Nearly Optimal Robust Subspace Tracking

Praneeth Narayanamurthy [1] Namrata Vaswani [1]

Abstract

Robust subspace tracking (RST) can be simply understood as a dynamic (time-varying) extension of robust PCA. More precisely, it is the problem of tracking data lying in a fixed or slowly-changing low-dimensional subspace while being robust to sparse outliers. This work develops a recursive projected compressive sensing algorithm called "Nearly Optimal RST (NORST)", and obtains one of the first guarantees for it. We show that NORST provably solves RST under weakened standard RPCA assumptions, slow subspace change, and a lower bound on (most) outlier magnitudes. Our guarantee shows that (i) NORST is online (after initialization) and enjoys near-optimal values of tracking delay, lower bound on required delay between subspace change times, and of memory complexity; and (ii) it has a significantly improved worst-case outlier tolerance compared with all previous robust PCA or RST methods without requiring any model on how the outlier support is generated.

1. Introduction

According to its modern definition (Candès et al., 2011), *robust PCA (RPCA)* is the problem of decomposing a 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). A common application is in video analytics in separating a video into a slow-changing background image sequence (modeled as a low-rank matrix) and a foreground image sequence consisting of moving objects or people (sparse) (Candès et al., 2011). Many fast and provably correct RPCA approaches have appeared in recent years: PCP (Candès et al., 2011; Chandrasekaran et al., 2011; Hsu et al., 2011), AltProj (Netrapalli et al., 2014), RPCA-GD (Yi et al., 2016), NO-RMC (Cherapanamjeri et al., 2016).

Robust Subspace Tracking (RST) can be simply interpreted as a dynamic (time-varying) extension of RPCA. It assumes that the true data lie in a low-dimensional subspace that can change with time, albeit slowly. The goal is to track this changing subspace over time in the presence of sparse outliers. The offline version of this problem can be called *dynamic (or time-varying) RPCA*. RST requires the tracking delay to be small, while dynamic RPCA does not. Time-varying subspaces is a more appropriate model for long data sequences, e.g., long surveillance videos. The reason is, if one tries to use a single lower dimensional subspace to represent the entire data sequence, the required dimension may end up being quite large. Short tracking delays are critical for applications where real-time or near real-time estimates are needed, e.g., video-based surveillance/tracking, or detecting dynamic social network anomalies (Ozdemir et al., 2017). While standard RPCA has been extensively studied in recent years, there is much lesser work on provable dynamic RPCA or RST and only includes original-ReProCS (Qiu et al., 2014; Lois & Vaswani, 2015; Zhan et al., 2016), modified-PCP (Zhan & Vaswani, 2015), and simple-ReProCS (Narayanamurthy & Vaswani, 2018a). Another related approach that comes with a partial guarantee[1] is ORPCA (Feng et al., 2013).

Modeling subspace change. All existing guarantees for subspace tracking algorithms (except our previous RST work mentioned above) assume the statistically stationary setting of data being generated from a *single* unknown subspace; and almost all of them are partial guarantees. All work on subspace tracking without outliers, and with or without missing data, e.g, (Yang, 1995; Chi et al., 2013; Zhang & Balzano, 2016) is in this category.

On the other hand, the most general (nonstationary) model that allows the subspace to change at each time is not even identifiable since at least r data points are needed to compute an r-dimensional subspace even in the ideal setting of no noise or missing entries. When the data is noisy, missing or outlier corrupted, even more data points are needed. Since only one data vector comes in at each time t, the only way to ensure this is to assume that the subspace remains

[1]Department of Electrical and Computer Engineering, Iowa State University, USA. Correspondence to: Namrata Vaswani <namrata@iastate.edu>.

Proceedings of the 35th International Conference on Machine Learning, Stockholm, Sweden, PMLR 80, 2018. Copyright 2018 by the author(s).

[1]Needs assumptions on intermediate algorithm estimates.

constant for a while and then changes (piecewise constant). We first introduced this assumption in (Qiu et al., 2014). Here, we use a weaker version of the same model.

Robust Subspace Tracking (RST) and Dynamic RPCA Problem. At each time t, we get a $\boldsymbol{y}_t \in \mathbb{R}^n$ that satisfies

$$\boldsymbol{y}_t := \boldsymbol{\ell}_t + \boldsymbol{x}_t + \boldsymbol{v}_t, \text{ for } t = 1, 2, \ldots, d.$$

where $\boldsymbol{v}_t$ is small unstructured noise, $\boldsymbol{x}_t$ is the sparse outlier vector, and $\boldsymbol{\ell}_t$ is the true data vector that lies in a fixed or slowly changing low-dimensional subspace of $\mathbb{R}^n$, i.e., $\boldsymbol{\ell}_t = \boldsymbol{P}_{(t)}\boldsymbol{a}_t$ where $\boldsymbol{P}_{(t)}$ is an $n \times r$ *basis matrix* (matrix with mutually orthonormal columns) with $r \ll n$ and with $\|(\boldsymbol{I} - \boldsymbol{P}_{(t-1)}\boldsymbol{P}_{(t-1)}{}')\boldsymbol{P}_{(t)}\|$ small compared to $\|\boldsymbol{P}_{(t)}\| = 1$. Here, and elsewhere, $\|.\|$ denotes the induced l_2 norm and $'$ denotes transpose. We use $\mathcal{T}_t$ to denote the support set of $\boldsymbol{x}_t$. Given an initial subspace estimate, $\hat{\boldsymbol{P}}_0$, the goal is to track $\operatorname{span}(\boldsymbol{P}_{(t)})$ and $\boldsymbol{\ell}_t$ either immediately or within a short delay. A by-product is that $\boldsymbol{\ell}_t$, $\boldsymbol{x}_t$, and $\mathcal{T}_t$ can also be tracked on-the-fly. The initial subspace estimate, $\hat{\boldsymbol{P}}_0$, can be computed by applying any of the solutions for static RPCA, e.g., PCP or AltProj, for the first roughly r data points, $\boldsymbol{Y}_{[1,t_{\text{train}}]}$ with $t_{\text{train}} = Cr$. 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 its columns indexed by entries in the set $\mathcal{T}$.

Dynamic RPCA is the offline version of the above problem. Define matrices $\boldsymbol{L}, \boldsymbol{X}, \boldsymbol{V}, \boldsymbol{Y}$ with $\boldsymbol{L} = [\boldsymbol{\ell}_1, \boldsymbol{\ell}_2, \ldots \boldsymbol{\ell}_d]$ and $\boldsymbol{Y}, \boldsymbol{X}, \boldsymbol{V}$ similarly defined. The goal is to recover the matrix $\boldsymbol{L}$ and its column space with ϵ error. We use $r_{\scriptscriptstyle L}$ to denote the rank of $\boldsymbol{L}$. The maximum fraction of nonzeros in any row (column) of the outlier matrix $\boldsymbol{X}$ is denoted by max-outlier-frac-row (max-outlier-frac-col).

For basis matrices $\boldsymbol{P}_1, \boldsymbol{P}_2$, we use $\operatorname{SE}(\boldsymbol{P}_1, \boldsymbol{P}_2) := \|(\boldsymbol{I} - \boldsymbol{P}_1\boldsymbol{P}_1{}')\boldsymbol{P}_2\|$ as a measure of Subspace Error (distance) between their respective column spans. This is the sine of the largest principal angle between the subspaces.
We reuse the letters C, c to denote different numerical constants in each use with $C \geq 1$ and $c < 1$.

Identifiability. The above problem definition does not ensure identifiability since either of $\boldsymbol{L}$ or $\boldsymbol{X}$ can be both low-rank and sparse. Also, if the subspace changes at every time, it is impossible to correctly estimate all the subspaces.

