ is recovered as $\hat{\mathbf{L}} = \hat{\mathbf{L}}_{\text{new}} + \mathbf{G}\hat{\mathbf{X}}'$. Modified-PCP is inspired by our recent work on sparse recovery using partial support knowledge called modified-CS [?]. Notice that modified-PCP is equivalent

Algorithm 4 $[\hat{P}, \hat{\Sigma}]$ = inc-SVD($\hat{P}$, $\hat{\Sigma}$, D)

```
1) set $D_{\|,proj} \leftarrow \hat{P}'D$ and $D_\perp \leftarrow (I - \hat{P}\hat{P}')D$
2) compute QR decomposition of $D_\perp$, i.e. $D_\perp \overset{QR}{=} JK$ (here $J$ is a basis matrix and $K$ is an upper triangular matrix)
3) compute the SVD: $\begin{bmatrix} \hat{\Sigma} & D_{\|,proj} \\ 0 & K \end{bmatrix} \overset{SVD}{=} \tilde{P}\tilde{\Sigma}\tilde{V}'$
4) update $\hat{P} \leftarrow [\hat{P}\ J]\tilde{P}$ and $\hat{\Sigma} \leftarrow \tilde{\Sigma}$
```

Note: As explained in [**?**], due to numerical errors, step 4 done too often can eventually result in $\hat{P}$ no longer being a basis matrix. This typically occurs when one tries to use inc-SVD at every time t, i.e. when D is a column vector. This can be addressed using the modified Gram Schmidt re-orthonormalization procedure whenever loss of orthogonality is detected [**?**].

to the following

$$\begin{array}{ll} \text{minimize}_{\tilde{\mathbf{L}},\tilde{\mathbf{S}}} & \|(I - \mathbf{G}\mathbf{G}')\tilde{\mathbf{L}}\|_* + \lambda\|\tilde{\mathbf{S}}\|_1 \\ \text{subject to} & \tilde{\mathbf{L}} + \tilde{\mathbf{S}} = \mathbf{M} \end{array} \tag{21}$$

and the above is easily understood as being inspired by modified-CS.

C. Solving the modified-PCP program

We give below an algorithm based on the Inexact Augmented Lagrange Multiplier (ALM) method [**?**] to solve the modified-PCP program, i.e. solve (**??**). This algorithm is a direct modification of the algorithm designed to solve PCP in [**?**]. The only difference is that it uses the idea of [**?**], [**?**] for the sparse recovery step.

For the modified-PCP program (**??**), the Augmented Lagrangian function is:

$$\begin{aligned} \mathbb{L}(\tilde{\mathbf{L}}_{\text{new}}, \tilde{\mathbf{S}}, \mathbf{Y}, \tau) = \|\tilde{\mathbf{L}}_{\text{new}}\|_* + \lambda\|\tilde{\mathbf{S}}\|_1 + \langle \mathbf{Y}, \mathbf{M} - \tilde{\mathbf{L}}_{\text{new}} - \tilde{\mathbf{S}} \\ - \mathbf{G}\tilde{\mathbf{X}}'\rangle + \frac{\tau}{2}\|\mathbf{M} - \tilde{\mathbf{L}}_{\text{new}} - \tilde{\mathbf{S}} - \mathbf{G}\tilde{\mathbf{X}}'\|_F^2, \end{aligned}$$

Thus, with similar steps in [**?**], we have following algorithm. In Algorithm **??**, Lines 3 solves $\tilde{\mathbf{S}}_{k+1} = \arg\min_{\tilde{\mathbf{S}}} \|\tilde{\mathbf{L}}_{\text{new},k}\|_* +$

Algorithm 5 Algorithm for solving Modified-PCP (??)

```
Require: Measurement matrix $\mathbf{M} \in \mathbb{R}^{n_1 \times n_2}$, $\lambda = 1/\sqrt{\max\{n_1, n_2\}}$, $\mathbf{G}$.
1: $\mathbf{Y}_0 = \mathbf{M}/\max\{\|\mathbf{M}\|, \|\mathbf{M}\|_\infty/\lambda\}$; $\tilde{\mathbf{S}}_0 = 0$; $\tau_0 > 0$; $v > 1$; $k = 0$.
2: while not converged do
3:     $\tilde{\mathbf{S}}_{k+1} = \mathfrak{S}_{\lambda\tau_k^{-1}}[\mathbf{M} - \mathbf{G}\tilde{\mathbf{X}}_k - \tilde{\mathbf{L}}_{\text{new},k} + \tau_k^{-1}\mathbf{Y}_k]$.
4:     $(\tilde{\mathbf{U}}, \tilde{\mathbf{\Sigma}}, \tilde{\mathbf{V}}) = \text{svd}((I - \mathbf{G}\mathbf{G}')(\mathbf{M} - \tilde{\mathbf{S}}_{k+1} + \tau_k^{-1}\mathbf{Y}_k))$;
5:     $\tilde{\mathbf{L}}_{\text{new},k+1} = \tilde{\mathbf{U}}\mathfrak{S}_{\tau_k^{-1}}[\tilde{\mathbf{\Sigma}}]\tilde{\mathbf{V}}^T$.
6:     $\tilde{\mathbf{X}}_{k+1} = \mathbf{G}'(\mathbf{M} - \tilde{\mathbf{S}}_{k+1} + \tau_k^{-1}\mathbf{Y}_k)$
7:     $\mathbf{Y}_{k+1} = \mathbf{Y}_k + \tau_k(\mathbf{M} - \tilde{\mathbf{S}}_{k+1} - \tilde{\mathbf{L}}_{\text{new},k+1} - \mathbf{G}\tilde{\mathbf{X}}_{k+1})$.
8:     $\tau_{k+1} = \min(v\tau_k, \bar{\tau})$.
9:     $k \leftarrow k + 1$.
10: end while
Ensure: $\hat{\mathbf{L}}_{\text{new}} = \tilde{\mathbf{L}}_{\text{new},k}, \hat{\mathbf{S}} = \tilde{\mathbf{S}}_k, \hat{L} = \mathbf{M} - \tilde{\mathbf{S}}_k$.
```

$\lambda\|\tilde{\mathbf{S}}\|_1 + \langle \mathbf{Y}_k, \mathbf{M} - \tilde{\mathbf{L}}_{\text{new},k} - \tilde{\mathbf{S}} - \mathbf{G}\tilde{\mathbf{X}}_k'\rangle + \frac{\tau}{2}\|\mathbf{M} - \tilde{\mathbf{L}}_{\text{new},k} - \tilde{\mathbf{S}} - \mathbf{G}\tilde{\mathbf{X}}_k'\|_F^2$; Line 4-6 solve $[\tilde{\mathbf{L}}_{\text{new},k+1}, \tilde{\mathbf{X}}_{k+1}] = \arg\min_{\tilde{\mathbf{L}}_{\text{new}},\tilde{\mathbf{X}}} \|\tilde{\mathbf{L}}_{\text{new}}\|_* + \lambda\|\tilde{\mathbf{S}}_{k+1}\|_1 + \langle \mathbf{Y}_k, \mathbf{M} - \tilde{\mathbf{L}}_{\text{new}} - \tilde{\mathbf{S}}_{k+1} - \mathbf{G}\tilde{\mathbf{X}}'\rangle + \frac{\tau}{2}\|\mathbf{M} - \tilde{\mathbf{L}}_{\text{new}} - \tilde{\mathbf{S}}_{k+1} - \mathbf{G}\tilde{\mathbf{X}}_k'\|_F^2$. The soft-thresholding operator is defined as

$$\mathfrak{S}_\epsilon[x] = \begin{cases} x - \epsilon, & \text{if } x > \epsilon; \\ x + \epsilon, & \text{if } x < -\epsilon; \\ 0, & \text{otherwise,} \end{cases} \tag{22}$$

Parameters are set as suggested in [**?**], i.e., $\tau_0 = 1.25/\|\mathbf{M}\|, v = 1.5, \bar{\tau} = 10^7\tau_0$ and iteration is stopped when $\|\mathbf{M} - \tilde{\mathbf{S}}_{k+1} - \tilde{\mathbf{L}}_{\text{new},k+1} - \mathbf{G}\tilde{\mathbf{X}}_{k+1}\|_F/\|\mathbf{M}\|_F < 10^{-7}$.

