Pursuit of Low-Rank Models of Time-Varying Matrices Robust to Sparse and Measurement Noise

Albert Akhriev, Jakub Marecek, Andrea Simonetto
IBM Research – Ireland
{albert_akhriev, jakub.marecek}@ie.ibm.com, andrea.simonetto@ibm.com

Abstract

In tracking of time-varying low-rank models of time-varying matrices, we present a method robust to both uniformly-distributed measurement noise and arbitrarily-distributed "sparse" noise. In theory, we bound the tracking error. In practice, our use of randomised coordinate descent is scalable and allows for encouraging results on changedetection.net, a benchmark.

Introduction

Dimension reduction is a staple of Statistics and Machine Learning. In principal component analysis, its undergraduate-textbook version, possibly correlated observations are transformed to a combination of linearly uncorrelated variables, called principal components. Often, a low number of principal components suffice for the so-called low-rank model to represent the phenomenon observed. Notoriously, however, a small amount of noise can change the principal components considerably. A considerable effort has focussed on the development of robust approaches to principal component analysis (RPCA). Two challenges remained: robustness to both sparse and non-sparse noise and theoretical guarantees in the time-varying setting.

We present the pursuit of time-varying low-rank models of time-varying matrices, which is robust to both dense uniformly-distributed measurement noise and sparse arbitrarily-distributed noise. Consider, for example, background subtraction problem in Computer Vision, where one wishes to distinguish fast-moving foreground objects from slowly-varying background in video data (Liu et al. 2013). There, a matrix represents a constant number of frames of the video data, flattened to one row-vector per frame. At any point in time, the low-rank model is captured by a short-and-wide matrix. The time-varying low-rank model makes it possible to capture slower changes, e.g., lighting conditions slowly changing with the cloud cover. There may also be slight but rapid changes, e.g., leaves of grass moving in the wind, which could be captured by the uniformly-distributed dense noise. Finally, the moving objects are captured by the sparse noise. Clearly, low-rank modelling has wide-ranging applications beyond Computer Vision, wherever one needs to analyse high-dimensional streamed data and flag abnormal observations to operators, while adapting the model of what is normal over time.

Our contributions are as follows:

- we extend the low-rank + sparse model to low-rank + dense uniformly-distributed noise + sparse, where low-rank can be time-varying
- we provide an algorithm with convergence-rate guarantees for the time-invariant case
- we provide an algorithm with guarantees for the time-varying case. In Theorem 2, we bound the tracking error of an algorithm for any low-rank factorisation problem for the first time. That is: we show that a sequence of approximately optimal costs eventually reaches the optimal cost trajectory.
- we improve upon the statistical performance of RPCA approaches on changedetection.net of (Goyette et al. 2012), a well-known benchmark: the F1 score across 6 categories of changedetection.net improves by 28%, from 0.44643 to 0.57099. On the baseline category, it is 0.80254.
- we improve upon run time per frame of the same RPCA approaches, as detailed in Table 1. Compared to TTD_3WD, to give an example of a method which is still considered efficient in the literature, our single-threaded implementation is 103 times faster.

Related Work

Traditional approaches to robustness in low-rank models (Candès and Recht 2009, to name some of the pioneering work) are based on a long history of work in robust statistics (Huber 1981). In such approaches (Candès et al. 2011; Feng et al. 2013; Guo, Qiu, and Vaswani 2014; Mardani, Mateos, and Giannakis 2013), sometimes known as "Low-rank + Sparse", one balances the number of samples of the "sparse" noise and the rank of the model, or the nuclear norm as a proxy for the rank. There are a number of excellent implementations, including some focused on the incremental update (Lin, Liu, and Su 2011; He, Balzano, and Lui 2011; Balzano and Wright 2013; Oreifej, Li, and Shah 2013; Meng and Torre 2013; Rodriguez and Wohlberg 2013; Dutta and Li 2017; Dutta, Li, and Richtárik 2017; Ma and Aybat 2018; Lerman and Maunu 2018; Vaswani and Narayana-

Table 1: A comparison of our approach against five of the best-known RPCA implementations and the recent OMoGMF, featuring the F1 score on the baseline category of http://changedetection.net and mean run time (in seconds per input frame, single-threaded) on the "baseline/highway" video-sequence of the same benchmark.