- One way to ensure identifiability of changing subspaces is to assume that they are piecewise constant, i.e.,

  $$\boldsymbol{P}_{(t)} = \boldsymbol{P}_j \text{ for all } t \in [t_j, t_{j+1}),\ j = 1, 2, \ldots, J.$$

  and to lower bound $d_j := t_{j+1} - t_j$. Let $t_0 = 1$ and $t_{J+1} = d$. With this model, $r_{\scriptscriptstyle L} = rJ$ in general (except if subspace directions are repeated).

- One can ensure that $\boldsymbol{X}$ is not low-rank by imposing upper bounds on max-outlier-frac-col and max-outlier-frac-row. For the RST problem, max-outlier-frac-col $:= \max_t |\mathcal{T}_t|/n$. Consider max-outlier-frac-row. Since RST is a tracking problem, it involves processing incoming data either one sample at a time or using a mini-batch of α samples at a time. The subspace update step of our proposed algorithm does the latter. Because of this we need to replace a bound on max-outlier-frac-row by a bound on max-outlier-frac-row$^\alpha$ defined as follows.

  Definition 1.1. *Define max-outlier-frac-row$^\alpha$ as the maximum fraction of outliers (nonzeros) per row of any sub-matrix of $\boldsymbol{X}$ with α consecutive columns. For a time interval, $\mathcal{J}$, let $\gamma(\mathcal{J}) := \max_{i=1,2,\ldots,n} \frac{1}{|\mathcal{J}|}\sum_{t\in\mathcal{J}} \mathbf{1}_{\{i\in\mathcal{T}_t\}}$ where $\mathbf{1}_S$ is the indicator function for statement S. Thus $\gamma(\mathcal{J})$ is the maximum outlier fraction in any row of the sub-matrix $\boldsymbol{X}_{\mathcal{J}}$ of $\boldsymbol{X}$. Let $\mathcal{J}^\alpha$ denote a time interval of duration α. Then max-outlier-frac-row$^\alpha := \max_{\mathcal{J}^\alpha\subseteq[1,d]} \gamma(\mathcal{J}^\alpha)$.*

- One way to ensure that $\boldsymbol{L}$ is not sparse is by requiring that its left and right singular vectors are dense (non-sparse) or "incoherent" w.r.t. a sparse vector. This is quantified as follows (Candès et al., 2011).

  Definition 1.2. *An $n \times r_P$ basis matrix $\boldsymbol{P}$ is μ-incoherent if $\max_i \|\boldsymbol{P}^{(i)}\|^2 \leq \mu\frac{r_P}{n}$ ($\boldsymbol{P}^{(i)}$ is i-th row of $\boldsymbol{P}$).*

  Using our subspace model, the union of the column spans of all the $\boldsymbol{P}_j$'s is equal to the span of the left singular vectors of $\boldsymbol{L}$. Thus, for RST, left incoherence is equivalent to assuming that the $\boldsymbol{P}_j$'s are μ-incoherent. We replace right incoherence by statistical assumptions on the $\boldsymbol{a}_t$'s: assume that $\boldsymbol{a}_t$'s are element-wise bounded, mutually independent, have identical covariances and zero mean. Refer Remark 3.5 of (Narayanamurthy & Vaswani, 2018b) to see the connection.

Contributions. (1) We develop a novel algorithm for RST and dynamic RPCA that we call "Nearly Optimal RST" or NORST for short. NORST relies on the previously introduced recursive projected compressive sensing solution framework (Qiu et al., 2014), but has a significantly improved, and simpler, subspace update step. Our most important contribution is the first provable guarantee for RST that ensures near optimal tracking delay and needs a near optimal lower bound on how long a subspace needs to remain constant. Both equal $Cr \log n \log(1/\epsilon)$. It also needs no other model on how the subspace changes, *and* still tolerates $O(1)$ maximum fraction of outliers in any row and an $O(1/r)$ fraction in any column. Of course it uses extra assumptions: slow subspace change and lower bound on most

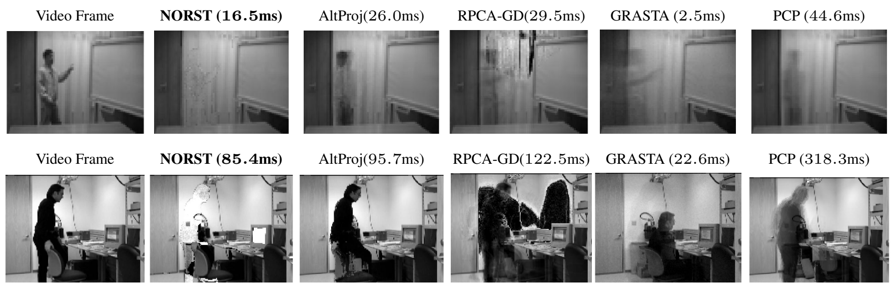

Figure 1. Background Recovery. NORST is the only technique that works for both sequences. Row 2: only NORST does not contain the person or even his shadow. NORST is faster than all except GRASTA (which does not work). The GRASTA output slightly lags the actual frame, and hence, it appears that the person is sitting. Time taken per frame is shown in parentheses.

outlier magnitudes. Finally, NORST is provably online (after initialization), fast (has the same complexity as vanilla r-SVD), and nearly memory optimal, it has memory complexity of order $nr\log n \log(1/\epsilon)$. Here the term "online" means the following: after each subspace change, the algorithm updates the subspace estimate every $\alpha = Cr\log n$ time instants, and the subspace recovery error bound decays exponentially with each such step[2]. This means one gets an ϵ-accurate estimate within $C\log(1/\epsilon)$ steps.

(2) Our guarantees hold under weakened standard RPCA assumptions, slow subspace change, and a lower bound on (most) outlier magnitudes. We say "weakened" because we show that that, after initialization, NORST can tolerate a constant maximum fraction of outliers per row without needing any assumptions on outlier support. For the video application, this implies that it tolerates slow moving and occasionally static foreground objects much better than all other existing RPCA approaches. This fact is also corroborated on real videos, e.g., see Fig 1. *All* other existing RPCA approaches (except simple-ReProCS) need more: AltProj, RPCA-GD, and NO-RMC need this fraction to be $O(1/r_L)$; PCP and mod-PCP need the outlier support to uniformly random (strong requirement: for video it implies that objects are very small sized and jumping around randomly); and original-ReProCS needs it to satisfy a specific moving object model described later (very restrictive). Moreover, NORST can also detect a subspace change within a short delay of $Cr\log n$.

(4) Unlike previous dynamic RPCA work, NORST only needs a coarse subspace initialization which can be computed using at most $C\log r$ iterations of any batch RPCA method such as AltProj applied to Cr initial samples. In fact, if the outlier magnitudes were very large (or if the outliers were absent) for an initial set of $O(r\log n\log r)$ time instants, even a random initialization would suffice.

[2]The reason that just $O(r\log n)$ samples suffice for each update is because we assume that the $\boldsymbol{a}_t$'s are bounded, $\boldsymbol{v}_t$ is very small and with effective dimension r or smaller (see Theorem 2.1), and both are mutually independent over time. These along with the specific structure of the PCA problem we encounter (noise/error seen by the PCA step depends on the $\boldsymbol{\ell}_t$'s and thus has "effective dimension" r) is why so few samples suffice.

(3) A direct corollary of our result is a guarantee that a minor modification of NORST also solves the subspace tracking with missing data (ST-missing) and the dynamic matrix completion problems (dynamic MC); see follow-up work (**?**). All existing guarantees for ST-missing hold only for the case of a *single unknown subspace* and are *partial guarantees*. Ours is a complete guarantee that provably allows tracking of changing subspaces. From the MC perspective, our solution does not assume *any* model on the observed entries unlike most others. The disadvantage is that it needs more observed entries than other MC solutions.

