A FAST AND MEMORY-EFFICIENT ALGORITHM FOR ROBUST PCA (MEROP)

Praneeth Narayanamurthy and Namrata Vaswani

{pkurpadn, namrata} @iastate.edu,
Department of Electrical and Computer Engineering,
Iowa State University, Ames, IA

ABSTRACT

Robust PCA (RPCA) is the problem of separating a given data matrix into the sum of a sparse matrix and a low-rank matrix. Static RPCA is the RPCA problem in which the subspace from which the true data is generated remains fixed over time. Dynamic RPCA instead assumes that the subspace can change with time, although usually the changes are slow. We propose a Recursive Projected Compressed Sensing based algorithm called MERoP (Memory-Efficient Robust PCA) to solve the static RPCA problem. A simple extension of MERoP has been shown in our other work to also solve the dynamic RPCA problem. To the best of our knowledge, MERoP is the first online solution for RPCA that is provably correct under mild assumptions on input data and requires no assumption on intermediate algorithm estimates. Moreover, MERoP enjoys nearly-optimal memory complexity and is almost as fast as vanilla SVD. We corroborate our theoretical claims through extensive numerical experiments on both synthetic data and real videos.

***Index Terms*—** Online Robust PCA, Nearly memory-optimal

1. INTRODUCTION

Robust Principal Components Analysis (RPCA) is the problem of decomposing a given data matrix into the sum of a low-rank matrix (true data) and a sparse matrix (outliers). The column space of the low-rank matrix then gives the desired principal subspace (PCA solution). In recent years, the RPCA problem has been extensively studied, e.g., [1, 2, 3, 4, 5, 6]. A common application of RPCA is in video analytics in separating video into a slow-changing background image sequence and a foreground image sequence consisting of moving objects or people [7]. In this work we distinguish between the *static* and *dynamic* RPCA problems. This work develops a provably correct online solution for static RPCA.

Problem Statement. For *static RPCA*, at each time t we observe data vectors $\boldsymbol{y}_t \in \mathbb{R}^n$ that satisfy

$$\boldsymbol{y}_t = \boldsymbol{\ell}_t + \boldsymbol{x}_t, \ t = 1, 2, \dots d \tag{1}$$

where $\boldsymbol{x}_t$ is the sparse outlier vector and $\boldsymbol{\ell}_t$ is the true data vector that lies in a low-dimensional subspace of $\mathbb{R}^n$. To be precise, $\boldsymbol{\ell}_t = \boldsymbol{P}\boldsymbol{a}_t$ where $\boldsymbol{P}$ is an $n \times r$ *basis matrix*[1] with $r \ll n$. Here and below, $'$ denotes matrix transpose and $\|\cdot\|$ refers to the l_2 norm of a vector or the induced l_2 norm of a matrix. We use $\mathcal{T}_t$ to denote the support set of $\boldsymbol{x}_t$ and assume that $|\mathcal{T}_t| \le s$ for all t. Define the $n \times d$ data matrix $\boldsymbol{Y} := [\boldsymbol{y}_1, \boldsymbol{y}_2, \dots \boldsymbol{y}_d] = \boldsymbol{L} + \boldsymbol{X}$ where $\boldsymbol{L}, \boldsymbol{X}$ are similarly defined. Given an initial subspace estimate, $\hat{\boldsymbol{P}}_{\text{init}}$, the goal is to estimate $\text{span}(\boldsymbol{P})$ within ε-accuracy, quickly and provably. A by-product of doing this is that the true data vectors $\boldsymbol{\ell}_t$, the sparse outliers $\boldsymbol{x}_t$, and their support sets $\mathcal{T}_t$ can also be tracked on-the-fly. The initial subspace estimate, $\hat{\boldsymbol{P}}_{\text{init}}$, can be computed by applying any static (batch) RPCA technique, e.g., PCP [1] or AltProj [4], to the first t_{train} data frames, $\boldsymbol{Y}_{[1,t_{\text{train}}]}$. Here and below, $[a, b]$ refers to all integers between a and b, inclusive, $[a, b) := [a, b-1]$, and $\boldsymbol{M}_{\mathcal{T}}$ denotes a sub-matrix of $\boldsymbol{M}$ formed by columns indexed by entries in $\mathcal{T}$. For basis matrices $\hat{\boldsymbol{P}}, \boldsymbol{P}$, that are used to denote two r-dimensional subspaces, we use $\sin\theta_{\max}(\hat{\boldsymbol{P}}, \boldsymbol{P})$ to denote the *subspace error* [8] and it can be computed as $\sin\theta_{\max}(\hat{\boldsymbol{P}}, \boldsymbol{P}) = \sigma_{\max}((\boldsymbol{I} - \hat{\boldsymbol{P}}\hat{\boldsymbol{P}}')\boldsymbol{P})$. Here $\sigma_{\max}(\cdot)$ denotes the largest singular value.