V. Performance Guarantees for ReProCS and Modified-PCP

A. Results for ReProCS

In recently published work [?], we obtained the following partial result for ReProCS. We call it a partial result because its last assumption depends on intermediate algorithm estimates. In ongoing work that will be presented at ICASSP 2015 [?] and at ISIT 2015 [?], we now have a complete correctness result. We state this later in Theorem ??.

Theorem 5.1. *Consider Algorithm ??. Let $c := c_{\max}$ and $r := r_0 + (J-1)c$. Pick a ζ that satisfies*

$$\zeta \leq \min\left(\frac{10^{-4}}{r^2}, \frac{1.5 \times 10^{-4}}{r^2 f}, \frac{1}{r^3 \gamma_*^2}\right)$$

Assume that the initial subspace estimate is accurate enough, i.e. $\|(I - \hat{P}_0 \hat{P}_0')P_0\| \leq r_0 \zeta$. If the following conditions hold:

1) *The algorithm parameters are set as $\xi = \xi_0(\zeta)$, $7\xi \leq \omega \leq S_{\min} - 7\xi$, $K = K(\zeta)$, $\alpha \geq \alpha_{add}(\zeta)$ where these quantities are defined in Definition 4.1 of [?].*
2) *L_t satisfies Model ?? or Model ?? with*
   a) *$0 \leq c_{j,\text{new}} \leq c_{\max}$ for all j (thus $r_j \leq r_{\max} := r_0 + J c_{\max}$),*
   b) *the a_t's mutually independent over t,*
   c) *$\|a_t\|_\infty \leq \gamma_*$ for all t (a_t's bounded);,*
3) *Slow subspace change holds: (??) holds with $d = K\alpha$; (??) holds with $g^+ = 1.41$ and (??) holds with $v = 1.2$; and c and γ_{new} are small enough so that $14\xi_0(\zeta) \leq S_{\min}$.*
4) *Denseness holds: equation (??) holds with $\kappa_{2s,*}^+ = 0.3$ and $\kappa_{2s,\text{new}}^+ = 0.15$*
5) *The matrices $D_{j,\text{new},k} := (I - \hat{P}_{j-1}\hat{P}_{j-1}' - \hat{P}_{j,\text{new},k}\hat{P}_{j,\text{new},k}')P_{j,\text{new}}$ and $Q_{j,\text{new},k} := (I - P_{j,\text{new}}P_{j,\text{new}}{}')\hat{P}_{j,\text{new},k}$ satisfy*

$$\max_j \max_{1\leq k\leq K} \kappa_s(D_{j,\text{new},k}) \leq \kappa_s^+ := 0.152$$
$$\max_j \max_{1\leq k\leq K} \kappa_{2s}(Q_{j,\text{new},k}) \leq \tilde{\kappa}_{2s}^+ := 0.15$$

then, with probability at least $(1 - n^{-10})$, at all times, t, all of the following hold:

1) *at all times, t,*

$$\hat{\mathcal{T}}_t = \mathcal{T}_t \quad \text{and}$$

$$\|e_t\|_2 = \|L_t - \hat{L}_t\|_2 = \|\hat{S}_t - S_t\|_2 \leq 0.18\sqrt{c}\gamma_{\text{new}} + 1.2\sqrt{\zeta}(\sqrt{r} + 0.06\sqrt{c}).$$

2) *the subspace error $SE_{(t)} := \|(I - \hat{P}_{(t)}\hat{P}_{(t)}')P_{(t)}\|_2$ satisfies*

$$SE_{(t)} \leq \begin{cases} (r_0 + (j-1)c)\zeta + 0.4c\zeta + 0.6^{k-1} \\ \qquad\qquad \text{if } t \in \mathcal{I}_{j,k},\ k = 1,2\dots K \\ (r_0 + jc)\zeta \qquad \text{if } t \in \mathcal{I}_{j,K+1} \end{cases}$$
$$\leq \begin{cases} 10^{-2}\sqrt{\zeta} + 0.6^{k-1} \\ \qquad\qquad \text{if } t \in \mathcal{I}_{j,k},\ k = 1,2\dots K \\ 10^{-2}\sqrt{\zeta} \qquad \text{if } t \in \mathcal{I}_{j,K+1} \end{cases}$$

3) *the error $e_t = \hat{S}_t - S_t = L_t - \hat{L}_t$ satisfies the following at various times*

$$\|e_t\|_2 \leq \begin{cases} 0.18\sqrt{c}0.72^{k-1}\gamma_{\text{new}} + \\ \qquad 1.2(\sqrt{r} + 0.06\sqrt{c})(r_0 + (j-1)c)\zeta\gamma_* \\ \qquad\qquad \text{if } t \in \mathcal{I}_{j,k},\ k = 1,2\dots K \\ 1.2(r_0 + jc)\zeta\sqrt{r}\gamma_* \quad \text{if } t \in \mathcal{I}_{j,K+1} \end{cases}$$
$$\leq \begin{cases} 0.18\sqrt{c}0.72^{k-1}\gamma_{\text{new}} + 1.2(\sqrt{r} + 0.06\sqrt{c})\sqrt{\zeta} \\ \qquad\qquad \text{if } t \in \mathcal{I}_{j,k},\ k = 1,2\dots K \\ 1.2\sqrt{r}\sqrt{\zeta} \quad \text{if } t \in \mathcal{I}_{j,K+1} \end{cases}$$

The above result says the following. Consider Algorithm ??. Assume that the initial subspace error is small enough. If the algorithm parameters are appropriately set, if slow subspace change holds, if the subspaces are dense, if the condition number of $\text{Cov}[a_{t,\text{new}}]$ is small enough, and if the currently unestimated part of the newly added subspace is dense enough (this is an assumption on the algorithm estimates), then, w.h.p., we will get exact support recovery at all times. Moreover, the sparse recovery error will always be bounded by $0.18\sqrt{c}\gamma_{\text{new}}$ plus a constant times $\sqrt{\zeta}$. Since ζ is very small, $\gamma_{\text{new}} < S_{\min}$, and c is also small, the normalized reconstruction error for recovering S_t will be small at all times. In the second conclusion, we bound the subspace estimation error, $\text{SE}_{(t)}$. When a subspace change occurs, this error is initially bounded by one. The above result shows that, w.h.p., with each projection PCA step, this error decays exponentially and falls below $0.01\sqrt{\zeta}$ within K projection PCA steps. The third conclusion shows that, with each projection PCA step, w.h.p., the sparse recovery error as well as the error in recovering L_t also decay in a similar fashion.