2. The Algorithm and Main Result

Nearly-Optimal RST via ReProCS (ReProCS-NORST). ReProCS-NORST starts with a "good" estimate of the initial subspace, which is obtained by $C\log r$ iterations of AltProj on $\boldsymbol{Y}_{[1,t_{\text{train}}]}$ with $t_{\text{train}} = Cr$. It then iterates between (a) *Projected Compressive Sensing (CS) or Robust Regression* in order to estimate the sparse outliers, $\boldsymbol{x}_t$'s, and hence the $\boldsymbol{\ell}_t$'s, and (b) *Subspace Update* to update the subspace estimate $\hat{\boldsymbol{P}}_{(t)}$. Projected CS is borrowed from original-ReProCS (Qiu et al., 2014), while the subspace update step is new and significantly simplified. Projected CS proceeds as follows. At time t, if the previous subspace estimate, $\hat{\boldsymbol{P}}_{(t-1)}$, is accurate enough, because of slow subspace change, projecting $\boldsymbol{y}_t$ 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, $\tilde{\boldsymbol{y}}_t = \boldsymbol{\Psi}\boldsymbol{x}_t + \boldsymbol{\Psi}(\boldsymbol{\ell}_t + \boldsymbol{v}_t)$ and $\|\boldsymbol{\Psi}(\boldsymbol{\ell}_t + \boldsymbol{v}_t)\|$ is small due to slow subspace change and small $\boldsymbol{v}_t$. Recovering $\boldsymbol{x}_t$ from $\tilde{\boldsymbol{y}}_t$ is now a CS / sparse recovery problem in small noise (Candes, 2008). We compute $\hat{\boldsymbol{x}}_{t,cs}$ using noisy l_1 minimization followed by thresholding based support estimation to obtain $\hat{\mathcal{T}}_t$. 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 for the Subspace Update step which involves (i) detecting subspace change, and (ii) obtaining improved estimates of the new subspace by K steps of r-SVD, each done with a new set of α samples of $\hat{\boldsymbol{\ell}}_t$. While this step is designed under the piecewise constant subspace assumption (needed for identifiability), if the goal is only to get good estimates of $\boldsymbol{\ell}_t$ or $\boldsymbol{x}_t$, the method works even for real videos (where this assumption may or may not hold). For ease of understanding, we present a basic version of NORST in Algorithm 1. This assumes the change times t_j are known. The actual algorithm that we study and implement detects these automatically is given as Algorithm 2 in the long version (Narayanamurthy & Vaswani, 2018b).

Main Result. We use $\hat{t}_j$ to denote the time-instant at which the j-th subspace change time is detected.

Theorem 2.1. *Consider NORST (Algorithm 2 of (Narayanamurthy & Vaswani, 2018b))*[3]. *Let* $\boldsymbol{\Lambda} := \mathbb{E}[\boldsymbol{a}_1\boldsymbol{a}_1{}']$, $\lambda^+ := \lambda_{\max}(\boldsymbol{\Lambda})$, $\lambda^- := \lambda_{\min}(\boldsymbol{\Lambda})$, $f := \lambda^+/\lambda^-$, $\alpha := Cf^2 r\log n$, *and let* $x_{\min} := \min_t \min_{i\in\mathcal{T}_t}(\boldsymbol{x}_t)_i$ *denote the minimum outlier magnitude. Pick an* $\varepsilon \le \min(0.01, 0.03\min_j \mathrm{SE}(\boldsymbol{P}_{j-1},\boldsymbol{P}_j)^2/f)$. *Let* $K := C\log(1/\varepsilon)$. *If*

1. *incoherence:* $\boldsymbol{P}_j$*'s are* μ*-incoherent; and* $\boldsymbol{a}_t$*'s are zero mean, mutually independent over time* t*, have identical covariance matrices, i.e.* $\mathbb{E}[\boldsymbol{a}_t\boldsymbol{a}_t{}'] = \boldsymbol{\Lambda}$*, are element-wise uncorrelated (*$\boldsymbol{\Lambda}$ *is diagonal), are element-wise bounded (for a numerical constant* η*,* $(\boldsymbol{a}_t)_i^2 \le \eta\lambda_i(\boldsymbol{\Lambda})$*), and are independent of all outlier supports* $\mathcal{T}_t$*;*
2. $\boldsymbol{v}_t$*:* $\|\boldsymbol{v}_t\|^2 \le cr\|\mathbb{E}[\boldsymbol{v}_t\boldsymbol{v}_t{}']\|$, $\|\mathbb{E}[\boldsymbol{v}_t\boldsymbol{v}_t{}']\| \le c\varepsilon^2\lambda^-$*, zero mean, mutually independent, independent of* $\boldsymbol{x}_t, \boldsymbol{\ell}_t$*;*
3. *max-outlier-frac-col* $\le \frac{c_1}{\mu r}$*, max-outlier-frac-row*$^\alpha \le \frac{c_2}{f^2}$*;*
4. *subspace change: let* $\Delta := \max_j \mathrm{SE}(\boldsymbol{P}_{j-1},\boldsymbol{P}_j)$*,*
   (a) $t_{j+1} - t_j > (K+2)\alpha$*,*
   (b) $\Delta \le 0.8$ *and* $C_1\sqrt{r\lambda^+}(\Delta + 2\varepsilon) \le x_{\min}$*;*
5. *init*[4]*:* $\mathrm{SE}(\hat{\boldsymbol{P}}_0,\boldsymbol{P}_0) \le 0.25$, $C_1\sqrt{r\lambda^+}\mathrm{SE}(\hat{\boldsymbol{P}}_0,\boldsymbol{P}_0) \le x_{\min}$*;*

and (6) algorithm parameters are appropriately set; then, with probability (w.p.) $\ge 1 - 10dn^{-10}$*, at all times, t,*

- $\hat{\mathcal{T}}_t = \mathcal{T}_t$, $t_j \le \hat{t}_j \le t_j + 2\alpha$, $\mathrm{SE}(\hat{\boldsymbol{P}}_{(t)},\boldsymbol{P}_{(t)}) \le$

$$\begin{cases} (\varepsilon+\Delta) & \text{if } t \in [t_j, \hat{t}_j+\alpha), \\ (0.3)^{k-1}(\varepsilon+\Delta) & \text{if } t \in [\hat{t}_j+(k-1)\alpha, \hat{t}_j+k\alpha), \\ \varepsilon & \text{if } t \in [\hat{t}_j+K\alpha+\alpha, t_{j+1}), \end{cases}$$

  and $\|\hat{\boldsymbol{x}}_t - \boldsymbol{x}_t\| = \|\hat{\boldsymbol{\ell}}_t - \boldsymbol{\ell}_t\| \le 1.2(\mathrm{SE}(\hat{\boldsymbol{P}}_{(t)},\boldsymbol{P}_{(t)})+\varepsilon)\|\boldsymbol{\ell}_t\|$.
- *Offline-NORST (lines 26-30):* $\mathrm{SE}(\hat{\boldsymbol{P}}_{(t)}^{off},\boldsymbol{P}_{(t)}) \le \varepsilon$, $\|\hat{\boldsymbol{x}}_t^{off} - \boldsymbol{x}_t\| = \|\hat{\boldsymbol{\ell}}_t^{off} - \boldsymbol{\ell}_t\| \le \varepsilon\|\boldsymbol{\ell}_t\|$ *at all t.*
- *Memory complexity is* $O(nr\log n\log(1/\varepsilon))$ *and time complexity is* $O(ndr\log(1/\varepsilon))$*.*

[3]Algorithm 1 with a subspace change detection step included

[4]This can be satisfied by applying AltProj (Netrapalli et al., 2014) on first Cr data samples and assuming that these have outlier fractions in any row or column bounded by c/r.