Dynamic RPCA refers to the time-varying extension of static RPCA in which the subspace in which the true data lies can change with time, i.e., $\boldsymbol{\ell}_t = \boldsymbol{P}_{(t)}\boldsymbol{a}_t$ and $\boldsymbol{P}_{(t)}$ changes in such a way that $\boldsymbol{L}$ has a rank, $r_L \ll \min(n, d)$. The goal is to track $\boldsymbol{P}_{(t)}$ over time. To make the problem identifiable, one assumes that $\boldsymbol{P}_{(t)}$ is piecewise constant with time [9, 10, 11].

Contributions. To the best of our knowledge, there is no work even for static RPCA that simultaneously addresses the following questions. (A) What is the minimum number of samples needed to guarantee an ϵ-accurate solution in some metric? (B) Is it possible to obtain such an estimate whilst using nearly optimal memory? (C) Is it possible to design an algorithm that is nearly as fast as computing a vanilla rank r-SVD on the data matrix? In this paper, we seek to obtain positive answers to all the above questions. We propose a provably correct algorithm called MERoP (Memory-Efficient Robust PCA) to solve the static RPCA problem. MERoP relies on the recently introduced Recursive Projected Compressive Sensing (ReProCS) solution framework [12, 9] which was originally developed to solve the *dynamic RPCA* problem. In [13, 10, 11], we have obtained progressively simpler correctness guarantees for ReProCS for dynamic RPCA. In this work, we extend the ReProCS line of work to demonstrate that MERoP and its dynamic extension from [11] together solve both static and dynamic RPCA under weakened standard RPCA assumptions, a lower bound on most outlier magnitudes, and either fixed or slowly changing subspace of the true data. Moreover, unlike other existing work on RPCA [1, 2, 3, 5, 6], this is done in a fast, memory-efficient, and highly robust fashion.

In particular, the current paper shows that (i) MERoP has a running time of $\mathcal{O}(ndr \log(1/\epsilon))$, which is the cost of performing a rank-r vanilla SVD on the data matrix; (ii) can tolerate an order wise larger fraction of outlier per row than other RPCA solutions (can tolerate slow moving and occasionally static foreground objects' occlusions in videos) as long as one main extra assumption holds (most outlier magnitudes are lower bounded); and (iii) has nearly optimal memory complexity: we need $\mathcal{O}(nr \log n \log(1/\epsilon))$ memory to obtain an ϵ-accurate subspace estimate. This is larger than nr, which is the minimum required to even output a subspace estimate, by only logarithmic factors. Finally, if the application only requires solving

This work was supported by NSF grant CCF-1526870

[1]tall matrix with mutually orthonormal columns

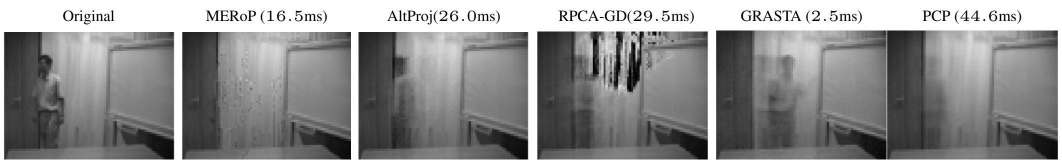

Fig. 1: Background recovery. Only MERoP background does not contain the person or even his shadow. MERoP is also faster than all except GRASTA. Time taken per frame is shown in parentheses. The output of GRASTA slightly lags and hence it appears that the person is sitting.