Method	Model	Guarantees	F1	Run-time
LRR_FastLADMAP	Low-rank + Sparse	Off-line: limit point KKT	0.36194	4.611
MC_GROUSE	Low-rank + Sparse, L_2	—	0.31495	10.621
OMoGMF	GMM(Low-rank) + Sparse	—	0.72611	0.123
RPCA_FPCP	Low-rank + Sparse	—	0.37900	0.504
ST_GRASTA	Rank-1 + Sparse, L_1	—	0.42367	3.266
TTD_3WD	Low-rank + Turbulence + Sparse	Off-line: limit point feasible	0.40297	10.343
Our approach	Low-rank + Uniform + Sparse	**On-line: tracking error**	**0.80254**	**0.103**

murthy 2018; Balzano, Chi, and Lu 2018; Yong et al. 2018, e.g.). In our comparison, we focus five of the best-known implementations and one very recent one. LRR_FastLADMAP (Lin, Liu, and Su 2011), RPCA_FPCP (Rodriguez and Wohlberg 2013), and MC_GROUSE (Balzano and Wright 2013) use the low-rank + sparse model. ST_GRASTA (He, Balzano, and Lui 2011) uses rank-1 + sparse. TTD_3WD (Oreifej, Li, and Shah 2013) uses low-rank + turbulence + sparse. The most recent formulation we consider is OMoGMF (Yong et al. 2018), which utilises a Gaussian mixture model (GMM) structure over the low-rank model, plus sparse noise on top. We refer to (Bhojanapalli, Neyshabur, and Srebro 2016; Boumal, Voroninski, and Bandeira 2016; Jain and Kar 2017; Boumal, Absil, and Cartis 2018; Bhojanapalli et al. 2018) for the present-best theoretical analyses in the off-line, time-invariant case, but stress that no guarantees have been known for the on-line, time-varying case. We refer to the recent handbook (Bouwmans, Aybat, and Zahzah 2016) and to the August 2018 special issue of the Proceedings of the IEEE (Vaswani, Chi, and Bouwmans 2018) for up-to-date surveys.

Problem Formulation

Consider N streams with n-dimensional measurements, coming from N sensors with uniform sampling period h from t_k till $t_k + hT$ (possibly with many missing values), packaged in a (possibly partial) matrix $\mathbf{M}_k \in \mathbb{R}^{T \times nN}$. Every time a new observation comes in, its flattening is added at the bottom row to the matrix and the first row is discarded. In this way, the observation matrix slowly varies over time, i.e., $\mathbf{M}_{k+1}$ is different from $\mathbf{M}_k$, in general.

It is natural to assume that any row d may resemble a linear combination of $r \ll T$ prototypical rows. Prior to the corruption by sparse noise, we assume that there exists $\mathbf{R}_k \in \mathbb{R}^{r \times nN}$, such that flattened observations $\mathbf{x}_d \in \mathbb{R}^{1 \times nN}$ are

$$\mathbf{x}_d = \mathbf{c}_d \mathbf{R}_k + \mathbf{e}_d, \tag{1}$$

where the row vector $\mathbf{c}_d \in \mathbb{R}^{1 \times r}$ weighs the rows of matrix $\mathbf{R}_k$, while $\mathbf{e}_d \in \mathbb{R}^{1 \times nN}$ is the noise row vector, where each entry be uniformly distributed between known, fixed $-\Delta$ and Δ. Further, this formulation (1) is extended towards the contamination model (Huber 1981), where "sparse errors" replace readings of some of the sensors. That is: Either we receive a measurement belonging to our model, or not:

$$(\mathbf{x}_d)_i = (\mathbf{1}_n - \mathbb{I}_{i,k}) \circ [(\mathbf{c}_d \mathbf{R}_k)_i + (\mathbf{e}_d)_i] + \mathbb{I}_{i,k} \circ \mathbf{s}_i, \tag{2}$$

where index i enumerates sensors, $\mathbf{s}_i \in \mathbb{R}^{1 \times n}$ is a generic noise vector, while the Boolean vector $\mathbb{I}_{i,k} \in \{0,1\}^n$ has entries that are all zeros or ones depending on whether we receive a measurement belonging to our model or not. The operation $\circ$ represents element-wise multiplication.

Considering the matrix representation, we assume that the matrix $\mathbf{M}_k$ can be decomposed into slowly varying low-rank model ($\mathbf{C}_k \mathbf{R}_k$) and additive deviation ($\mathbf{E}_k$) from the model comprising noise and anomalies:

$$\mathbf{M}_k = \begin{bmatrix} \cdots \\ \hline \mathbf{x}_d \\ \hline \cdots \end{bmatrix} = \mathbf{C}_k \mathbf{R}_k + \mathbf{E}_k, \tag{3}$$

where T is the number of samples stacked in rows of matrix $\mathbf{M}_k$, r is the number of prototypes in the low-rank approximation, $\mathbf{x}_d$ is a d-th row-vector in matrix $\mathbf{M}_k$, $\mathbf{C}_k \in \mathbb{R}^{T \times r}$ and $\mathbf{E}_k \in \mathbb{R}^{T \times nN}$ are the matrices incorporating the coefficient vectors $\mathbf{c}_d$'s and noise $\mathbf{e}_d$'s as $\mathbf{C}_k = [\ldots; \mathbf{c}_d; \ldots]$, and $\mathbf{E}_k = [\ldots; \mathbf{e}_d; \ldots]$, respectively.