Algorithm 1 Basic-NORST (with t_j known). The actual algorithm that detects t_j automatically is Algo. 3 in (Narayanamurthy & Vaswani, 2018b).
Notation: $\hat{\boldsymbol{L}}_{t;\alpha} := [\hat{\boldsymbol{\ell}}_{t-\alpha+1}, \cdots, \hat{\boldsymbol{\ell}}_t]$ and $SVD_r[\boldsymbol{M}]$ refers to the top of r left singular vectors of the matrix $\boldsymbol{M}$.
Obtain $\hat{\boldsymbol{P}}_0$ by $C(\log r)$ iterations of AltProj on $\boldsymbol{Y}_{[1,t_{\text{train}}]}$ with $t_{\text{train}} = Cr$ followed by SVD on the output $\hat{\boldsymbol{L}}$.

```
1:  Input: $\boldsymbol{y}_t$, Output: $\hat{\boldsymbol{x}}_t$, $\hat{\boldsymbol{\ell}}_t$, $\hat{\boldsymbol{P}}_{(t)}$, $\hat{\mathcal{T}}_t$
2:  Parameters: $K \leftarrow C\log(1/\varepsilon)$, $\alpha \leftarrow Cf^2 r\log n$, $\omega_{supp} \leftarrow x_{\min}/2$, $\xi \leftarrow x_{\min}/15$, $r$
3:  Initialize: $j \leftarrow 1, k \leftarrow 1$ $\hat{\boldsymbol{P}}_{t_{\text{train}}} \leftarrow \hat{\boldsymbol{P}}_0$
4:  for $t > t_{\text{train}}$ do
5:      $\boldsymbol{\Psi} \leftarrow \boldsymbol{I} - \hat{\boldsymbol{P}}_{(t-1)}\hat{\boldsymbol{P}}_{(t-1)}{}'$;
6:      $\tilde{\boldsymbol{y}}_t \leftarrow \boldsymbol{\Psi}\boldsymbol{y}_t$.
7:      $\hat{\boldsymbol{x}}_{t,cs} \leftarrow \arg\min_{\tilde{\boldsymbol{x}}} \|\tilde{\boldsymbol{x}}\|_1$ s.t. $\|\tilde{\boldsymbol{y}}_t - \boldsymbol{\Psi}\tilde{\boldsymbol{x}}\| \le \xi$.
8:      $\hat{\mathcal{T}}_t \leftarrow \{i : |\hat{\boldsymbol{x}}_{t,cs}| > \omega_{supp}\}$.
9:      $\hat{\boldsymbol{x}}_t \leftarrow \boldsymbol{I}_{\hat{\mathcal{T}}_t}(\boldsymbol{\Psi}_{\hat{\mathcal{T}}_t}{}'\boldsymbol{\Psi}_{\hat{\mathcal{T}}_t})^{-1}\boldsymbol{\Psi}_{\hat{\mathcal{T}}_t}{}'\tilde{\boldsymbol{y}}_t$.
10:     $\hat{\boldsymbol{\ell}}_t \leftarrow \boldsymbol{y}_t - \hat{\boldsymbol{x}}_t$.
11:     if $t = t_j + u\alpha$ for $u = 1, 2, \cdots, K$ then
12:         $\hat{\boldsymbol{P}}_{j,k} \leftarrow SVD_r[\hat{\boldsymbol{L}}_{t;\alpha}]$, $\hat{\boldsymbol{P}}_{(t)} \leftarrow \hat{\boldsymbol{P}}_{j,k}$, $k \leftarrow k+1$.
13:     else
14:         $\hat{\boldsymbol{P}}_{(t)} \leftarrow \hat{\boldsymbol{P}}_{(t-1)}$.
15:     end if
16:     if $t = t_j + K\alpha$ then
17:         $\hat{\boldsymbol{P}}_j \leftarrow \hat{\boldsymbol{P}}_{(t)}$, $k \leftarrow 1$, $j \leftarrow j+1$
18:     end if
19: end for
```

Remark 2.2 (Bi-level outliers)**.** *With minor changes, we can actually prove the following which relaxes our outlier magnitudes lower bounded requirement to only requiring that most outlier magnitudes are lower bounded, while the others have small enough magnitudes so that their squared sum is upper bounded. Both bounds decrease as the subspace estimate improves.*
Assume that the outlier magnitudes are such that the following holds: $\boldsymbol{x}_t$ *can be split as* $\boldsymbol{x}_t = (\boldsymbol{x}_t)_{small} + (\boldsymbol{x}_t)_{large}$ *with the two components having disjoint supports and being such that,* $\|(\boldsymbol{x}_t)_{small}\| \le b_{v,t}$ *and the smallest nonzero entry of* $(\boldsymbol{x}_t)_{large}$ *is greater than* $30b_{v,t}$ *with* $b_{v,t}$ *defined as follows:* $b_{v,t} = C(2\varepsilon+\Delta)\sqrt{r\lambda^+}$ *for* $t \in [t_j, \hat{t}_j+\alpha)$ *(before the first subspace update),* $b_{v,t} := 0.3^{k-1}C(2\varepsilon+\Delta)\sqrt{r\lambda^+}$ *for* $t \in [\hat{t}_j+(k-1)\alpha, \hat{t}_j+k\alpha-1]$, $k = 2, 2, \ldots, K$ *(after the k-th subspace update), and* $b_{v,t} := C\varepsilon\sqrt{r\lambda^+}$ *for* $t \in \hat{t}_j + K\alpha, t_{j+1})$*.*
If the above is true, and if the vectors $(\boldsymbol{x}_t)_{small}$ *are zero mean, mutually independent, and independent of* $\boldsymbol{\ell}_t$*'s and of the support of* $(\boldsymbol{x}_t)_{large}$*, then all conclusions of Theorem 2.1 hold except the exact support recovery conclusion (this gets replaced by exact recovery of the support of* $(\boldsymbol{x}_t)_{large}$*).*

Discussion. Theorem 2.1 shows that, with high probability (whp), when using NORST, the subspace change gets detected within a delay of at most $2\alpha = Cf^2(r\log n)$ time instants, and the subspace gets estimated to ε error within at most $(K+2)\alpha = Cf^2(r\log n)\log(1/\varepsilon)$ time instants. The same is also true for the recovery error of $\boldsymbol{x}_t$ and $\boldsymbol{\ell}_t$. If offline processing is allowed, with a delay of at most $Cf^2(r\log n)\log(1/\varepsilon)$ samples, we can guarantee all recoveries within normalized error ε.

Theorem 2.1 *allows a constant maximum fraction of outliers per row (after initialization), without making any assumption on how the outliers are generated.* As noted earlier, this is better than what all other RPCA solutions allow. The same is true for the NORST memory complexity which is almost d/r times better. The time complexity is worse than that of only NO-RMC, but NO-RMC needs $d \geq cn$ (unreasonable requirement for videos which often have much fewer frames d than the image size n). NO-RMC is so fast because it is actually a robust matrix completion solution and it deliberately undersamples the entire data matrix $\boldsymbol{Y}$ to get a faster RPCA algorithm. But this also means that it cannot recover $\boldsymbol{X}$. Finally, NORST also needs the best outlier fraction per column bound of $O(1/r)$ instead of $O(1/r_L)$. Notice that if J is large, e.g. if $J = d/(r\log n)$, it is possible that $r_L \gg r$.