Algorithm 1 MERoP and Offline MERoP

```
1: **Input**: $\boldsymbol{y}_t$    **Output**: $\hat{\boldsymbol{x}}_t, \hat{\boldsymbol{\ell}}_t, \hat{\boldsymbol{P}}$
2: **Initialization**: $\hat{\boldsymbol{P}}_{\text{init}} \leftarrow \text{SVD}_r[\text{AltProj}(\boldsymbol{Y}_{1:Cr})]$
3: **Parameters:** $K \leftarrow C\log(1/\varepsilon)$, $\alpha \leftarrow Cf^2 r\log n$, $\omega_{supp} \leftarrow x_{\min}/2$, $\xi \leftarrow x_{\min}/15$, $r$.
4: $\hat{\boldsymbol{P}}_{(t_{\text{train}})} \leftarrow \hat{\boldsymbol{P}}_{\text{init}}$; $k \leftarrow 1$.
5: **for** $t > t_{\text{train}}$ **do**
6:     $\boldsymbol{\Psi} \leftarrow \boldsymbol{I} - \hat{\boldsymbol{P}}_{(t-1)}\hat{\boldsymbol{P}}_{(t-1)}{}'$;
7:     $\tilde{\boldsymbol{y}}_t \leftarrow \boldsymbol{\Psi}\boldsymbol{y}_t$.
8:     $\hat{\boldsymbol{x}}_{t,cs} \leftarrow \arg\min_{\tilde{\boldsymbol{x}}} \|\tilde{\boldsymbol{x}}\|_1$ s.t $\|\tilde{\boldsymbol{y}}_t - \boldsymbol{\Psi}\tilde{\boldsymbol{x}}\| \leq \xi$.
9:     $\hat{\mathcal{T}}_t \leftarrow \{i : |\hat{\boldsymbol{x}}_{t,cs}| > \omega_{supp}\}$.
10:    $\hat{\boldsymbol{x}}_t \leftarrow \boldsymbol{I}_{\hat{\mathcal{T}}_t}(\boldsymbol{\Psi}_{\hat{\mathcal{T}}_t}{}'\boldsymbol{\Psi}_{\hat{\mathcal{T}}_t})^{-1}\boldsymbol{\Psi}_{\hat{\mathcal{T}}_t}{}'\tilde{\boldsymbol{y}}_t$.
11:    $\hat{\boldsymbol{\ell}}_t \leftarrow \boldsymbol{y}_t - \hat{\boldsymbol{x}}_t$.
12:    **if** $t = t_{\text{train}} + u\alpha$ for $u = 1,\ 2,\ \cdots,\ K$ **then**
13:        $\hat{\boldsymbol{P}}_k \leftarrow SVD_r[\hat{\boldsymbol{L}}_{t;\alpha}]$, $\hat{\boldsymbol{P}}_{(t)} \leftarrow \hat{\boldsymbol{P}}_k$, $k \leftarrow k+1$.
14:    **else**
15:        $\hat{\boldsymbol{P}}_{(t)} \leftarrow \hat{\boldsymbol{P}}_{(t-1)}$
16:    **end if**
17:    **if** $t = t_{\text{train}} + K\alpha$ **then**
18:        $\hat{\boldsymbol{P}} \leftarrow \hat{\boldsymbol{P}}_k$
19:        $\boldsymbol{\Psi} \leftarrow \boldsymbol{I} - \hat{\boldsymbol{P}}\hat{\boldsymbol{P}}'$
20:    **end if**
21: **end for**
22: **for** $t > t_{\text{train}}$ **do**                              }
23:     $\hat{\boldsymbol{x}}_t \leftarrow \boldsymbol{I}_{\hat{\mathcal{T}}_t}(\boldsymbol{\Psi}_{\hat{\mathcal{T}}_t}{}'\boldsymbol{\Psi}_{\hat{\mathcal{T}}_t})^{-1}\boldsymbol{\Psi}_{\hat{\mathcal{T}}_t}{}'\boldsymbol{y}_t$    } Offline MERoP.
24:     $\hat{\boldsymbol{\ell}}_t \leftarrow \boldsymbol{y}_t - \hat{\boldsymbol{x}}_t$.                    }
25: **end for**                                              }
```

the static PCA problem (and not estimating $\boldsymbol{\ell}_t$ or $\boldsymbol{x}_t$), the run time of MERoP is $\mathcal{O}(nr^2\log n\log^2(1/\epsilon))$ which is the fastest among all existing RPCA solutions. This is possible because we can use only the first $Cr\log n\log(1/\epsilon)$ samples to obtain an ϵ-accurate estimate of $\boldsymbol{P}$, and stop the algorithm after these many time samples.

2. ALGORITHM AND MAIN RESULT

2.1. MERoP Algorithm

Algorithm 1 proceeds as follows. A coarse subspace estimate is obtained using PCP [1] or AltProj [4] applied to the first $t_{\text{train}} = Cr$ frames $\boldsymbol{Y}_{[1,t_{\text{train}}]}$. For $t > t_{\text{train}}$, the algorithm proceeds as follows. At time t, let $\hat{\boldsymbol{P}}_{(t-1)}$ denote the subspace estimate from $(t-1)$. If this estimate is accurate enough, projecting $\boldsymbol{y}_t := \boldsymbol{x}_t + \boldsymbol{\ell}_t$ onto its orthogonal complement will nullify most of $\boldsymbol{\ell}_t$. We compute $\tilde{\boldsymbol{y}}_t := \boldsymbol{\Psi}\boldsymbol{y}_t$ where $\boldsymbol{\Psi} := \boldsymbol{I} - \hat{\boldsymbol{P}}_{(t-1)}\hat{\boldsymbol{P}}_{(t-1)}{}'$. Thus, $\tilde{\boldsymbol{y}}_t = \boldsymbol{\Psi}\boldsymbol{x}_t + \boldsymbol{b}_t$ where $\boldsymbol{b}_t := \boldsymbol{\Psi}\boldsymbol{\ell}_t$ and $\|\boldsymbol{b}_t\|$ is small. Recovering $\boldsymbol{x}_t$ from $\tilde{\boldsymbol{y}}_t$ is thus a regular compressive sensing (CS) / sparse recovery problem in small noise [14]. We compute $\hat{\boldsymbol{x}}_{t,cs}$ using l_1 minimization followed by thresholding based support estimation to get $\hat{\mathcal{T}}_t$. A Least Squares (LS) based debiasing step on $\hat{\mathcal{T}}_t$ returns the final $\hat{\boldsymbol{x}}_t$. We then estimate $\boldsymbol{\ell}_t$ as $\hat{\boldsymbol{\ell}}_t = \boldsymbol{y}_t - \hat{\boldsymbol{x}}_t$. The $\hat{\boldsymbol{\ell}}_t$'s are used to update the subspace estimate. This is done using K steps of r-SVD, each done with a new set of α frames of $\hat{\boldsymbol{\ell}}_t$. Here r-SVD means compute the top r left singular vectors of $\hat{\boldsymbol{L}}_{t;\alpha} := [\hat{\boldsymbol{\ell}}_{t-\alpha+1}, \hat{\boldsymbol{\ell}}_{t-\alpha+2}, \ldots, \hat{\boldsymbol{\ell}}_t]$.