The missing entries in $\mathbf{M}_k$ can represent either really absent data or outliers, such as moving objects in the case of video-processing applications. One can assume that normal behaviour exhibits certain regularity, which could be captured by a low-rank structure, and that events or anomalies are sparse across both time and space. The sparsity should be construed quite loosely, for example, comprising dense blobs of pixels moving coherently in video data, while occupying a relatively small fraction of image pixels in total. This notion of anomaly detection is widely used in monitoring streamed data, event recognition, and computer vision.

If we can identify the low-rank model, any deviation from the measurement model (1) can be interpreted as an anomaly or event. When there are few measurements for which $\mathbb{I}_{i,k} = \mathbf{1}_n$ and those are different from standard measurements, i.e., the aggregated $\mathbb{I}_k \in \{0,1\}^{nN}$, which stacks all the individual $\mathbb{I}_k$ for a specific time k, is sparse, and samples of s_i fall outside of some range $[\underline{M}_{k,ij}, \overline{M}_{k,ij}]$ (defined below), it is possible to identify samples of s_i perfectly. In this paper, we provide a way to detect such anomalies, i.e., measurements for which $\mathbb{I}_{i,k} = \mathbf{1}_n$. Hence, we are effectively proposing a principal component pursuit algorithm robust to uniform and sparse noise.

We compute matrices $\mathbf{C}_k$ and $\mathbf{R}_k$ by resorting to a low-rank approximation of the matrix $\mathbf{M}_k$ with an explicit consideration of the uniformly-distributed error in the measurements. Let $M_{k,ij}$ be the (i, j) element of $\mathbf{M}_k$. Consider the interval uncertainty set $[M_{k,ij} - \Delta, M_{k,ij} + \Delta]$ around each observation. Finding $(\mathbf{C}_k, \mathbf{R}_k)$ can be seen as matrix completion with element-wise lower bounds $\underline{M}_{k,ij} := M_{k,ij} - \Delta$ and element-wise upper bounds $\overline{M}_{k,ij} := M_{k,ij} + \Delta$. Let $\mathbf{C}_{k,i:}$ and $\mathbf{R}_{k,:j}$ be the i-th row and j-th column of $\mathbf{C}_k$ and $\mathbf{R}_k$, respectively. With Frobenius-norm regularisation, the completion problem we solve is:

$$\underset{\mathbf{C}_k \in \mathbb{R}^{T \times r},\ \mathbf{R}_k \in \mathbb{R}^{r \times nN}}{\text{minimise}} f(\mathbf{C}_k, \mathbf{R}_k; \mathbf{M}_k), \tag{4}$$

where:

$$\begin{aligned} f(\mathbf{C}_k, \mathbf{R}_k; \mathbf{M}_k) := {} & \tfrac{1}{2} \textstyle\sum_{(ij)} \ell(\underline{M}_{k,ij} - \mathbf{C}_{k,i:} \mathbf{R}_{k,:j}) \\ + {} & \tfrac{1}{2} \textstyle\sum_{(ij)} \ell(\mathbf{C}_{k,i:} \mathbf{R}_{k,:j} - \overline{M}_{k,ij}) \\ + {} & \tfrac{\nu}{2} \|\mathbf{C}_k\|_F^2 + \tfrac{\nu}{2} \|\mathbf{R}_k\|_F^2, \end{aligned} \tag{5}$$

where $\ell : \mathbb{R} \to \mathbb{R}$ is the square of the maximum of the two-element set composed of the argument and 0, as detailed in Section "A Derivation of the Step Size" of (Akhriev, Marecek, and Simonetto 2018), and $\nu > 0$ is a weight.

Our only further assumption is that we have the element-wise constraints on all elements of the matricial variable:

Assumption 1. *For each (i, j) of $\mathbf{M}_k$ there is a finite element-wise upper bound $\overline{M}_{k,ij}$ and a finite element-wise lower bound $\underline{M}_{k,ij}$.*

This assumption is satisfied even for any missing values at ij when the measurements lie naturally in a bounded set, e.g., $[0, 255]$ in many computer-vision applications.