We should clarify that NORST allows the maximum fraction of outliers per row to be $O(1)$ but this does not necessarily imply that the number of outliers in each row can be this high. The reason is it only allows the fraction per column to only be $O(1/r)$. Thus, for a matrix of size $n \times \alpha$, it allows the total number of outliers to be $O(\min(n\alpha, n\alpha/r)) = O(n\alpha/r)$. Thus the average fraction allowed is only $O(1/r)$.

NORST has the above advantages only if a few extra assumptions hold. The first is element-wise boundedness of the $\boldsymbol{a}_t$'s. This, along with mutual independence and identical covariances, of $\boldsymbol{a}_t$'s is similar to the right incoherence assumption needed by all static RPCA methods, see Remark 3.5 in longer version. The zero-mean assumption on $\boldsymbol{a}_t$'s is a minor one. The assumption that $\boldsymbol{\Lambda}$ be diagonal is also minor[5]. The main extra requirement is that $x_{\min}$ be lower bounded as given in the last two assumptions of Theorem 2.1, or that *most outliers are lower bounded (more relaxed requirement)* as given in the long version. The required lower bound is reasonable as long as the initial subspace estimate is accurate enough and the subspace changes slowly enough so that both Δ and $\mathrm{SE}(\hat{\boldsymbol{P}}_0, \boldsymbol{P}_0)$ are $O(1/\sqrt{r})$. This requirement may seem restrictive on first glance but actually is not. The reason is that SE(.) is only measuring the largest principal angle. This bound on SE still allows the chordal distance[6] between the two subspaces to be $O(1)$. This also matches what s-ReProCS requires.

[5]It only implies that $\boldsymbol{P}_j$ is the matrix of principal components of $\mathbb{E}[\boldsymbol{L}_j\boldsymbol{L}_j']$ where $\boldsymbol{L}_j := [\boldsymbol{\ell}_{t_j}, \boldsymbol{\ell}_{t_j+1}, \ldots, \boldsymbol{\ell}_{t_{j+1}-1}]$. If $\boldsymbol{\Lambda}$ is not diagonal, it is easy to right multiply $\boldsymbol{P}_j$ by an orthonormal matrix $\boldsymbol{R}$ that is such that $\boldsymbol{R}'\boldsymbol{\Lambda}\boldsymbol{R}$ is diagonal and to use $\boldsymbol{\ell}_t = (\boldsymbol{P}_j\boldsymbol{R})\tilde{\boldsymbol{a}}_t$ with $\tilde{\boldsymbol{a}}_t := \boldsymbol{R}'\boldsymbol{a}_t$. $(\boldsymbol{P}_j\boldsymbol{R})$ has the same incoherence as $\boldsymbol{P}_j$.

Why NORST is better than RPCA-every-α. RPCA-every-α refers to using any batch RPCA solution on $\alpha = Cr\log n$ samples at a time. (a) NORST is significantly better than RPCA-every-α for settings in which the number of outliers in some rows is very large: our guarantee shows that NORST can tolerate a constant maximum fraction of outliers per row because it exploits "slow subspace change" and one other mild assumption (lower bound on most outlier magnitudes). On the other hand, all RPCA methods require this fraction to be only $O(1/r)$, which is much smaller. (b) Moreover, NORST provides guarantees for subspace recovery (and recovery of $\boldsymbol{\ell}_t$, $\boldsymbol{x}_t$ and $\mathcal{T}_t$) at each time t, including during the "dwell time" (the first $\alpha = Cr\log n$ time instants after a subspace change). This is possible because NORST assumes slow subspace change and the algorithm is recursive (uses the previous subspace estimate at the current time). As shown in Theorem 2.1, during the dwell time, the subspace recovery error is essentially bounded by the amount of subspace change. However, RPCA-every-α will not provide any estimates during the dwell time; it will require a delay of this much time before it provides any estimates. (c) NORST can detect subspace changes automatically and quickly (with just one projection-SVD step), while RPCA-every-α will take much more computation time to do this. Finally, in practical comparisons for videos involving slow moving objects (large number of outliers in some rows), NORST is both significantly better than, and faster than, all RPCA methods. This is true both when RPCA methods are applied to the entire dataset, as well as when they are applied to α pieces of the dataset. We do not report these results due to lack of space and since applying RPCA to the whole matrix resulted in better performance out of the two options.

Outlier v/s Subspace Assumptions. When there are fewer outliers in the data or when outliers are easy to detect, one would expect to need weaker assumptions on the true data's subspace and/or on its rate of change. This is indeed true. The max-outlier-frac-col bound relates max-outlier-frac-col to μ (not-denseness parameter) and r (subspace dimension). The upper bound on Δ implies that, if $x_{\min}$ is larger (outliers are easier to detect), a larger amount of subspace change Δ can be tolerated. The relation of max-outlier-frac-row to rate of subspace change is not evident from the way the guarantee is stated above because we have assumed max-outlier-frac-row $\leq b_0 := c/f^2$ with c being a numerical constant, and used this to get a sim-

[6]l_2 norm of the vector containing the sine of all principal angles.

Table 1. Comparing all RPCA or RST solutions. All algorithms also require left and right incoherence or left incoherence and $\boldsymbol{a}_t$'s bounded (which is similar to right incoherence), and hence these are not compared. The incoherence parameter μ and the condition numbers are treated as constants. In general $r_L = rJ$. **Strong or unrealistic assumptions are shown in red.**

Algorithm	Outlier tolerance	Other Assumptions	Memory, Time,	# params
PCP(C) mod-PCP	max-outlier-frac-row $\in O(1)$ max-outlier-frac-col $\in O(1)$	**outlier support: unif. random,** $r_L \leq c \min(n,d)/\log^2 n$	Memory: $O(nd)$ Time: $O(nd^2 \frac{1}{\epsilon})$	zero
AltProj	max-outlier-frac-row $\in O\left(1/r_L\right)$ max-outlier-frac-col $\in O\left(1/r_L\right)$	$d \geq cr_L$	Memory: $O(nd)$ Time: $O(ndr_L^2 \log \frac{1}{\epsilon})$	2
RPCA-GD	max-outlier-frac-row $\in O(1/r_L^{1.5})$ max-outlier-frac-col $\in O(1/r_L^{1.5})$	$d \geq cr_L$	Memory: $O(nd)$ Time: $O(ndr_L \log \frac{1}{\epsilon})$	5
NO-RMC	max-outlier-frac-row $\in O\left(1/r_L\right)$ max-outlier-frac-col $\in O(1/r_L)$	$\mathbf{c_2 n \geq d \geq cn}$	Memory: $O(nd)$ Time: $O(nr_L^3 \log^2 n \log^2 \frac{1}{\epsilon})$	4
orig-ReProCS dynamic RPCA, tracking delay too large	max-outlier-frac-row $\in O(1)$ max-outlier-frac-col $\in O(1/r_L)$	**outlier support: moving object model,** **unrealistic subspace change model,** **changed eigenvalues small for some time,** outlier mag. lower bounded, init data: AltProj assu's, $\boldsymbol{a}_t$'s independent, $d \geq cr^2/\epsilon^2$	Memory: $O(nr^2/\epsilon^2)$ Time: $O(ndr \log \frac{1}{\epsilon})$	5
s-ReProCS: solves d-RPCA & RST w/ sub-optimal delay	max-outlier-frac-row $\in O(1)$ max-outlier-frac-col $\in O(1/r)$	**subspace change: only 1 direc at a time**, outlier mag. lower bounded, $\boldsymbol{a}_t$'s independent, $d \geq cr \log n \log \frac{1}{\epsilon}$. init data: AltProj assumptions	Memory: $O(nr \log n \log \frac{1}{\epsilon})$ Time: $O(ndr \log \frac{1}{\epsilon})$	4
NORST **(this work):** solves **d-RPCA & RST w/** **near-optimal delay**	max-outlier-frac-row $\in O(1)$ max-outlier-frac-col $\in O(1/r)$	subspace change: none, outlier mag. lower bounded, $\boldsymbol{a}_t$'s independent, $d \geq cr \log n \log \frac{1}{\epsilon}$, first Cr samples: AltProj assumptions	Memory: $O(nr \log n \log \frac{1}{\epsilon})$ Time: $O(ndr \log \frac{1}{\epsilon})$	4

ple expression for K. If we did not do this, we would get $K = C\lceil \frac{1}{-\log(\sqrt{b_0} f)} \log(\frac{c\Delta}{0.8\varepsilon})\rceil$, see Remark A.1 of long version. Since we need $t_{j+1} - t_j \geq (K+2)\alpha$, a smaller b_0 means a larger Δ can be tolerated for the same delay, or vice versa.