Using the result of [15] we show that with high probability, the subspace estimation error decreases exponentially after every α frames, and thus, after K-SVD steps, we obtain an ε-accurate estimate of the subspace. In offline MERoP, we use this to re-estimate all the previous $\hat{\boldsymbol{x}}_t$'s and $\hat{\boldsymbol{\ell}}_t$'s which can also shown to be ε-accurate estimates.

2.2. Assumptions and Main Result

Assumption on principal subspace coefficients $\boldsymbol{a}_t$. We assume that the $\boldsymbol{a}_t$'s are zero mean, *element-wise bounded* and *mutually independent* random variables (r.v.) with *identical* covariance matrices $\boldsymbol{\Lambda}$, and that $\boldsymbol{\Lambda}$ is diagonal. Since the $\boldsymbol{a}_t$'s are element-wise bounded, there exists an $\eta < \infty$, such that $\max_{j=1,2,\ldots r}\max_t \frac{(\boldsymbol{a}_t)_j^2}{\lambda_j(\boldsymbol{\Lambda})} \leq \eta$, where $\lambda_j(\cdot)$ refers to the j-th largest eigenvalue of the matrix. For most bounded distributions, η is a little more than one, e.g., if the entries of $\boldsymbol{a}_t$ are zero mean uniform, then $\eta = 3$. The bounded-ness assumption along with mutual independence and identical covariances is similar to the right singular vectors' incoherence assumption needed by all the other RPCA solutions [3, 4]. There are minor differences since we impose statistical assumptions on $\boldsymbol{a}_t$'s. See [11, Remark C.1]

Incoherence of left singular vectors of $\boldsymbol{L}$. In order to separate the $\boldsymbol{\ell}_t$'s from the sparse outliers $\boldsymbol{x}_t$, we need to assume that the $\boldsymbol{\ell}_t$'s are themselves not sparse (thus we sometimes refer to this property as "denseness"). This is ensured if we assume that column vectors of $\boldsymbol{P}$ are dense enough. To quantify this, we define μ as the smallest real number that satisfies

$$\max_{i=1,2,\ldots,n}\|\boldsymbol{I}_i{}'\boldsymbol{P}\| \leq \sqrt{\frac{\mu r}{n}} \tag{2}$$

The above assumption is also referred to as *left incoherence* and is needed by all existing RPCA solutions [3, 4].

Bound on outlier fractions. Similar to earlier RPCA results, we also need outlier fractions to be bounded. However, we need different bounds on this fraction per-column and per-row. The row bound can be much larger. Let max-outlier-frac-col $:= \max_t |\mathcal{T}_t|/n$ denote the maximum outlier fraction in any column of $\boldsymbol{Y}$. Since MERoP is a nearly-online algorithm, we need the fraction of outliers per row of a sub-matrix of $\boldsymbol{Y}$ with α consecutive columns to be bounded. To quantify this, for a time interval, $\mathcal{J}$, define

$$\gamma(\mathcal{J}) := \max_{i=1,2,\ldots,n}\frac{1}{|\mathcal{J}|}\sum_{t\in\mathcal{J}}\boldsymbol{1}_{\{i\in\mathcal{T}_t\}}. \tag{3}$$

where $\boldsymbol{1}_S$ is the indicator function for event S. Thus $\gamma(\mathcal{J})$ is the maximum outlier fraction in any row of the sub-matrix $\boldsymbol{Y}_{\mathcal{J}}$ of $\boldsymbol{Y}$.

Table 1: Comparing assumptions, time and memory complexity. For simplicity, we ignore all dependence on condition numbers.