The most important limitation of the above result is that it requires an assumption on $D_{\text{new},k}$ and $Q_{\text{new},k}$ which depend on intermediate algorithm estimates, $\hat{P}_{j,\text{new},k}$. Moreover, it studies an algorithm that requires knowledge of model parameters. The first limitation was removed in our recent work [**?**] and both limitations were removed in our very recent work [**?**]. We briefly summarize the result of [**?**] using the notation and model of this bookchapter (this is the notation and model of our earlier published work [**?**] and has minor differences from the notation and model used in [**?**]). The key point is to replace condition 5 (which is an assumption that depends on intermediate algorithm estimates, $\hat{P}_{j,\text{new},k}$) by an assumption on support change of S_t. The new assumption just says that the support should change at least every β frames and when it does, the change should be by at least s/ρ (with $\rho^2\beta$ being small enough) and it should be such that the sets of changed indices are mutually disjoint for a period of α frames.

Theorem 5.2 (Complete Correctness Result for ReProCS [**?**])**.** *Consider Algorithm **??** but with sparse recovery done only simple ℓ_1 minimization and support recovery done only by thresholding; and with ξ, ω, α and K set as given below.*

1) *Replace condition 1) of Theorem **??** by the following: set algorithm parameters as $\xi = \sqrt{r_{\text{new}}}\gamma_{\text{new}} + (\sqrt{r} + \sqrt{r_{\text{new}}})\sqrt{\zeta}$; $\omega = 7\xi$; $K = \left\lceil \frac{\log(0.16 r_{\text{new}}\zeta)}{\log(0.83)} \right\rceil$; and $\alpha = C(\log(6(K+1)J) + 11\log(n))$ for a constant $C \geq C_{add}$ with $C_{add} := 32 \cdot 100^2 \frac{\max\{16, 1.2(\sqrt{\zeta} + \sqrt{r_{\text{new}}}\gamma_{\text{new}})^4\}}{(r_{\text{new}}\zeta\lambda^-)^2}$*
2) *Assume conditions 2), 3), 4) of Theorem **??** hold.*
3) *Replace condition 5) of Theorem **??** by the following assumption on support change: 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.*
   a) *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 $|\mathcal{T}^{[k]}| \leq s$.*
   b) *Let ρ be a positive integer so that for any k,*
   $$\mathcal{T}^{[k]} \cap \mathcal{T}^{[k+\rho]} = \emptyset;$$
   assume that
   $$\rho^2\beta \leq 0.01\alpha.$$
   c) *For any k,*
   $$\sum_{i=k+1}^{k+\alpha} \left|\mathcal{T}^{[i]} \setminus \mathcal{T}^{[i+1]}\right| \leq n$$
   and for any $k < i \leq k+\alpha$,
   $$(\mathcal{T}^{[k]} \setminus \mathcal{T}^{[k+1]}) \cap (\mathcal{T}^{[i]} \setminus \mathcal{T}^{[i+1]}) = \emptyset.$$
   (One way to ensure $\sum_{i=k+1}^{k+\alpha} |\mathcal{T}^{[i]} \setminus \mathcal{T}^{[i+1]}| \leq n$ is to require that for all i, $|\mathcal{T}^{[i]} \setminus \mathcal{T}^{[i+1]}| \leq \frac{s}{\rho_2}$ with $\frac{s}{\rho_2}\alpha \leq n$.)

*Then all conclusions of Theorem **??** hold.*

One example application that satisfies the above model on support change is a video application consisting of a foreground with one object of length s or less that can remain 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, i.e. $\frac{s}{\rho_2}\alpha \leq n$. Anytime after it has disappeared another object could appear.

B. Modified-PCP correctness result: static and online robust PCA cases

1) Modified-PCP correctness result for the static case: Consider the modified-PCP program given in (**??**). As explained in [**?**], we need that $\mathbf{S}$ is not low rank in order to separate it from $\mathbf{L}_{\text{new}}$ in a batch fashion (recall that modified-PCP is a batch program; modified-PCP for online robust PCA is a piecewise batch solution). One way to ensure that $\mathbf{S}$ is full rank w.h.p. is by selecting the support of $\mathbf{S}$ uniformly at random [**?**]. We assume this here too. In addition, we need a denseness assumption on the columns of $\mathbf{G}$ and on the left and right singular vectors of $\mathbf{L}_{\text{new}}$.

Let $n_{(1)} = \max(n_1, n_2)$ and $n_{(2)} = \min(n_1, n_2)$. Assume that following hold with a constant ρ_r

$$\max_i \|[\mathbf{G}\ \mathbf{U}_{\text{new}}]'\mathbf{e}_i\|^2 \leq \frac{\rho_r n_{(2)}}{n_1 \log^2 n_{(1)}}, \tag{23}$$

$$\max_i \|\mathbf{V}_{\text{new}}'\mathbf{e}_i\|^2 \leq \frac{\rho_r n_{(2)}}{n_2 \log^2 n_{(1)}}, \tag{24}$$

and

$$\|\mathbf{U}_{\text{new}}\mathbf{V}_{\text{new}}'\|_\infty \leq \sqrt{\frac{\rho_r}{n_{(1)} \log^2 n_{(1)}}}. \tag{25}$$

Theorem 5.3. *Consider the problem of recovering $\mathbf{L}$ and $\mathbf{S}$ from $\mathbf{M}$ using partial subspace knowledge $\mathbf{G}$ by solving modified-PCP (**??**). Assume that Ω, the support set of $\mathbf{S}$, is uniformly distributed with size m satisfying*

$$m \leq 0.4\rho_s n_1 n_2 \tag{26}$$

*Assume that $\mathbf{L}$ satisfies (**??**), (**??**) and (**??**) and ρ_s, ρ_r, are small enough and n_1, n_2 are large enough. The explicit bounds are available in Assumption 3.2 of [**?**]. Then, Modified-PCP (**??**) with $\lambda = 1/\sqrt{n_{(1)}}$ recovers $\mathbf{S}$ and $\mathbf{L}$ exactly with probability at least $1 - 23n_{(1)}^{-10}$.*

2) Modified-PCP correctness result for online robust PCA: Consider the online / recursive robust PCA problem where data vectors $M_t := S_t + L_t$ come in sequentially and their subspace can change over time. Starting with an initial knowledge of the subspace, the goal is to estimate the subspace spanned by $L_1, L_2, \dots L_t$ and to recover the S_t's. For the above model, the following is an easy corollary.