Algorithm Parameters. Algorithm 1 (and Algorithm 2 of (Narayanamurthy & Vaswani, 2018b)) assumes knowledge of 4 model parameters: r, λ^+, λ^- and $x_{\min}$ to set the algorithm parameters. The initial dataset used for estimating $\hat{\boldsymbol{P}}_0$ (using AltProj) can be used to get an accurate estimate of r, λ^- and λ^+ using standard techniques. Thus one really only needs to set $x_{\min}$. If continuity over time is assumed, we can let it be time-varying and set it as $\min_{i \in \hat{\mathcal{T}}_{t-1}} |(\hat{\boldsymbol{x}}_{t-1})_i|$ at t.

Related Work. For a summary of comparisons, see Table 1. The earliest dynamic RPCA result was a partial guarantee (a guarantee that depended on intermediate algorithm estimates satisfying certain assumptions) for the original ReProCS approach (original-ReProCS) (Qiu et al., 2014). This was followed up by two complete guarantees for ReProCS-based approaches with minor modifications (Lois & Vaswani, 2015; Zhan et al., 2016). For simplicity we will still call these "original-ReProCS". Parallel work also included a guarantee for modified-PCP which is a solution for RPCA with partial subspace knowledge (Zhan & Vaswani, 2015). This provides a piecewise batch solution to dynamic RPCA. More recent work developed and analyzed simple-ReProCS (s-ReProCS) which removed most of the disadvantages of previous works.

S-ReProCS has the same tracking delay and memory complexity as NORST and needs the same outlier assumptions; however, it assumes that only one subspace direction can change at each change time. This is a much more restrictive model than what NORST needs (allows all r directions to change) *and* it implies that the tracking delay of s-ReProCS is r-times sub-optimal. Moreover, s-ReProCS required a complicated projection-SVD step for subspace update (as opposed to SVD in NORST). These two facts imply that s-ReProCS needs an ϵ-accurate subspace initialization in order to ensure that the later changed subspaces can be tracked with ϵ-accuracy; *and* it does not provide a useful solution for ST-missing or dynamic MC. The advantage of s-ReProCS over NORST is that it is a little faster (needs 1-SVD instead of r-SVD most of the time), and its required bound on outlier fractions and delay between subspace change times have a weaker dependence on the condition number of the true data covariance.

The original-ReProCS guarantees needed very strong assumptions and their tracking delay was $O(nr^2/\epsilon^2)$. Since ϵ can be very small, this factor can be quite large, and hence one cannot claim that original-ReProCS solves RST. Our work is a very significant improvement over all these works. (i) The guaranteed memory complexity, tracking delay, and required delay between subspace change times of NORST are all r/ϵ^2 times lower than that of original-ReProCS. (ii) The original-ReProCS guarantees needed a very specific assumption on how the outlier support could change: they required an outlier support model inspired by a video moving object that moves in one direction for a long time; and whenever it moves, it must move by a fraction of $s := \max_t |\mathcal{T}_t|$. This is very specific model with the requirement of moving by a fraction of s being the most restrictive. Our result replaces this with just a bound on max-outlier-frac-row. We explain in Sec. 3 why this is possible. (ii) Their subspace change model required one or more new directions orthogonal to $\boldsymbol{P}_{j-1}$ to be added at each t_j. This is an unrealistic model for slow subspace change, e.g., in 3D, it implies that the subspace needs to change from the x-y plane to the y-z plane. Moreover because of this model, their results needed the "energy" (eigenvalues) along the newly added directions to be small for a period of time after each subspace change. This is a strong (and hard to interpret) requirement. We replace all these requirements with a bound on $\mathrm{SE}(\boldsymbol{P}_{j-1}, \boldsymbol{P}_j)$ which is much more realistic. Thus, in 3D, we allow the x-y plane to change to its slightly tilted version.

Since the modified-PCP guarantee adapted the PCP proof techniques from (Candès et al., 2011), its pros and cons are similar to those of PCP, e.g., it also needs a uniformly randomly generated outlier support, and it also cannot detect subspace change. Also see Table 1.

We also provide a comparison with provably correct standard RPCA approaches in Table 1. In summary, NORST has significantly better memory complexity than all of them, all of which are batch; it has the best outlier tolerance (after initialization), and the second-best time complexity, as long as its extra assumptions hold. It can also detect subspace change quickly, which can be a useful feature.

3. Proof Outline with $\boldsymbol{v}_t = 0$

Remark 3.1. *When stating Theorem 2.1, we just used numerical constants c_1, c_2, C_1 for simplicity. The result holds with $c_1 = 0.01$, $c_2 = 0.01$, and $C_1 = 15\sqrt{\eta}$.*

For simplicity, we outline the proof for the $\boldsymbol{v}_t = 0$ case. The changes with $\boldsymbol{v}_t \neq 0$ are minor, see (Narayanamurthy & Vaswani, 2018b).

First consider the simpler case when t_j's are known, i.e., consider Algorithm 1. In this case, $\hat{t}_j = t_j$. Define

1. bound on max-outlier-frac-row: $b_0 := 0.01/f^2$.
2. $q_0 := 1.2(\varepsilon + \mathrm{SE}(\boldsymbol{P}_{j-1}, \boldsymbol{P}_j))$, $q_k = (0.3)^k q_0$
3. $\boldsymbol{e}_t := \hat{\boldsymbol{x}}_t - \boldsymbol{x}_t$. Since $\boldsymbol{v}_t = 0$, $\boldsymbol{e}_t = \boldsymbol{\ell}_t - \hat{\boldsymbol{\ell}}_t$
4. Events: $\Gamma_{0,0} := \{\text{assumed bound on } \mathrm{SE}(\hat{\boldsymbol{P}}_0, \boldsymbol{P}_0)\}$, $\Gamma_{0,k} := \Gamma_{0,k-1} \cap \{\mathrm{SE}(\hat{\boldsymbol{P}}_{0,k}, \boldsymbol{P}_0) \leq 0.3^k \mathrm{SE}(\hat{\boldsymbol{P}}_0, \boldsymbol{P}_0)\}$, $\Gamma_{j,0} := \Gamma_{j-1,K}$, $\Gamma_{j,k} := \Gamma_{j,k-1} \cap \{\mathrm{SE}(\hat{\boldsymbol{P}}_{j,k}, \boldsymbol{P}_j) \leq q_{k-1}/4\}$ for $j = 1, 2, \ldots, J$ and $k = 1, 2, \ldots, K$.
5. Using the expression for K given in the theorem, it follows that $\Gamma_{j,K}$ implies $\mathrm{SE}(\hat{\boldsymbol{P}}_{j,K}, \boldsymbol{P}_j) \leq \varepsilon$.