Proposed Algorithms

In this section, we first present the overall schema of our approach in Algorithm 1. Second, we present Algorithm 2 for on-line inequality-constrained matrix completion, a crucial sub-problem.

The Overall Schema

Overall, we interleave the updates to the low-rank model via the inequality-constrained matrix completion, detection of sparse noise, and updating of the inputs to the inequality-constrained matrix completion, which disregards the sparse noise.

At each time step, we acquire new measurements $\mathbf{x}_d$ and compute their projection coefficients onto the low-rank subspace as

$$\mathbf{v} = \arg \min_{\mathbf{v} \in \mathbb{R}^{1 \times r}} \|\mathbf{x}_d - \mathbf{v} \mathbf{R}_{k-1}\|_p, \tag{6}$$

where p can be the $1, 2, \infty$ norm, or the 0 pseudo-norm. Since for a very large number of sensors, even solving (6) can be challenging, we subsample $\mathbf{x}_d$ by picking only a few sensors uniformly at random. Let $i \in \tilde{\mathcal{N}}$ be the sampled sensors, with $|\tilde{\mathcal{N}}| = \tilde{N}$. We form a low-dimensional measurement vector $\tilde{\mathbf{x}}_d \in \mathbb{R}^{1 \times n\tilde{N}}$ and solve the subsampled:

$$\tilde{\mathbf{v}} = \arg \min_{\mathbf{v} \in \mathbb{R}^{1 \times r}} \|\tilde{\mathbf{x}}_d - \mathbf{v} (\mathbf{R}^i_{k-1})_{i \in \tilde{\mathcal{N}}}\|_p, \tag{7}$$

where $(\mathbf{R}^i_{k-1})_{i \in \tilde{\mathcal{N}}} \in \mathbb{R}^{r \times n\tilde{N}}$ is the matrix whose columns corresponds to the sensors, which are sampled uniformly at random. Solving (7) yields solutions $\tilde{\mathbf{v}}$ such that the norm $\|\mathbf{v} - \tilde{\mathbf{v}}\|_p$ is very small, while being considerably less demanding computationally.

```
**Input**: Initial matrices $(\mathbf{C}_0, \mathbf{R}_0)$, rank $r$
**Output**: $(\mathbf{C}_k, \mathbf{R}_k)$ and events for each $k$
for each time $t_k : k = 1, 2, \ldots, t_{k+1} - t_k = h$ do
    acquire new measurements $\mathbf{x}_d$
    subsample $\mathbf{x}_d$ uniformly at random to obtain $\tilde{\mathbf{x}}_d$
    compute $\tilde{\mathbf{v}}$ via the subsampled projection (7)
    for each sensor $i$ in parallel do
        compute residuals $r_i = \|(\mathbf{x}_d)_i - (\tilde{\mathbf{v}} \mathbf{R}_{k-1})_i\|$
    end for
    compute $\lambda$ as a function of $\{r_i\}_i$ as described in
    (Akhriev, Marecek, and Simonetto 2018)
    compute $T$ as a value at risk at $\lambda$ of $\{r_i\}$
    initialise $\mathbf{y}$ as a boolean all-False vector of same
    dimension as $\mathbf{x}_d$
    for each sensor $i$ in parallel do
        if $r_i < T$ then
            set $\mathbf{y}_i$ to True, as value at sensor $i$ is likely
            to come from our model
            add $(\mathbf{x}_d)_i$ to $\mathbf{M}_k$
        end if
    end for
    compute $(\mathbf{C}_k, \mathbf{R}_k)$ via Algorithm 2 with rank $r$
end for
return $(\mathbf{C}_k, \mathbf{R}_k, \mathbf{y})$
```

Algorithm 1: Pursuit of low-rank models of time-varying matrices robust to both sparse and measurement noise.

Once the projection coefficients $\mathbf{v}$ have been computed, we can compute the discrepancy between the measurement $(\mathbf{x}_d)_i$ coming from sensor i and our projection (7), $\|(\mathbf{x}_d)_i - (\mathbf{v} \mathbf{R}_{k-1})_i\|_p$, also known as the residual for sensor i. We use the residuals in a two-step thresholding procedure inspired by (Malistov 2014). In the first step, we use residuals to compute a coefficient $\lambda > 0$. In the second step, we consider the individual residuals as samples of an empirical distribution, and take the value at risk (VaR) at λ as a threshold. We provide details in (Akhriev and Marecek 2019; Akhriev, Marecek, and Simonetto 2018). The test as to whether residual at each sensor is below the threshold results in a binary map, suggesting whether the observation of each sensor is likely to have come from our model or not. For a positive value at i in the map, the measurement $(\mathbf{x}_d)_i$ is kept in $\mathbf{M}_k$. Otherwise, it is discarded.