Algorithm	Outlier tolerance, rank of $(\boldsymbol{L})$	Assumptions	Memory, Time,	# params.
PCP(C)[1] (offline)	max-outlier-frac-row $= \mathcal{O}(1)$ max-outlier-frac-col $= \mathcal{O}(1)$ $r \leq \frac{c \min(n,d)}{\log^2 n}$	strong incoh, unif. rand. support,	Mem: $\mathcal{O}(nd)$ Time: $\mathcal{O}(nd^2 \frac{1}{\epsilon})$	zero
AltProj[4], (offline)	max-outlier-frac-row $= O(1/r)$ max-outlier-frac-col $= O(1/r)$		Mem: $\mathcal{O}(nd)$ Time: $\mathcal{O}(ndr^2 \log \frac{1}{\epsilon})$	2
RPCA-GD [5] (offline)	max-outlier-frac-row $= \mathcal{O}(1/r^{1.5})$ max-outlier-frac-col $= \mathcal{O}(1/r^{1.5})$		Mem: $\mathcal{O}(nd)$ Time: $\mathcal{O}(ndr \log \frac{1}{\epsilon})$	5
PG-RMC [6] (offline)	max-outlier-frac-row $= O(1/r)$ max-outlier-frac-col $= \mathcal{O}(1/r)$	$d \approx n$	Mem: $\mathcal{O}(nd)$ Time: $\mathcal{O}(nr^3 \log^2 n \log^2 \frac{1}{\epsilon})$	4
MERoP **(this work)** (*online*)	**max-outlier-frac-row** $= \mathcal{O}(1)$ **max-outlier-frac-col** $= \mathcal{O}(1/r)$	$\boldsymbol{a}_t$'s independent, init data: AltProj assu's, outlier mag. lower bounded fixed or slowly changing subspace	**Mem:** $\mathcal{O}(nr \log n \log \frac{1}{\epsilon})$ **Time:** $\mathcal{O}(ndr \log \frac{1}{\epsilon})$	4
MERoP-only-PCA			**Time:** $\mathcal{O}\left(nr^2 \log n \log^2 \frac{1}{\epsilon}\right)$	

Note: The table assumes an $n \times d$ data matrix $\boldsymbol{Y} := \boldsymbol{L} + \boldsymbol{X}$, where $\boldsymbol{L}$ has rank r and the outlier matrix $\boldsymbol{X}$ is sparse. It compares a few provable methods versus MERoP for solving the original robust PCA problem. Recall that the subspace that MERoP recovers is of dimension r. With only a mild extra assumption on outlier magnitudes, MERoP is able to achieve a significant gain in outlier tolerance per row. We also note that all the algorithms require *left and right* incoherence, and thus we do not list this in the third column.

Let $\mathcal{J}^\alpha$ denote a time interval of duration α. We will bound

$$\text{max-outlier-frac-row} := \max_{\mathcal{J}^\alpha \subseteq [t_{\text{train}}, d]} \gamma(\mathcal{J}^\alpha). \quad (4)$$

Initialization. Assume that the initial data, $\boldsymbol{Y}_{[1,t_{\text{train}}]}$ (with $t_{\text{train}} = Cr$), satisfies PCP(H) [3] or AltProj [4] assumptions: (i) let $\boldsymbol{L}_{\text{init}} \overset{\text{SVD}}{=} \boldsymbol{P\Sigma V}'$; $\boldsymbol{P}$ and $\boldsymbol{V}$ satisfy incoherence with parameter μ, and (ii) outlier fractions per row and per column are both upper bounded by $c/(\mu r)$; and, (iii) we run enough iterations of AltProj so that the initial subspace estimate satisfies $\sin\theta_{\max}(\hat{\boldsymbol{P}}_{\text{init}}, \boldsymbol{P}) \leq 0.05/\sqrt{r}$. This can be satisfied by running $c \log r$ iterations to ensure that $\sin\theta_{\max}(\hat{\boldsymbol{P}}_{\text{init}}, \boldsymbol{P}) = \mathcal{O}(1/\sqrt{r})$.

We use λ^+ and λ^- to denote the maximum and minimum eigenvalues of $\boldsymbol{\Lambda}$ and let $f := \lambda^+/\lambda^-$ denote the condition number.

Theorem 2.1 (Static RPCA). *Pick an $\varepsilon > 0$. For $t > t_{\text{train}}$, assume*

1. *assumptions on $\boldsymbol{a}_t$'s hold,*
2. *the initialization assumption given above on $\hat{\boldsymbol{P}}_{\text{init}}$ holds,*
3. $0.75\sqrt{\eta\lambda^+} \leq x_{\min}/15$,
4. *max-outlier-frac-row $\leq b_0$ where $b_0 = 0.02/f^2$, and max-outlier-frac-col $\leq 0.09/(\mu r)$,*

Then, with probability at least $1 - 10dn^{-10}$, the output of Offline MERoP satisfies $\sin\theta_{\max}(\hat{\boldsymbol{P}}, \boldsymbol{P}) \leq \varepsilon$, $\|\hat{\boldsymbol{\ell}}_t - \boldsymbol{\ell}_t\| \leq \varepsilon\|\boldsymbol{\ell}_t\|$ and $\hat{\mathcal{T}}_t = \mathcal{T}_t$ for all t.