Observe that if we can show that $\Pr(\Gamma_{J,K}|\Gamma_{0,0}) \geq 1 - dn^{-10}$ we will have obtained all the subspace recovery bounds of Theorem 2.1. The next two lemmas applied sequentially help show that this is true for Algorithm 1 (t_j known). The correctness of the actual algorithm follows using these and Lemma 3.6.

Lemma 3.2 (first subspace update interval)**.** *Under the conditions of Theorem 2.1, conditioned on $\Gamma_{j,0}$,*

1. *for all $t \in [\hat{t}_j, \hat{t}_j + \alpha)$, $\|\boldsymbol{\Psi}\boldsymbol{\ell}_t\| \leq (\varepsilon + \Delta)\sqrt{\eta r \lambda^+} < x_{\min}/15$, $\|\hat{\boldsymbol{x}}_{t,cs} - \boldsymbol{x}_t\| \leq 7x_{\min}/15 < x_{\min}/2$, $\hat{\mathcal{T}}_t = \mathcal{T}_t$, and the error $\boldsymbol{e}_t := \hat{\boldsymbol{x}}_t - \boldsymbol{x}_t = \boldsymbol{\ell}_t - \hat{\boldsymbol{\ell}}_t$ satisfies*
$$\boldsymbol{e}_t = \boldsymbol{I}_{\mathcal{T}_t} \left(\boldsymbol{\Psi}_{\mathcal{T}_t}{}' \boldsymbol{\Psi}_{\mathcal{T}_t}\right)^{-1} \boldsymbol{I}_{\mathcal{T}_t}{}' \boldsymbol{\Psi} \boldsymbol{\ell}_t, \tag{1}$$
and $\|\boldsymbol{e}_t\| \leq 1.2(\varepsilon + \Delta)\sqrt{\eta r \lambda^+}$.
2. *w.p. at least $1 - 10n^{-10}$, the first subspace estimate $\hat{\boldsymbol{P}}_{j,1}$ satisfies $\mathrm{SE}(\hat{\boldsymbol{P}}_{j,1}, \boldsymbol{P}_j) \leq (q_0/4)$, i.e., $\Gamma_{j,1}$ holds.*

Lemma 3.3 (k-th subspace update interval)**.** *Under the conditions of Theorem 2.1, conditioned on $\Gamma_{j,k-1}$,*

1. *for all $t \in [\hat{t}_j + (k-1)\alpha, \hat{t}_j + k\alpha - 1)$, all claims of the first part of Lemma 3.2 holds, $\|\boldsymbol{\Psi}\boldsymbol{\ell}_t\| \leq 0.3^{k-1}(\varepsilon + \Delta)\sqrt{\eta r \lambda^+}$, and $\|\boldsymbol{e}_t\| \leq (0.3)^{k-1} \cdot 1.2(\varepsilon + \Delta)\sqrt{\eta r \lambda^+}$.*
2. *w.p. at least $1 - 10n^{-10}$ the subspace estimate $\hat{\boldsymbol{P}}_{j,k}$ satisfies $\mathrm{SE}(\hat{\boldsymbol{P}}_{j,k}, \boldsymbol{P}_j) \leq (q_{k-1}/4)$, i.e., $\Gamma_{j,k}$ holds.*

Remark 3.4. *For the case of $j = 0$, in both the lemmas above, Δ gets replaced with $\mathrm{SE}(\hat{\boldsymbol{P}}_0, \boldsymbol{P}_0)$ and ε by zero.*

We prove these lemmas in the long version. The projected CS proof (part one of both lemmas) uses the following lemma from (Qiu et al., 2014) that relates the s-Restricted Isometry Constant (RIC) (Candes, 2008) of a projection matrix to the incoherence of its orthogonal complement.

Lemma 3.5. *For an $n \times r$ basis matrix $\boldsymbol{P}$, $\delta_s(\boldsymbol{I} - \boldsymbol{P}\boldsymbol{P}') = \max_{|\mathcal{T}| \leq s} \|\boldsymbol{I}_{\mathcal{T}}{}'\boldsymbol{P}\|^2 \leq s \max_{i=1,2,\ldots,n} \|\boldsymbol{I}_i{}'\boldsymbol{P}\|^2 \leq s\mu r/n$.*

The last bound follows using Definition 1.2. We apply this lemma with s = max-outlier-frac-col $\cdot\, n$. The subspace update step proof uses a guarantee for PCA in sparse data-dependent noise, Corollary 4.17, due to (Vaswani & Narayanamurthy, 2017). Notice that $\boldsymbol{e}_t = \boldsymbol{\ell}_t - \hat{\boldsymbol{\ell}}_t$ is the noise/error seen by the subspace update step. By (1), this is sparse and depends on the true data $\boldsymbol{\ell}_t$.

A careful application of the (Vaswani & Narayanamurthy, 2017) result is the reason why we are able to remove the moving object model assumption on the outlier support needed by the earlier dynamic RPCA guarantees (or orig-ReProCS). Applied to our problem, this result requires $\|\sum_{t \in \mathcal{J}^\alpha} \boldsymbol{I}_{\mathcal{T}_t} \boldsymbol{I}_{\mathcal{T}_t}{}'/\alpha\|$ to be bounded by a constant less than one. It is not hard to see that $\max_{\mathcal{J}^\alpha \in [1,d]} \|\sum_{t \in \mathcal{J}^\alpha} \boldsymbol{I}_{\mathcal{T}_t} \boldsymbol{I}_{\mathcal{T}_t}{}'/\alpha\| = \text{max-outlier-frac-row}$. This is also why a constant bound on max-outlier-frac-row suffices for our setting.

The key to our overall proof is to show that the subspace recovery error after each PCA step is sufficiently smaller (e.g. 0.3 times smaller) than the instantaneous noise/error level, $\|\mathbb{E}[\boldsymbol{e}_t \boldsymbol{e}_t{}']\|$, seen by the $\hat{\boldsymbol{\ell}}_t$'s used for this step. The reason we can show this is because the PCA step error is proportional to the ratio between the time-averaged noise level (plus time-averaged signal-noise correlatin) and the minimum signal space eigenvalue, λ^- (Vaswani & Narayanamurthy, 2017). Using the sparsity of $\boldsymbol{e}_t$ with support $\mathcal{T}_t$ and the max-outlier-frac-row bound one can show that the time-averaged noise plus signal-noise correlation, $(\|\sum_t \mathbb{E}[\boldsymbol{e}_t \boldsymbol{e}_t{}']\| + \|\sum_t \mathbb{E}[\boldsymbol{\ell}_t \boldsymbol{e}_t{}']\|)/\alpha$, is at least $\sqrt{\text{max-outlier-frac-row}}$ times its instantaneous value, $\|\mathbb{E}[\boldsymbol{e}_t \boldsymbol{e}_t{}']\| + \|\mathbb{E}[\boldsymbol{\ell}_t \boldsymbol{e}_t{}']\|$.

The above, in turn, implies that both the instantaneous and time-averaged error/noise level in the next interval is 0.3 times smaller. Put together, one can show exponential decay of both $\text{SE}(\hat{\boldsymbol{P}}_{j,k}, \boldsymbol{P}_j)$ and the error/noise level.

Consider the actual t_j unknown case. The following lemma is used to show that, whp, we can detect subspace change within 2α time instants. This lemmas assumes detection threshold $\omega_{evals} = 2\varepsilon^2 \lambda^+$ (see the long version).