Corollary 5.4 (modified-PCP for online robust PCA)**.** *Assume Model **??** holds with $\sum_j (c_{j,\text{new}} - c_{j,\text{old}}) \leq c_{dif}$. Let $\mathbf{M}^j := [M_{t_j}, M_{t_j+1}, \dots M_{t_{j+1}-1}]$, $\mathbf{L}^j := [L_{t_j}, L_{t_j+1}, \dots L_{t_{j+1}-1}]$, $\mathbf{S}^j := [S_{t_j}, S_{t_j+1}, \dots S_{t_{j+1}-1}]$ and let $\mathbf{L}^{full} := [\mathbf{L}^1, \mathbf{L}^2, \dots \mathbf{L}^J]$ and $\mathbf{S}^{full} := [\mathbf{S}^1, \mathbf{S}^2, \dots \mathbf{S}^J]$. Suppose that the following hold.*

1) *$\mathbf{S}^{full}$ satisfies the assumptions of Theorem **??**.*
2) *The initial subspace $\text{span}(\mathbf{P}_0)$ is exactly known, i.e. we are given $\hat{\mathbf{P}}_0$ with $\text{span}(\hat{\mathbf{P}}_0) = \text{span}(\mathbf{P}_0)$.*
3) *For all $j = 1, 2, \dots J$, (**??**), (**??**), and (**??**) hold with $n_1 = n$, $n_2 = t_{j+1} - t_j$, $\mathbf{G} = \mathbf{P}_{j-1}$, $\mathbf{U}_{\text{new}} = \mathbf{P}_{j,\text{new}}$ and $\mathbf{V}_{\text{new}}$ being the matrix of right singular vectors of $\mathbf{L}_{\text{new}} = (\mathbf{I} - \mathbf{P}_{j-1}\mathbf{P}_{j-1}')\mathbf{L}^j$.*
4) *We solve modified-PCP at every $t = t_{j+1}$, using $\mathbf{M} = \mathbf{M}^j$ and with $\mathbf{G} = \mathbf{G}_j = \hat{\mathbf{P}}_{j-1}$ where $\hat{\mathbf{P}}_{j-1}$ is the matrix of left singular vectors of the reduced SVD of $\hat{\mathbf{L}}_{j-1}$ (the low-rank matrix obtained from modified-PCP on $\mathbf{M}^{j-1}$). At $t = t_1$ we use $\mathbf{G} = \hat{\mathbf{P}}_0$.*

Then, modified-PCP recovers $\mathbf{S}^{full}, \mathbf{L}^{full}$ exactly and in a piecewise batch fashion with probability at least $(1 - 23n^{-10})^J$.

Discussion w.r.t. PCP. Two possible corollaries for PCP can be stated depending on whether we want to compare PCP and mod-PCP when both have the same memory and time complexity or when both are provided the same total data.

The following is the corollary for PCP applied to the entire data matrix $\mathbf{M}^{\text{full}}$ in one go. With doing this, PCP gets all the same data that modified-PCP has access to. But PCP needs to store the entire data matrix in memory and also needs to operate on it, i.e. its memory complexity is roughly J times larger than that for modified-PCP and its time complexity is $poly(J)$ times larger than that of modified-PCP.

Corollary 5.5 (PCP for online robust PCA)**.** *Assume Model **??** holds with $\sum_j (c_{j,\text{new}} - c_{j,\text{old}}) \leq c_{dif}$. If $\mathbf{S}^{full}$ satisfies the assumptions of Theorem **??** and if (**??**), (**??**), and (**??**) hold with $n_1 = n$, $n_2 = t_{J+1} - t_1$, $\mathbf{G}_{PCP} = [\ ]$, $\mathbf{U}_{\text{new},PCP} = \mathbf{U} = [\mathbf{P}_0, \mathbf{P}_{1,\text{new}}, \dots \mathbf{P}_{J,\text{new}}]$ and $\mathbf{V}_{\text{new},PCP} = \mathbf{V}$ being the right singular vectors of $\mathbf{L}^{full} := [\mathbf{L}^1, \mathbf{L}^2, \dots \mathbf{L}^J]$, then, we can recover $\mathbf{L}^{full}$ and $\mathbf{S}^{full}$ exactly with probability at least $(1 - 23n^{-10})$ by solving PCP (**??**) with input $\mathbf{M}^{full}$. Here $\mathbf{M}^{full} := \mathbf{L}^{full} + \mathbf{S}^{full}$.*

When we compare this with the result for modified-PCP, the second and third condition are clearly significantly weaker than those for PCP. The first conditions cannot be easily compared. The LHS contains at most $r_{\max} + c = r_0 + c_{dif} + c$ columns for modified-PCP, while it contains $r_0 + Jc$ columns for PCP. However, the RHS for PCP is also larger. If $t_{j+1} - t_j = d$, then the RHS is also J times larger for PCP than for modified-PCP. Thus with the above PCP corollary, the first condition of the above and of modified-PCP cannot be compared.

The above advantages for mod-PCP come with two caveats. First, modified-PCP assumes knowledge of the subspace change times while PCP does not need this. Secondly, modified-PCP succeeds w.p. $(1 - 23n^{-10})^J \geq 1 - 23Jn^{-10}$ while PCP succeeds w.p. $1 - 23n^{-10}$.

Alternatively if PCP is solved at every $t = t_{j+1}$ using $\mathbf{M}^j$, we get the following corollary. This keeps the memory and time complexity of PCP the same as that for modified-PCP, but PCP gets access to less data than modified-PCP.

Corollary 5.6 (PCP for $\mathbf{M}^j$)**.** *Assume Model* **??** *holds with* $\sum_j (c_{j,\text{new}} - c_{j,\text{old}}) \leq c_{dif}$. *Solve PCP, i.e. (*__??__*), at* $t = t_{j+1}$ *using* $\mathbf{M}^j$. *If* $\mathbf{S}^{full}$ *satisfies the assumptions of Theorem* **??** *and if (*__??__*), (*__??__*), and (*__??__*) hold with* $n_1 = n$, $n_2 = t_{j+1} - t_j$, $\mathbf{G}_{PCP} = [\ ]$, $\mathbf{U}_{\text{new},PCP} = \mathbf{P}_j$ *and* $\mathbf{V}_{\text{new},PCP} = \mathbf{V}_j$ *being the right singular vectors of* $\mathbf{L}^j$ *for all* $j = 1, 2, \ldots, J$, *then, we can recover* $\mathbf{L}^{full}$ *and* $\mathbf{S}^{full}$ *exactly with probability at least* $(1 - 23n^{-10})^J$.

When we compare this with modified-PCP, the second and third condition are significantly weaker than those for PCP when $c_{j,\text{new}} \ll r_j$. The first condition is exactly the same when $c_{j,\text{old}} = 0$ and is only slightly stronger as long as $c_{j,\text{old}} \ll r_j$.

C. Discussion

In this discussion we use the correctness result of Theorem **??** for ReProCS. This was proved in our very recent work [**?**]. We should point out first that the matrix completion problem can be interpreted as a special case of the robust PCA problem where the support sets $\mathcal{T}_t$ are the set of missing entries and hence are known. Thus an easy corollary of our result is a result for online matrix completion. This can be compared with the corresponding result for nuclear norm minimization (NNM) [**?**] from [**?**].

Our result requires accurate initial subspace knowledge. As explained earlier, for video analytics, this corresponds to requiring an initial short sequence of background-only video frames whose subspace can be estimated via SVD (followed by using a singular value threshold to retain a certain number of top left singular vectors). Alternatively if an initial short sequence of the video data satisfies the assumptions required by a batch method such as PCP, that can be used to estimate the low-rank part, followed by SVD to get the column subspace.