Proof: Proof is provided in the full version at [11]. It is a special case of Theorem 2.2 with $r \equiv r_L$ and $J \equiv 1$.

Theorem 2.2 (Dynamic RPCA ([11])). *Consider the dynamic RPCA problem and the dynamic extension of MERoP (Algorithm 2 of [11]). If slow subspace change holds, i.e., if $\boldsymbol{\ell}_t = \boldsymbol{P}_j \boldsymbol{a}_t$ for $t \in [t_j, t_j + 1)$ with $\sin\theta_{\max}(\boldsymbol{P}_{j-1}, \boldsymbol{P}_j) \leq 0.05\sqrt{r}$ and $t_{j+1} - t_j \geq Cr \log n \log(1/\varepsilon)$; and all other assumptions of Theorem 2.1 hold, then with the same probability, at all times t, offline MEDRoP ensures $\|\hat{\boldsymbol{\ell}}_t - \boldsymbol{\ell}_t\| \leq \varepsilon\|\boldsymbol{\ell}_t\|$, $\hat{\mathcal{T}}_t = \mathcal{T}_t$, and $\sin\theta_{\max}(\hat{\boldsymbol{P}}_{(t)}, \boldsymbol{P}_{(t)}) \leq \varepsilon$.*

Remark 2.3. *The outlier magnitudes lower bound assumption of Theorem 2.1 can be relaxed to a lower bound on most outlier magnitudes. In particular, the following suffices: Let $\Delta := \sin\theta_{\max}(\hat{\boldsymbol{P}}_{\text{init}}, \boldsymbol{P})$; assume that $\boldsymbol{x}_t$ can be split into $\boldsymbol{x}_t = (\boldsymbol{x}_t)_{small} + (\boldsymbol{x}_t)_{large}$ that are such that, in the k-th subspace update interval, $\|(\boldsymbol{x}_t)_{small}\| \leq 0.3^{k-1}(\varepsilon + \Delta)\sqrt{r\lambda^+}$ and the smallest nonzero entry of $(\boldsymbol{x}_t)_{large}$ is larger than $C0.3^{k-1}(\varepsilon + \Delta)\sqrt{r\lambda^+}$. If there were a way to bound the element-wise error of the CS step (instead of the l_2 norm error), the above requirement could be relaxed further. A similar remark applies to Theorem 2.2 too.*

Remark 2.4 (MERoP-Only-PCA). *In applications such as "robust" dimensionality reduction [16, 17] where the objective is to just obtain the top-r directions along which the variability of data is maximized (Principal Components Analysis), we only need the first $K\alpha = Cr \log n \log(1/\varepsilon)$ samples to obtain an ε-accurate subspace estimate. Thus, the run-time of MERoP in this application reduces to $\mathcal{O}(nr^2 \log n \log^2(1/\varepsilon))$ which is, in fact, faster than PG-RMC by a factor of $\mathcal{O}(r \log n)$.*

3. DISCUSSION

There is a vast body of literature on analyzing RPCA. The first papers to provide guarantees for RPCA were [1, 2] which studied a convex relaxation to recover the "sparsest" $\boldsymbol{X}$ and the "least-rank" $\boldsymbol{L}$ using the l_1 norm, and the nuclear norm, respectively. Although these ideas offer an elegant theory, they are slow in practice: to obtain an ϵ-accurate solution, the time complexity is $\mathcal{O}(nd^2/\epsilon)$. Since then, there has been a rich development of faster and more efficient algorithms to attack this problem. In particular, AltProj [4] was one

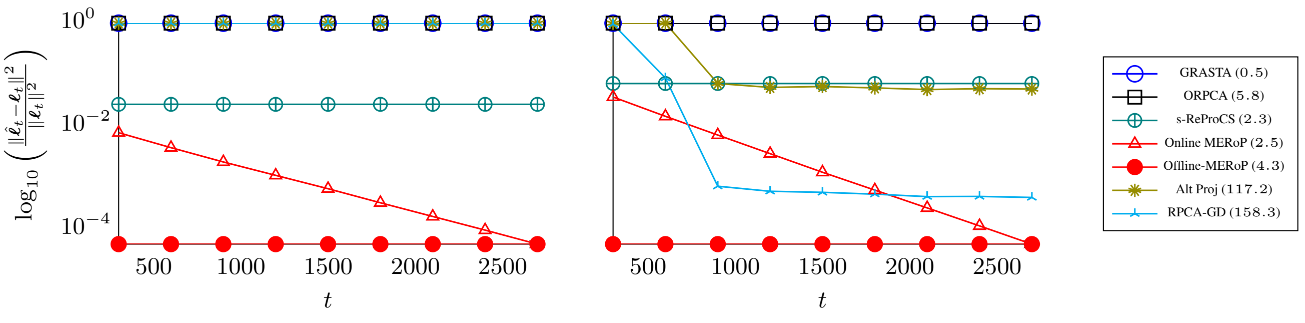

Fig. 2: Relative error in recovering ℓ_t's. Left: Moving object model on $\mathcal{T}_t$. Right: Bernoulli model on $\mathcal{T}_t$. Time taken per frame in milliseconds (ms) for the Bernoulli model is shown in parentheses in the legend. The errors are plotted every $k\alpha - 1$ time-frames. Observe the nearly-exponential decay of the subspace error with time.