Lemma 3.6 (Subspace Change Detection). *Consider an α-length time interval $\mathcal{J}^\alpha \subset [t_j, t_{j+1}]$ (so that $\boldsymbol{\ell}_t = \boldsymbol{P}_j \boldsymbol{a}_t$).*

1. *If $\boldsymbol{\Phi} := \boldsymbol{I} - \hat{\boldsymbol{P}}_{j-1} \hat{\boldsymbol{P}}_{j-1}{}'$ and $\text{SE}(\hat{\boldsymbol{P}}_{j-1}, \boldsymbol{P}_{j-1}) \le \varepsilon$, with probability at least $1 - 10n^{-10}$,*

$$\lambda_{\max}\left(\frac{1}{\alpha}\sum_{t \in \mathcal{J}^\alpha} \boldsymbol{\Phi} \hat{\boldsymbol{\ell}}_t \hat{\boldsymbol{\ell}}_t{}' \boldsymbol{\Phi}\right) \ge 0.8 \lambda^- \text{SE}^2(\boldsymbol{P}_{j-1}, \boldsymbol{P}_j) > \omega_{evals}$$

2. *If $\boldsymbol{\Phi} := \boldsymbol{I} - \hat{\boldsymbol{P}}_j \hat{\boldsymbol{P}}_j{}'$ and $\text{SE}(\hat{\boldsymbol{P}}_j, \boldsymbol{P}_j) \le \varepsilon$, with probability at least $1 - 10n^{-10}$,*

$$\lambda_{\max}\left(\frac{1}{\alpha}\sum_{t \in \mathcal{J}^\alpha} \boldsymbol{\Phi} \hat{\boldsymbol{\ell}}_t \hat{\boldsymbol{\ell}}_t{}' \boldsymbol{\Phi}\right) \le 1.37 \varepsilon^2 \lambda^+ < \omega_{evals}$$

We prove these lemmas in the long version. Theorem 2.1 is an easy consequence of these.

4. Empirical Evaluation

The complete details for all experiments are provided in (Narayanamurthy & Vaswani, 2018b). All codes can be found at `https://github.com/praneethmurthy/NORST.`.

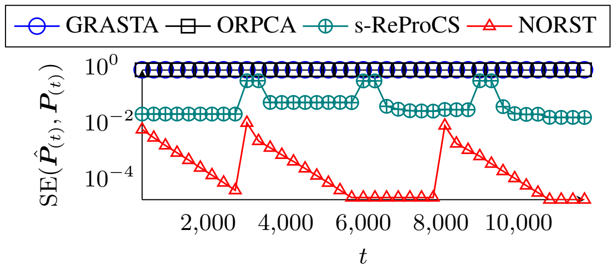

Figure 2. The plot of subspace error versus time for the online RST algorithms, plotted every $\alpha = 300$ time instants. Time taken per sample (milliseconds) is shown in legend parentheses.

Outlier Model	RPCA-GD (92.2ms)	AltProj (70.8ms)	**Offline-NORST (1.7ms)**
Mov. Obj.	4.283	4.363	$\mathbf{3.5 \times 10^{-4}}$
Bernoulli	0.158	0.269	$\mathbf{3.4 \times 10^{-4}}$

Table 2. Comparison of $\|\hat{\boldsymbol{L}} - \boldsymbol{L}\|_F / \|\boldsymbol{L}\|_F$ for offline RPCA methods including offline NORST. Average time given in parentheses.

Synthetic Data. We generated the subspaces using $\boldsymbol{P}_j = e^{\tilde{\delta}_j \boldsymbol{B}_j} \boldsymbol{P}_{j-1}$ (as done in (He et al., 2012)) where $\tilde{\delta}_j$ controls the subspace change and $\boldsymbol{B}_j$'s are skew-symmetric matrices. In our experiments we use $n = 1000$, $d = 12000$, $r = 30$, $J = 2$, $t_1 = 3000$, $t_2 = 8000$, $r = 30$, $\tilde{\delta}_1 = \tilde{\delta}_2 = 0.001$. $\boldsymbol{P}_0$ is generated by ortho-normalizing columns of an $n \times r$ i.i.d standard normal matrix. The subspace coefficients $\boldsymbol{a}_t \in \mathbb{R}^r$ are generated as independent zero-mean, bounded (uniform) random variables with condition number of their covariance $f = 50$. We generated the support $\mathcal{T}_t$ of sparse outliers using the Moving Object Model of (Narayanamurthy & Vaswani, 2018a). We used the first $t_{\text{train}} = 3.3r = 100$ frames as the training data with fewer outliers: for this period we used $b_0 = 0.01$, and for $t > t_{\text{train}}$, we used $s/n = 0.05$ and $b_0 = 0.3$. The non-zero magnitudes of $\boldsymbol{x}_t$ are generated as i.i.d as $uniform[10, 20]$. The algorithm parameters are set as $K = 8$, $\alpha = 300$. As shown in Fig. 2, NORST is significantly better than all the RST methods - s-ReProCS, and two popular heuristics from literature - ORPCA and GRASTA. We provide a comparison of offline-NORST with the batch RPCA techniques in Table 2. As can be seen, offline-NORST outperforms all the batch RPCA methods, both for the moving object outlier support model and for the commonly used random Bernoulli support model. All results in this table are averaged over 10 independent runs.

Video. We also evaluated NORST for background subtraction; see Figure 1. The NORST parameters were set as $\alpha = 60$, $K = 3$, $r = 40$ and $\xi_t = \|\boldsymbol{\Psi} \hat{\boldsymbol{\ell}}_{t-1}\|$.

Acknowledgements

The authors thank Praneeth Netrapalli and Prateek Jain of Microsoft Research India for asking an important question that helped remove a redundant assumption.

References

Candes, E. The restricted isometry property and its implications for compressed sensing. *C. R. Math. Acad. Sci. Paris Serie I*, 2008.

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

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

Cherapanamjeri, Y., Gupta, K., and Jain, P. Nearly-optimal robust matrix completion. *ICML*, 2016.

Chi, Y., Eldar, Y. C., and Calderbank, R. Petrels: Parallel subspace estimation and tracking by recursive least squares from partial observations. *IEEE Trans. Sig. Proc.*, December 2013.

Feng, J., Xu, H., and Yan, S. Online robust pca via stochastic optimization. In *NIPS*, 2013.

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

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

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

Narayanamurthy, P. and Vaswani, N. Provable dynamic robust pca or robust subspace tracking. *arXiv:1705.08948, being revised for IEEE Trans. Info. Theory (short version in ISIT'18)*, 2018a.

Narayanamurthy, P. and Vaswani, N. Nearly optimal robust subspace tracking. In *Intnl. Conf. Machine Learning (ICML), longer version at arXiv:1712.06061[cs.IT] and submitted to IEEE Trans. Info Theory*, 2018b.

Netrapalli, P., Niranjan, U. N., Sanghavi, S., Anandkumar, A., and Jain, P. Non-convex robust pca. In *NIPS*, 2014.

Ozdemir, A., Bernat, E. M., and Aviyente, S. Recursive tensor subspace tracking for dynamic brain network analysis. *IEEE Transactions on Signal and Information Processing over Networks*, 2017.

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

Vaswani, N. and Narayanamurthy, P. Finite sample guarantees for pca in non-isotropic and data-dependent noise. In *Allerton 2017, long version at arXiv:1709.06255*, 2017.

Yang, B. Projection approximation subspace tracking. *IEEE Trans. Sig. Proc.*, pp. 95–107, 1995.

Yi, X., Park, D., Chen, Y., and Caramanis, C. Fast algorithms for robust pca via gradient descent. In *NIPS*, 2016.

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

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

Zhang, D. and Balzano, L. Global convergence of a grassmannian gradient descent algorithm for subspace estimation. In *AISTATS*, 2016.