In Model **??** or **??** and the slow subspace change assumption, we are placing a slow increase assumption on the eigenvalues along the new directions, $\boldsymbol{P}_{t_j,\text{new}}$, only for the interval $[t_j, t_{j+1})$. Thus after t_{j+1}, the eigenvalues along $\boldsymbol{P}_{t_j,\text{new}}$ can increase gradually or suddenly to any large value up to λ^+.

The assumption on $\mathcal{T}_t$ is a practical model for moving foreground objects in video. We should point out that this model is one special case of the general set of conditions that we need (see Model 5.1 of [**?**]).

As explained in [**?**], the model on $\mathcal{T}_t$ and the denseness condition of the theorem constrain s and s, r_0, r_new, J respectively. The model on $\mathcal{T}_t$ requires $s \leq \rho_2 n/\alpha$ for a constant ρ_2. Using the expression for α, it is easy to see that as long as $J \in \mathcal{O}(n)$, we have $\alpha \in \mathcal{O}(\log n)$ and so this needs $s \in \mathcal{O}(\frac{n}{\log n})$. With $s \in \mathcal{O}(\frac{n}{\log n})$, using (**??**), it is easy to see that the denseness condition will hold if $r_0 \in \mathcal{O}(\log n)$, $J \in \mathcal{O}(\log n)$ and r_new is a constant. This is one set of sufficient conditions that we allow on the rank-sparsity product.

Let $\boldsymbol{L}^\text{full} := [L_1, L_2, \ldots, L_{t_\text{max}}]$ and $\boldsymbol{S}^\text{full} := [S_1, S_2, \ldots, S_{t_\text{max}}]$. Let $r_\text{mat} := \text{rank}(\boldsymbol{L}^\text{full})$. Clearly $r_\text{mat} \leq r_0 + J r_\text{new}$ and the bound is tight. Let $s_\text{mat} := t_\text{max} s$ be a bound on the support size of the outliers' matrix $\boldsymbol{S}^\text{full}$. In terms of r_mat and s_mat, what we need is $r_\text{mat} \in \mathcal{O}(\log n)$ and $s_\text{mat} \in \mathcal{O}(\frac{n t_\text{max}}{\log n})$. This is stronger than what the PCP result from [**?**] needs ([**?**] allows $r_\text{mat} \in \mathcal{O}\left(\frac{n}{(\log n)^2}\right)$ while allowing $s_\text{mat} \in \mathcal{O}(n t_\text{max})$), but is similar to what the PCP results from [**?**], [**?**] need.

Other disadvantages of our result are as follows. (1) Our result needs accurate initial subspace knowledge and slow subspace change of L_t. As explained earlier and in [**?**, Fig. 6], both of these are often practically valid for video analytics applications. Moreover, we also need the L_t's to be zero mean and mutually independent over time. Zero mean is achieved by letting L_t be the background image at time t with an empirical 'mean background image', computed using the training data, subtracted out. The independence assumption then models independent background variations around a common mean. As we explain in Section **??**, this can be easily relaxed and we can get a result very similar to the current one under a first order autoregressive model on the L_t's. (2) Moreover, ReProCS needs four algorithm parameters to be appropriately set. The PCP or NNM results need this for none [**?**], [**?**] or at most one [**?**], [**?**] algorithm parameter. (3) Thirdly, our result for online RPCA also needs a lower bound on S_min while the PCP results do not need this. (4) Moreover, even with this, we can only guarantee accurate recovery of L_t, while PCP or NNM guarantee exact recovery.

(1) The advantage of our work is that we analyze an online algorithm (ReProCS) that is faster and needs less storage compared with PCP or NNM. It needs to store only a few $n \times \alpha$ or $n \times r_\text{mat}$ matrices, thus the storage complexity is $\mathcal{O}(n \log n)$ while that for PCP or NNM is $\mathcal{O}(n t_\text{max})$. In general t_max can be much larger than $\log n$. (2) Moreover, we do not need any assumption on the right singular vectors of $\boldsymbol{L}$ while all results for PCP or NNM do. (3) Most importantly, our results allow highly correlated changes of the set of missing entries (or outliers). From the assumption on $\mathcal{T}_t$, it is easy to see that we allow the number of missing entries (or outliers) per row of $\boldsymbol{L}$ to be $\mathcal{O}(t_\text{max})$ as long as the sets follow the support change

assumption [4]. The PCP results from [?], [?] need this number to be $\mathcal{O}(\frac{t_{\max}}{r_{\text{mat}}})$ which is stronger. The PCP result from [?] or the NNM result [?] need an even stronger condition - they need the set $(\cup_{t=1}^{t_{\max}} \mathcal{T}_t)$ to be generated uniformly at random.

In [?], Feng et. al. propose a method for online RPCA and prove a partial result for their algorithm. The approach is to reformulate the PCP program and use this reformulation to develop a recursive algorithm that converges asymptotically to the solution of PCP as long as the basis estimate $\hat{\boldsymbol{P}}_t$ is full rank at each time t. Since this result assumes something about the algorithm estimates, it is also only a *partial* result. Another somewhat related work is that of Feng et. al. [?] on online PCA with contaminated data. This does not model the outlier as a sparse vector but defines anything that is far from the data subspace as an outlier.

To compare with the correctness result of modified-PCP given earlier, two things can be said. First, like PCP, the result for modified PCP also needs uniformly randomly generated support sets. But its advantage is that its assumption on the rank-sparsity product is weaker than that of PCP, and hence weaker than that needed by ReProCS. Also see the simulations' section.

VI. Numerical experiments

A. Simulation experiments

1) Comparing ReProCS and PCP: We first provide some simulations that demonstrate the result we have proven above for ReProCS and how it compares with the result for PCP.

The data for Figure **??** was generated as follows. We chose $n = 256$ and $t_{\max} = 15,000$. Each measurement had $s = 20$ missing or corrupted entries, i.e. $|\mathcal{T}_t| = 20$. Each non-zero entry of the sparse vector was drawn uniformly at random between 2 and 6 independent of other entries and other times t. In Fig. **??** the support of S_t changes as assumed Theorem **??** with $\rho = 2$ and $\beta = 18$. So $\mathcal{T}_t$ changes by $\frac{s}{2} = 10$ indices every 18 time instants. When it reaches the bottom of the vector, it starts over again at the top. This pattern can be seen in the bottom half of the figure which shows the sparsity pattern of the matrix $\boldsymbol{S}$

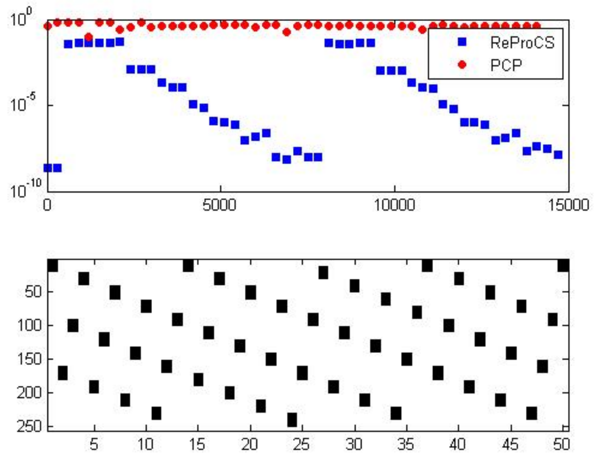
To form the low dimensional vectors L_t, we started with an $n \times r$ matrix of i.i.d. Gaussian entries and orthonormalized the columns using Gram-Schmidt. The first $r_0 = 10$ columns of this matrix formed $\boldsymbol{P}_{(0)}$, the next 2 columns formed $\boldsymbol{P}_{(1),\text{new}}$, and the last 2 columns formed $\boldsymbol{P}_{(2),\text{new}}$ We show two subspace changes which occur at $t_1 = 600$ and $t_2 = 8000$. The entries of $\boldsymbol{a}_{t,*}$ were drawn uniformly at random between -5 and 5, and the entries of $\boldsymbol{a}_{t,\text{new}}$ were drawn uniformly at random between $-\sqrt{3v_i^{t-t_j}\lambda^-}$ and $\sqrt{3v_i^{t-t_j}\lambda^-}$ with $v_i = 1.00017$ and $\lambda^- = 1$. Entries of $\boldsymbol{a}_t$ were independent of each other and of the other $\boldsymbol{a}_t$'s.