of the first algorithms to study a provable non-convex focmulation of RPCA. Their method relied on starting with an initial guess of the sparse outliers (which consisted of the "largest entries" of the observed matrix), followed by alternately projecting the "residuals" onto the non-convex sets of sparse, and, low-rank matrices. To circumvent the problem of sensitivity of the SVD to outliers, the algorithm proceeds in stages; incrementing the "target rank" at each stage until a halting criterion is attained. This algorithm was shown to be significantly faster than the convex approaches and enjoyed a run-time of $\mathcal{O}(ndr^2 \log(1/\epsilon))$. Following this, there have been additional improvements to the running time of subsequent algorithms. A recent work RPCA-GD [5] showed that it is indeed possible to design an algorithm that runs in time $\mathcal{O}(ndr \log(1/\epsilon))$. However, this approach requires a tighter bound on the allowed outlier-fractions (see Table 1). A recently proposed algorithm, PG-RMC [6] is the fastest exisiting batch algorithm for the RPCA with a run time of $\mathcal{O}(nr^3 \log^2 n \log^2(1/\epsilon))$. The reason for the nearly-linear speed is that PG-RMC uniformly-randomly selects a few entries of the data matrix, and thus needs to work with a fewer number of samples. Although this presents a significant improvement in applications where running time is a severe bottleneck, two notable disadvantages of this approach are that (i) one cannot recover the sparse matrix due to the uniform random sampling of the data matrix and (ii) it requires d to be of the same order as n. This is a very restrictive requirement in the high-dimensional regime. For a summary, see Table 1.

Solutions to RPCA in the online setting have also received considerable interest. We discuss some relevant algorithms and note that this is by no means an exhaustive list. Firstly, there are algorithms based on the GRASTA framework, which is similar to the ReProCS framework [12, 9] in the sense of alternating sparse recovery and subspace update for incoming data; but use stochastic Gradient Descent for the latter [18, 19]. Although this approach is very fast in practice, the authors do not provide theoretical guarantees and the algorithm performance is not good for large sized or slow moving foregrounds (see Section 4). Another approach is the "online-reformulation" of PCP [20], based on stochastic optimization techniques. This provides a *partial guarantee* which imposes an unnatural assumption on intermediate algorithm estimates (assumes that the intermediate subspace estimates are full rank), and additionally only provides an asymptotic convergence guarantee.

4. EMPIRICAL EVALUATION

In this section we present the results of numerical experiments[2] to compare the performance of MERoP with existing state-of-the-art algorithms on synthetic data and in the task of Foreground-Background separation in real videos. All experiments are performed on a Desktop Computer with Intel® Xeon E3-1240 8-core CPU @ 3.50GHz and 32GB RAM.

Synthetic Data. We perform an experiment on synthetic data to demonstrate the superiority of MERoP over existing algorithms. We generate the data as follows. $\boldsymbol{P}$ is generated by ortho-normalizing the columns of an $n \times r$ i.i.d standard normal matrix. We used $n = 1000$, $r = 30$, $d = 3000$. For the low-rank matrix $\boldsymbol{L}$ we generate the coefficients $\boldsymbol{a}_t \in \mathbb{R}^r$ according to $(\boldsymbol{a}_t)_i \overset{i.i.d}{\sim} Unif[-q_i, q_i]$ where $q_i = \sqrt{f} - \sqrt{f}i/2r$ for $i = 1, 2, \cdots, r-1$ and $q_r = 1$. thus the condition number is f and we selected $f = 50$. We used the first $t_{\text{train}} = 300$ frames as the training part, where we generated a smaller fraction of outliers. For the moving object model (see Appendix [10, Model G.24]) with parameters $s/n = 0.01$, $b_0 = 0.01$ and for $t > t_{\text{train}}$ we used $s/n = 0.05$ and $b_0 = 0.3$. For the Bernoulli model we set $\rho = 0.01$ for the first t_{train} frames and $\rho = 0.3$ for the subsequent frames. The sparse outlier magnitudes are generated uniformly at random from the interval $[x_{\min}, x_{\max}]$ with $x_{\min} = 10$ and $x_{\max} = 20$ in both experiments. The results are averaged over 50 independent trials. The results are shown in Fig. 2.