On-line Matrix Completion

Given $\mathbf{M}_k$, we utilise *inequality-constrained matrix completion*, to estimate the low-rank approximation $(\mathbf{C}_k, \mathbf{R}_k)$ of the input matrix considering interval uncertainty sets.

Clearly, solving the non-convex problem (4) for non-trivial dimensions of matrix $\mathbf{M}_k$ to a non-trivial accuracy at high-frequency requires careful algorithm design. We propose an algorithm that tracks the low-rank $\mathbf{R}_k$ over time, increasing the accuracy of the solution of (4) while new observations are brought in, and old ones are discarded. In particular, we propose the on-line alternating parallel randomised block-coordinate descent method summarized in Algorithm 2.

For each input k, the previously-found approximate solutions $(\mathbf{C}_{k-1}, \mathbf{R}_{k-1})$, are updated based on the new observation matrix $\mathbf{M}_k$, the correspondingly-derived element-wise lower and upper bounds $\underline{M}_{k,ij}, \overline{M}_{k,ij}$, and the desired rank r. The update is computed using the alternatig least squares (ALS) method, which is based on the observation that while f (4) is not convex jointly in $(\mathbf{C}_k, \mathbf{R}_k)$, it is convex in $\mathbf{C}_k$ for fixed $\mathbf{R}_k$ and in $\mathbf{R}_k$ for fixed $\mathbf{C}_k$. The update takes the form of a sequence $\{(\mathbf{C}_k^{T,\tau}, \mathbf{R}_k^{T,\tau})\}$ of solutions, which are progressively more accurate. If we could run a large number of iterations of the ALS, we would be in an off-line mode. In the on-line mode, we keep the number of iterations small, and apply the final update based on $\mathbf{M}_k$ at time t_{k+1}, when the next observation arrives.

The optimisation in each of the two alternating least-squares problems is based on parallel block-coordinate descent, as reinterpreted by (Nesterov 2012). Notice that in Nesterov's optimal variant, one requires the the modulus of Lipschitz continuity restricted to the sampled coordinates (Nesterov 2012, Equation 2.4) to compute the step δ. Considering that the modulus is not known *a priori*, we maintain an estimate $W_{i\hat{r}}^{T,\tau}$ of the modulus of Lipschitz continuity restricted to the $\mathbf{C}_{k,i\hat{r}}^{T,\tau}$ sampled, and estimate $V_{\hat{r}j}^{T,\tau}$ of the modulus of Lipschitz continuity restricted to the $\mathbf{R}_{k,\hat{r}j}^{T,\tau}$ sampled. We refer to (Akhriev, Marecek, and Simonetto 2018) for the details of the estimate and to (Nesterov 2012) for a high-level overview.

Overall, when looking at Algorithm 2, notice that there are several nested loops. The counter for the update of the input is k. For each input, we consider factors $\mathbf{C}$ and $\mathbf{R}$ as the optimisation variable alternatingly, with counter T. For each factor, we take a number of block-coordinate descent steps, with the blocks sampled randomly; the counter for the block-coordinate steps is τ. In particular, in Steps 3–8 of the algorithm, we fix $\mathbf{R}_k^{T,\tau}$, choose a random $\hat{r}$ and a random set $\hat{S}_{\text{row}}$ of rows of $\mathbf{C}_k$, and, in parallel for $i \in \hat{S}_{\text{row}}$, update $\mathbf{C}_{k,i\hat{r}}^{T,\tau+1}$ to $\mathbf{C}_{k,i\hat{r}}^{T,\tau} + \delta_{i\hat{r}}$, where the step is:

$$\delta_{i\hat{r}} := -\langle \nabla_{\mathbf{C}_k} f(\mathbf{C}_k^{T,\tau}, \mathbf{R}_k^{T,\tau}; \mathbf{M}_k), \mathbf{P}_{i\hat{r}} \rangle / W_{i\hat{r}}^{T,\tau}, \qquad (8)$$

and $\mathbf{P}_{i\hat{r}}$ is the $n \times r$ matrix with 1 in entry $(i\hat{r})$ and zeros elsewhere. The computation of $\langle \nabla_{\mathbf{C}_k} f(\mathbf{C}_k^{T,\tau}, \mathbf{R}_k^{T,\tau}; \mathbf{M}_k), \mathbf{P}_{\hat{r}j} \rangle$ can be simplified considerably, as explained in in Section "A Derivation of the Step Size" of (Akhriev, Marecek, and Simonetto 2018).