For this simulated data we compare the performance of ReProCS and PCP. The plots show the relative error in recovering L_t, that is $\|L_t - \hat{L}_t\|_2/\|L_t\|_2$. For the initial subspace estimate $\hat{\boldsymbol{P}}_0$, we used $\boldsymbol{P}_0$ plus some small Gaussian noise and then obtained orthonormal columns. We set $\alpha = 800$ and $K = 6$. For the PCP algorithm, we perform the optimization every α time instants using all of the data up to that point. So the first time PCP is performed on $[M_1, \dots, M_\alpha]$ and the second time it is performed on $[M_1, \dots, M_{2\alpha}]$ and so on.

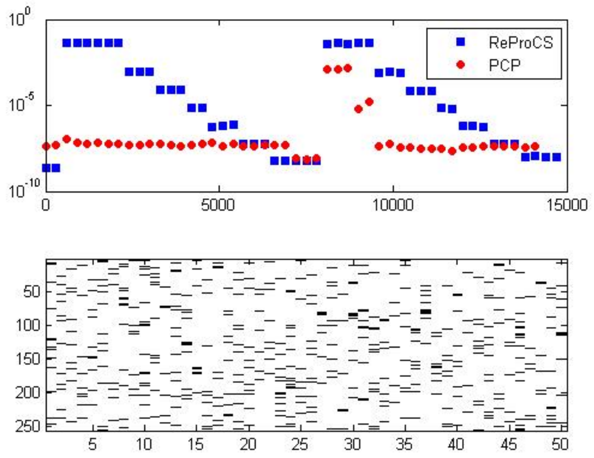
Figure **??** illustrates the result we have proven in Theorem **??**. That is ReProCS takes advantage of the initial subspace estimate and slow subspace change (including the bound on γ_{new}) to handle the case when the supports of S_t are correlated in time. Notice how the ReProCS error increases after a subspace change, but decays exponentially with each projection PCA step. For this data, the PCP program fails to give a meaningful estimate for all but a few times. The average time taken by the ReProCS algorithm was 52 seconds, while PCP averaged over 5 minutes. Simulations were coded in MATLAB® and run on a desktop computer with a 3.2 GHz processor. Compare this to Figure **??** where the only change in the data is that the support of S is chosen uniformly at random from all sets of size $\frac{st_{\max}}{n}$ (as assumed in [?]). Thus the total sparsity of the matrix S is the same for both figures. In Figure **??**, ReProCS performs almost the same as in Fig. **??**, while PCP does substantially better than in the case of correlated supports.

2) Comparing Modified-PCP, ReProCS, PCP and other algorithms: We generated data using Model **??** with $n = 256$, $J = 3$, $r_0 = 40$, $t_0 = 200$ and $c_{j,\text{new}} = 4$, $c_{j,\text{old}} = 4$, for each $j = 1, 2, 3$. We used $t_1 = t_0 + 6\alpha + 1$, $t_2 = t_0 + 12\alpha + 1$ and $t_3 = t_0 + 18\alpha + 1$ with $\alpha = 100$, $t_{\max} = 2600$ and $\gamma = 5$. The coefficients, $\mathbf{a}_{t,*} = \mathbf{P}_{j-1}' L_t$ were i.i.d. uniformly distributed in the interval $[-\gamma, \gamma]$; the coefficients along the new directions, $\mathbf{a}_{t,\text{new}} := \mathbf{P}_{j,\text{new}}' L_t$ generated i.i.d. uniformly distributed in the interval $[-\gamma_{\text{new}}, \gamma_{\text{new}}]$ (with a $\gamma_{\text{new}} \leq \gamma$) for the first 1700 columns after the subspace change and i.i.d. uniformly distributed in the interval $[-\gamma, \gamma]$ after that. We vary the value of γ_{new}; small values mean that "slow subspace change" required by ReProCS

[4]In a period of length α, the set $\mathcal{T}_t$ can occupy index i for at most $\rho\beta$ time instants, and this pattern is allowed to repeat every α time instants. So an index can be in the support for a total of $\rho\beta\frac{t_{\max}}{\alpha}$ time instants and the model assumes $\rho\beta \leq \frac{0.01\alpha}{\rho}$ for a constant ρ.



(a) correlated support change



(b) uniformly randomly selected support sets

Fig. 3. Comparison of ReProCS and PCP for the RPCA problem. In each subfigure, the top plot is the relative error $\|L_t - \hat{L}_t\|_2/\|L_t\|_2$. The bottom plot shows the sparsity pattern of $\boldsymbol{S}$ (black represents a non-zero entry). Results are averaged over 100 simulations and plotted every 300 time instants.

holds. The sparse matrix $\mathbf{S}$ was generated in two different ways to simulate uncorrelated and correlated support change. For partial knowledge, $\mathbf{G}$, we first did SVD decomposition on $[L_1, L_2, \cdots, L_{t_0}]$ and kept the directions corresponding to singular values larger than $\mathbf{E}(z^2)/9$, where $z \sim \text{Unif}[-\gamma_{\text{new}}, \gamma_{\text{new}}]$. We solved PCP and modified-PCP every 200 frames by using the observations for the last 200 frames as the matrix $\mathbf{M}$. The ReProCS algorithm was implemented with $\alpha = 100$. The averaged sparse part errors with two different sets of parameters over 20 Monte Carlo simulations are displayed in Fig. **??** and Fig. **??**.