We initialized MERoP and s-ReProCS [10] using AltProj [4] applied to $\boldsymbol{Y}_{[1,t_{\text{train}}]}$. The smaller outlier fraction helped achieve $\sin\theta_{\max}(\hat{\boldsymbol{P}}_{\text{init}}, \boldsymbol{P}) \approx 10^{-3}$. For the batch methods used in the comparisons – PCP, AltProj and RPCA-GD, we implement the algorithms on $\boldsymbol{Y}$. Further, we set the regularization parameter for PCP $1/\sqrt{n}$ in accordance with [1]. The other known parameters, r for Alt-Proj, outlier-fraction for RPCA-GD, are set equal to the true values. For online methods we implement ORPCA by [20] and GRASTA by [18]. The regularization parameter for ORPCA was set as with $\lambda_1 = 1/\sqrt{n}$ and $\lambda_2 = 1/\sqrt{d}$ according to [20].

We observe that since the outlier fraction per row is significantly higher, all other techniques fail for the Moving Object model, whereas for the Bernoulli model, RPCA-GD is able to obtain meaningful estimates at the cost of being very slow.

Video Experiments. We also illustrate the efficacy of MERoP on real videos in this section. In particular, we implement several algorithms on two datasets, `MR` (Meeting Room) and `SL` (Switch Light) which are a few commonly accepted benchmark datasets in background separation. We show one recovered background frame for each video in Fig. 1. All algorithms used $r = 40$ and default parameters in their codes. MERoP used $\alpha = 60$, $K = 3$, $\xi_t = \|\boldsymbol{\Psi}\hat{\boldsymbol{\ell}}_{t-1}\|$, $\omega_{supp} = \|\boldsymbol{y}_t\|/\sqrt{n}$. ORPCA failed completely, gave a black background and hence, it is not shown. Time taken in milliseconds per frame is shown above each image.

[2]The codes are available for download at https://github.com/praneethmurthy/MERoP

5. REFERENCES

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

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

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

[4] P. Netrapalli, U N Niranjan, S. Sanghavi, A. Anandkumar, and P. Jain, "Non-convex robust pca," in *Neural Info. Proc. Sys. (NIPS)*, 2014.

[5] X. Yi, D. Park, Y. Chen, and C. Caramanis, "Fast algorithms for robust pca via gradient descent," in *Neural Info. Proc. Sys. (NIPS)*, 2016.

[6] Yeshwanth Cherapanamjeri, Kartik Gupta, and Prateek Jain, "Nearly-optimal robust matrix completion," in *ICML*, 2017.

[7] F. Seidel, C. Hage, and M. Kleinsteuber, "pROST: a smoothed ℓ_p-norm robust online subspace tracking method for background subtraction in video," *Machine vision and applications*, vol. 25, no. 5, pp. 1227–1240, 2014.

[8] Ke Ye and Lek-Heng Lim, "Schubert varieties and distances between subspaces of different dimensions," *SIAM Journal on Matrix Analysis and Applications*, vol. 37, no. 3, pp. 1176–1197, 2016.

[9] C. Qiu, N. Vaswani, B. Lois, and L. Hogben, "Recursive robust pca or recursive sparse recovery in large but structured noise," *IEEE Trans. Info. Th.*, pp. 5007–5039, August 2014.

[10] P. Narayanamurthy and N. Vaswani, "Provable Dynamic Robust PCA or Robust Subspace Tracking," *arXiv:1705.08948*, 2017.

[11] P. Narayanamurthy and N. Vaswani, "Nearly Optimal Robust Subspace Tracking and Dynamic Robust PCA," *arXiv:1712.06061*, 2017.

[12] C. Qiu and N. Vaswani, "Real-time robust principal components' pursuit," in *Allerton Conf. on Communication, Control, and Computing*, 2010.

[13] J. Zhan, B. Lois, H. Guo, and N. Vaswani, "Online (and Offline) Robust PCA: Novel Algorithms and Performance Guarantees," in *Intnl. Conf. Artif. Intell. and Stat. (AISTATS)*, 2016.

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

[15] N. Vaswani and P. Narayanamurthy, "PCA in Data-Dependent Noise: Nearly Optimal Finite Sample Guarantees," *arXiv:1702.03070*, 2017.

[16] E. Keogh, K. Chakrabarti, M. Pazzani, and S. Mehrotra, "Dimensionality reduction for fast similarity search in large time series databases," *Knowledge and information Systems*, vol. 3, no. 3, pp. 263–286, 2001.

[17] J. C. Harsanyi and C. I. Chang, "Hyperspectral image classification and dimensionality reduction: an orthogonal subspace projection approach," *IEEE Transactions on geoscience and remote sensing*, vol. 32, no. 4, pp. 779–785, 1994.

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

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

[20] J. Feng, H. Xu, and S. Yan, "Online robust pca via stochastic optimization," in *Adv. Neural Info. Proc. Sys. (NIPS)*, 2013.