Likewise, in Steps 9–14, we fix $\mathbf{C}_k^{T,\tau+1}$, choose a $\hat{r}$ and a random set $\hat{S}_{\text{column}}$ of columns of $\mathbf{R}_k$, and, in parallel for $j \in \hat{S}_{\text{column}}$, update $R_{k,\hat{r}j}^{T,\tau+1}$ to $R_{k,\hat{r}j}^{T,\tau} + \delta_{\hat{r}j}$, where the step is:

$$\delta_{\hat{r}j} := -\langle \nabla_{\mathbf{R}_k} f(\mathbf{C}_k^{T,\tau+1}, \mathbf{R}_k; \mathbf{M}_k), \mathbf{P}_{\hat{r}j} \rangle / V_{\hat{r}j}^{T,\tau}, \qquad (9)$$

and $\mathbf{P}_{\hat{r}j}$ is the $r \times m$ matrix with 1 in entry $(\hat{r}j)$ and zeros elsewhere. Again, the computation of $\langle \nabla_{\mathbf{R}_k} f(\mathbf{C}_k^{T,\tau+1}, \mathbf{R}_k; \mathbf{M}_k), \mathbf{P}_{\hat{r}j} \rangle$ can be simplified.

```
Input: updated $\mathbf{M}_k$, $\underline{M}_{k,ij}, \overline{M}_{k,ij}$, previous iterate $(\mathbf{C}_{k-1}, \mathbf{R}_{k-1})$, rank $r$, limit $\overline{\tau}$
Output: $(\mathbf{C}_k, \mathbf{R}_k)$
1: Initialise: $(\mathbf{C}_k^{0,0} = \mathbf{C}_{k-1}, \mathbf{R}_k^{0,0} = \mathbf{R}_{k-1}), T = 0$
2: while $\mathbf{M}_{k+1}$ is not available do
3:   for $\tau = 0, 1, 2, \ldots, \overline{\tau}$ do
4:     choose $\hat{S}_{\text{row}} \subseteq \{1, \ldots, m\}$
5:     for $i \in \hat{S}_{\text{row}}$ in parallel do
6:       choose $\hat{r} \in \{1, \ldots, r\}$ uniformly at random
7:       compute $\delta_{i\hat{r}}$ using formula (8)
8:       update $\mathbf{C}_{k,i\hat{r}}^{T,\tau+1} \leftarrow \mathbf{C}_{k,i\hat{r}}^{T,\tau} + \delta_{i\hat{r}}$
9:     end for
10:  end for
11:  for $\tau = 0, 1, 2, \ldots, \overline{\tau}$ do
12:    choose $\hat{S}_{\text{column}} \subseteq \{1, \ldots, n\}$ uniformly at random
13:    for $j \in \hat{S}_{\text{column}}$ in parallel do
14:      choose $\hat{r} \in \{1, \ldots, r\}$ uniformly at random
15:      compute $\delta_{\hat{r}j}$ using (9)
16:      update $\mathbf{R}_{k,\hat{r}j}^{T,\tau+1} \leftarrow \mathbf{R}_{k,\hat{r}j}^{T,\tau} + \delta_{\hat{r}j}$
17:    end for
18:  end for
19:  set: $\mathbf{C}_k^{T+1,0} = \mathbf{C}_k^{T,\overline{\tau}+1}$, $\mathbf{R}_k^{T+1,0} = \mathbf{R}_k^{T,\overline{\tau}+1}$
20:  update: $T = T + 1$
21: end while
22: return $\mathbf{C}_k = \mathbf{C}_k^{T,0}, \mathbf{R}_k = \mathbf{R}_k^{T,0}$
```

Algorithm 2: On-line inequality-constrained matrix-completion via randomised coordinate descent.

Convergence Analysis

For the off-line inequality-constrained matrix completion problem (4), (Marecek, Richtarik, and Takac 2017) proposed an algorithm similar to Algorithm 2 and presented a convergence result, which states that the method is monotonic and, with probability 1, converges to the so-called bistable point, i.e., $\liminf_{T\to\infty} \|\nabla_{\mathbf{C}} f(\mathbf{C}^\tau, \mathbf{R}^\tau; \mathbf{M})\| = 0$, and $\liminf_{T\to\infty} \|\nabla_{\mathbf{R}} f(\mathbf{C}^\tau, \mathbf{R}^\tau; \mathbf{M})\| = 0$. Here, we need to show the rate of convergence to the bistable point and a distance of the bi-stable point to an optimum f^*:

Theorem 2. *There exists $\overline{\tau} > 0$, such that Algorithm 2 with the initialization to all-zero vector after at most $T = O(\log \frac{1}{\epsilon})$ steps has $f(\mathbf{C}^T, \mathbf{R}^T) \le f^* + \epsilon$ with probability 1.*

The proof is available on-line (Akhriev, Marecek, and Simonetto 2018) and should not be surprising, in light of (Bhojanapalli, Neyshabur, and Srebro 2016; Boumal, Voroninski, and Bandeira 2016; Jain and Kar 2017; Boumal, Absil, and Cartis 2018; Bhojanapalli et al. 2018).