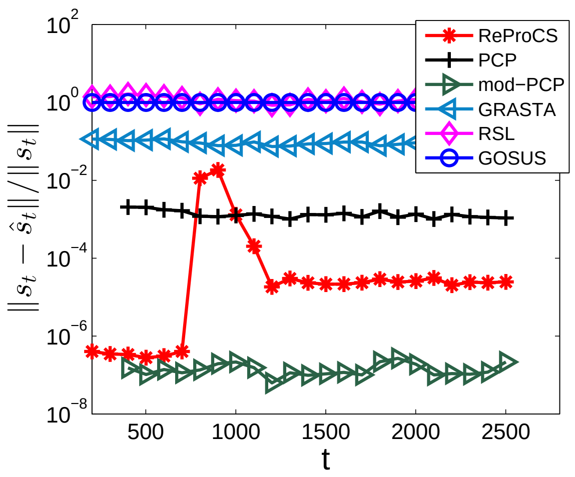
In the first case, Fig. **??**, we used $\gamma_{\text{new}} = \gamma$ and so "slow subspace change" does not hold. For the sparse vectors S_t, each index is chosen to be in support with probability 0.0781. The nonzero entries are uniformly distributed between $[20, 60]$. Since "slow subspace change" does not hold, ReProCS does not work well. Since the support is generated independently over time, this is a good case for both PCP and mod-PCP. Mod-PCP has the smallest sparse recovery error. In the second case, Fig. **??**, we used $\gamma_{\text{new}} = 1$ and thus "slow subspace change" holds. For sparse vectors, S_t, the support is generated in a correlated fashion. We used support size $s = 10$ for each S_t; the support remained constant for 25 columns and then moved down by $s/2 = 5$ indices. Once it reached n, it rolled back over to index one. Because of the correlated support change, PCP does not work. In this case, the sparse vectors are highly correlated over time, resulting in sparse matrix $\mathbf{S}$ that is even more low rank, thus neither mod-PCP nor PCP work for this data. In this case, only ReProCS works.

Thus from simulations, modified-PCP is able to handle correlated support change better than PCP but worse than ReProCS. Modified-PCP also works when slow subspace change does not hold; this is a situation where ReProCS fails. Of course, modified-PCP, GRASTA and ReProCS are provided the same partial subspace knowledge $\mathbf{G}$.

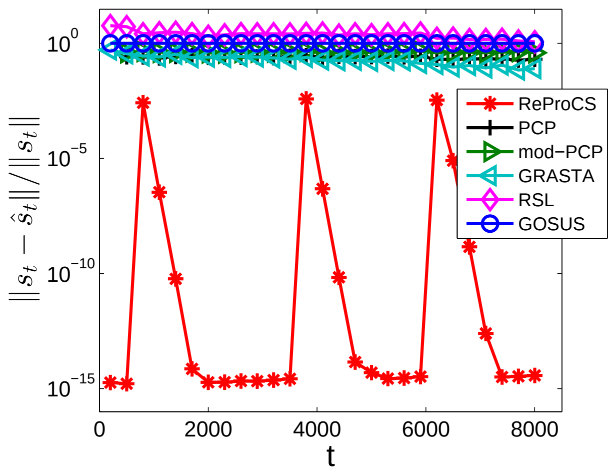
B. Video experiments

We show comparisons on two real video sequences. These are originally taken from http://perception.i2r.a-star.edu.sg/bk_model/bk_index.html and http://research.microsoft.com/en-us/um/people/jckrumm/wallflower/testimages.htm, respectively. The first is an indoor video of window curtains moving due to the wind. There was also some lighting variation. The latter part of this sequence also contains a foreground (various persons coming in, writing on the board and leaving). For $t > 1755$, in the foreground, a person with a black shirt walks in, writes on the board and then walk out, then a second person with a white shirt does the same and then a third person with a white shirt does the same. This video is challenging because (i) the white shirt color and the curtains' color is quite similar, making the corresponding S_t small in magnitude; and (ii) because the background variations are quite large while the foreground person moves slowly. As can be seen from Fig. **??**, ReProCS's performance is significantly better than that of the other algorithms for both foreground and background recovery. This is most easily seen from the recovered background images. One or more frames of the background recovered by PCP, RSL and GRASTA contains the person, while none of the ReProCS ones does.

The second sequence consists of a person entering a room containing a computer monitor that contains a white moving region. Background changes due to lighting variations and due to the computer monitor. The person moving in the foreground occupies a very large part of the image, so this is an example of a sequence in which the use of weighted ℓ_1 is essential (the



(a) uniformly distributed $\mathcal{T}_t$'s, slow subspace change does not hold



(b) correlated $\mathcal{T}_t$'s, slow subspace change holds

Fig. 4. NRMSE of sparse part $(n = 256,\ J = 3,\ r_0 = 40,\ t_0 = 200,\ c_{j,\text{new}} = 4,\ c_{j,\text{old}} = 4,\ j = 1, 2, 3)$

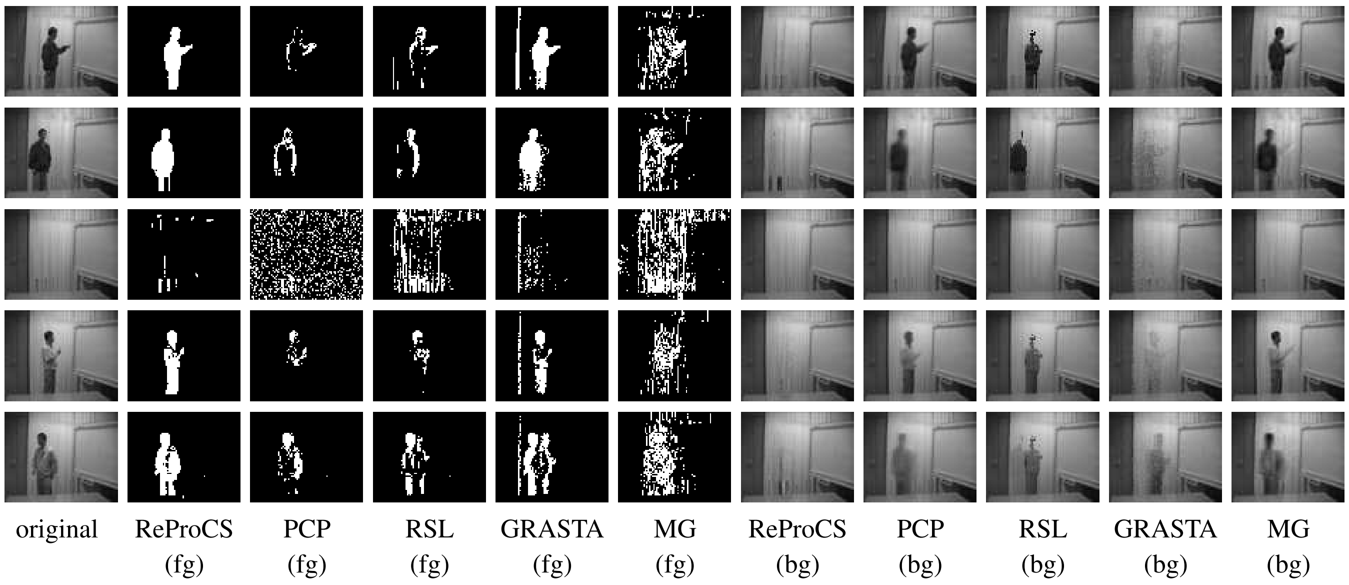


Fig. 5. Original video sequence at $t = t_{\text{train}} + 60, 120, 199, 475, 1148$ and its foreground (fg) and background (bg) layer recovery results using ReProCS (ReProCS-pCA) and other algorithms. For fg, we only show the fg support in white for ease of display.

support size is too large for simple ℓ_1 to work). As can be seen from Fig. **??**, for most frames, ReProCS is able to recover the person correctly. However, for the last few frames which consist of the person in a white shirt in front of the white part of the screen, the resulting S_t is too small even for ReProCS to correctly recover. The same is true for the other algorithms. Videos of all above experiments and of a few others are posted at http://www.ece.iastate.edu/~hanguo/PracReProCS.html.

Time Comparisons. The time comparisons are shown in Table **??**. In terms of speed, GRASTA is the fastest even though its performance is much worse. ReProCS is the second fastest. We expect that ReProCS can be speeded up by using mex files (C/C++ code) for the subspace update step. PCP and RSL are slower because they jointly process the entire image sequence. Moreover, ReProCS and GRASTA have the advantage of being recursive methods, i.e. the foreground/background recovery is available as soon as a new frame appears while PCP or RSL need to wait for the entire image sequence.

VII. Conclusions and Future Work

In this bookchapter we provided an overview of our recent work on the online or recursive robust PCA problem. We explained both the key idea of the ReProCS algorithm proposed in our recent work to solve this problem and how to develop its practical modification that can be directly used with real datasets. Extensive numerical experiments - both with simulated data and real videos - were shown demonstrating the advantage of ReProCS over many existing algorithms for robust PCA

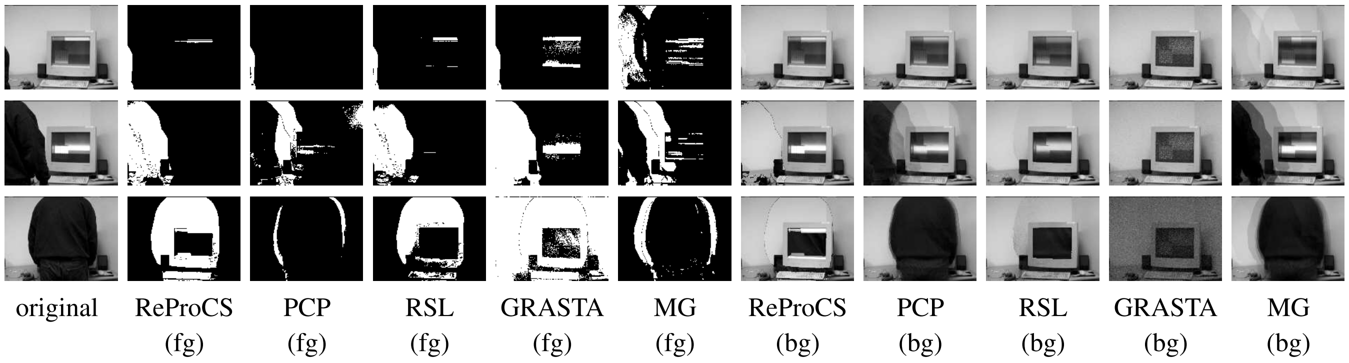


Fig. 6. Original video sequence at $t = t_{\text{train}} + 42, 44, 52$ and its foreground (fg) and background (bg) layer recovery results using ReProCS (ReProCS-pCA) and other algorithms. For fg, we only show the fg support in white for ease of display.

DataSet	Image Size	Sequence Length	ReProCS-pPCA	ReProCS-Recursive-PCA	PCP	RSL	GRASTA
Lake	72×90	$1420 + 80$	$2.99 + 19.97$ sec	$2.99 + 19.43$ sec	245.03 sec	213.36 sec	$39.47 + 0.42$ sec
Curtain	64×80	$1755 + 1209$	$4.37 + 159.02$ sec	$4.37 + 157.21$ sec	1079.59 sec	643.98 sec	$40.01 + 5.13$ sec
Person	120×160	$200 + 52$	$0.46 + 42.43$ sec	$0.46 + 41.91$ sec	27.72 sec	121.31 sec	$13.80 + 0.64$ sec

TABLE I
COMPARISON OF SPEED OF DIFFERENT ALGORITHMS. EXPERIMENTS WERE DONE ON A 64 BIT WINDOWS 8 LAPTOP WITH 2.40GHZ I7 CPU AND 8G RAM. SEQUENCE LENGTH REFERS TO THE LENGTH OF SEQUENCE FOR TRAINING PLUS THE LENGTH OF SEQUENCE FOR SEPARATION. FOR REPROCS AND GRASTA, THE TIME IS SHOWN AS TRAINING TIME + RECOVERY TIME.

- both batch and recursive. We also summarized and discussed the performance guarantees for ReProCS in detail. Lastly we briefly described our work on modified-PCP which is a piecewise batch approach that removes a key limitation of ReProCS (it allows a looser bound on the rank-sparsity product as compared with ReProCS), however it retains a key limitation of PCP (like PCP it also cannot handle highly correlated support change as well as ReProCS). Its correctness result was also given and discussed.

The tools introduced to prove the correctness result for ReProCS in [?], [?], [?] can also be used to get a correctness result for a practical modification of ReProCS with cluster-PCA (ReProCS-cPCA) which is Algorithm 2 of [?]. This algorithm was introduced to also remove the deleted directions from the subspace estimate. It does this by re-estimating the previous subspace at a time after the newly added subspace has been accurately estimated (i.e. at a time after $\hat{t}_j + K\alpha$). A partial result for this algorithm was proved in [?]. This result will need one extra assumption – it will need the eigenvalues of the covariance matrix of L_t to be clustered for a period of time after the subspace change has stabilized, i.e. for a period of d_2 frames in the interval $[t_j + d + 1, t_{j+1} - 1]$ – but it will have a key advantage. It will need a much weaker denseness assumption and hence a much weaker bound on r or r_{mat}. In particular, with this result we expect to be able to allow $r = r_{\text{mat}} \in \mathcal{O}(n)$ with the same assumptions on s and s_{mat} that we currently allow. This requirement is almost as weak as that of PCP.

The results for ReProCS presented here assume that the L_t's are independent over time and zero mean; this is a valid model when background images have independent random variations about a fixed mean. Using the tools developed in these works, a similar result can also be obtained for the more general case of L_t's following an autoregressive model. This will allow the L_t's to be correlated over time. A partial result for this case was obtained in [?]. The main change in this case will be that we will need to apply the matrix Azuma inequality from [?] instead of matrix Hoeffding. This is will also require algebraic manipulation of sums and some other important modifications, as explained in [?], so that the constant term after conditioning on past values of the matrix is small.

We expect that the tools introduced in [?], [?], [?] can also be used to analyze the noisy case, i.e. the case of $M_t = S_t + L_t + \boldsymbol{w}_t$ where $\boldsymbol{w}_t$ is small bounded noise. In most practical video applications, while the foreground is truly sparse, the background is only approximately low-rank. The modeling error can be treated as as $\boldsymbol{w}_t$. The proposed algorithms already apply without modification to this case (see [?] for results on real videos).

Finally, we expect both the algorithm and the proof techniques to apply with simple changes to the undersampled case $M_t = \boldsymbol{A}_t S_t + \boldsymbol{B}_t L_t + \boldsymbol{w}_t$ as long as $\boldsymbol{B}_t$ is *not* time-varying, i.e. $\boldsymbol{B}_t = \boldsymbol{B}_0$. A partial result for this case was obtained in [?] and experiments were shown in [?].

In other ongoing work, we are working to replace the ell-1 minimization in the projected sparse recovery step by algorithms designed under various other structured sparsity assumptions that may be valid for a given problem. For example, in [?], we introduced the support predicted modified-CS algorithm that uses a motion model to predict a moving object's support in the next frame and uses the predicted support as partial subspace knowledge for modified-CS. The motion model parameters are themselves tracked using a Kalman filter.