Building upon this, we can prove a bound on the error in the on-line regime. In particular, we will show that Algorithm 2 generates a sequence of matrices $\{(\mathbf{C}_k, \mathbf{R}_k)\}$ that in the large limit of $k \to \infty$ guarantees a bounded tracking error, i.e., $f(\mathbf{C}_k, \mathbf{R}_k; \mathbf{M}_k) \leq f(\mathbf{C}^*_k, \mathbf{R}^*_k; \mathbf{M}_k) + E$. The size of the tracking error E depends on how fast the time-varying matrices change:

Assumption 3. *The variation of the observation matrix $\mathbf{M}_k$ at two subsequent instant k and $k-1$ is so to guarantee that*

$$|f(\mathbf{C}_k, \mathbf{R}_k; \mathbf{M}_k) - f(\mathbf{C}_k, \mathbf{R}_k; \mathbf{M}_{k-1})| \leq e,$$

for all instants $k > 0$.

Now, let us bound the error in tracking, i.e., when M_k changes over time and we run only a limited number of iterations τ of our algorithm per time step, before obtaining new inputs.

Theorem 4. *Let Assumptions 1 and 3 hold. Then with probability 1, Algorithm 2 starting from an all-zero matrices generates a sequence of matrices $\{(\mathbf{C}_k, \mathbf{R}_k)\}$ for which*

$$f(\mathbf{C}_k, \mathbf{R}_k; \mathbf{M}_k) - f(\mathbf{C}^*_k, \mathbf{R}^*_k; \mathbf{M}_k) \leq$$
$$\eta_0 (f(\mathbf{C}_{k-1}, \mathbf{R}_{k-1}; \mathbf{M}_{k-1}) - f(\mathbf{C}^*_{k-1}, \mathbf{R}^*_{k-1}; \mathbf{M}_{k-1})) + \eta_0 e,$$

where $\eta_0 < 1$ is a constant. In the limit,

$$\limsup_{k \to \infty} f(\mathbf{C}_k, \mathbf{R}_k; \mathbf{M}_k) - f(\mathbf{C}^*_k, \mathbf{R}^*_k; \mathbf{M}_k) \leq \frac{\eta_0 e}{1 - \eta_0} =: E.$$

In other words, as time passes, our on-line algorithm generates a sequence of approximately optimal costs that eventually reaches the optimal cost *trajectory*, up to an asymptotic bound. We bound from above the maximum discrepancy between the approximate optimum and the true one at instant k, as k goes to infinity. The convergence to the bound is linear and the rate is η_0, and depends on the properties of the cost function, while the asymptotic bound depends on how fast the problem is changing over time.

This is a *tracking* result: we are pursuing a time-varying optimum by a finite number of iterations τ per time-step. If we could run a large number of iterations per each time step, then we would be back to a off-line case and we would not have a tracking error. This may not, however, be possible in settings, where inputs change faster than one can compute an iteration of the algorithm.

Experimental Evaluation

We have implemented Algorithms 1 and 2 in C++, and released the implementation[1] under Apache License 2.0. Based on limited experimentation, we have decided on the use of a time window of $T = 35$, rank $r = 4$, and half-width of the uniform noise $\Delta = 5$. We have used dual simplex from IBM ILOG CPLEX 12.8 as a linear-programming solver for solving solving (7) in Algorithm 1. To initialise the $\mathbf{C}_0$ and $\mathbf{R}_0$ in Algorithm 1, we have used the matrix completion of Algorithms 2 with 1 epoch per frame for 3 passes on each video (4,000 to 32,000 frames), starting from all-zero matrices. We note that in real-world deployments, such an initialisation may be unnecessary, as the the number of frames processed will render the initial error irrelevant.

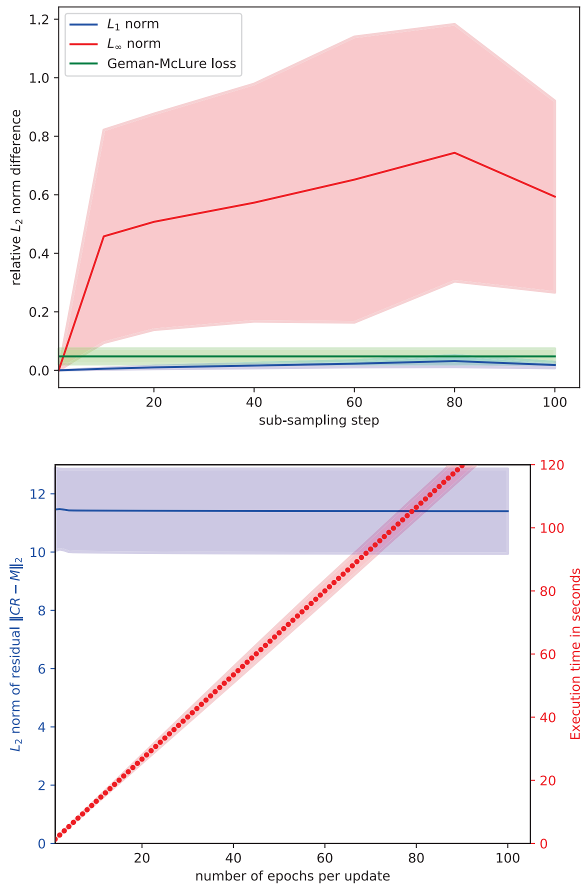

Figure 1: Top: Effects of subsampling in the projection (7). Bottom: Performance of Algorithm 2 as a function of the number of epochs per update.

First, let us highlight two aspects of the performance of the algorithm. In particular, on the top in Figure 1, we illustrate the effects of the subsampling on the projection (7). For projection in L_1 and L_∞, we present the L_2 norm of the difference $\tilde{\mathbf{v}} - \mathbf{v}$ as a function of the sample period of the subsampling (7), where $\mathbf{v}$ is the true value obtained in (6) without subsampling and $\tilde{\mathbf{v}}$ is the value obtained in (7) with subsampling, and the sample period is the ratio of the dimensions of $\mathbf{x}_d$ and $\tilde{\mathbf{x}}_d$. It is clear that L_1 is very robust to the subsampling. This corroborates the PAC bounds of (Marecek et al. 2018) and motivated our choice of L_1 with a sampling period of 100 pixels in the code. For completeness, we also present the performance of the Geman-McLure loss (Sawhney and Ayer 1996), where we do not consider subsampling, relative to the performance of L_1 norm without subsampling.

Next, on the bottom in Figure 1, we showcase the L_2 norm of residual $\mathbf{C}_k\mathbf{R}_k - \mathbf{M}_k$ and the per-iteration run-time

[1] https://github.com/jmarecek/OnlineLowRank

Table 2: Results of our Algorithm 2, compared to 6 other approaces on the "baseline" category of http://changedetection.net, evaluated on the 6 performance metrics of (Goyette et al. 2012). For each performance metric, the best result across the presented methods is highlighted in bold.

Approach / Performance metric	Recall	Specificity	FPR	FNR	Precision	F1
LRR_FastLADMAP (Lin, Liu, and Su 2011)	0.74694	0.93980	0.06021	0.25306	0.28039	0.36194
MC_GROUSE (Balzano and Wright 2013)	0.65640	0.89692	0.10308	0.34360	0.25425	0.31495
OMoGMF (Meng and Torre 2013; Yong et al. 2018)	**0.89943**	0.98289	0.01711	**0.10057**	0.62033	0.72611
RPCA_FPCP (Rodriguez and Wohlberg 2013)	0.73848	0.94733	0.05267	0.26152	0.29994	0.37900
ST_GRASTA (He, Balzano, and Lui 2011)	0.45340	0.98205	0.01795	0.54660	0.44009	0.42367
TTD_3WD (Oreifej, Li, and Shah 2013)	0.61103	0.97117	0.02883	0.38897	0.35557	0.40297
Algorithm 2 (w/ Geman-McLure)	0.85684	**0.99078**	**0.00922**	0.14316	**0.77210**	**0.80254**
Algorithm 2 (w/ L_1 norm)	0.84561	0.99063	0.00937	0.15439	0.76709	0.79421

Table 3: Results of our Algorithm 2, compared to 3 other approaches on 6 categories of http://changedetection.net, evaluated on the 6 performance metrics of (Goyette et al. 2012). For each pair of performance metric and category, the best result across the presented methods is highlighted in bold.

Approach and category / Performance metric	Recall	Specificity	FPR	FNR	Precision	F1
Algorithm 2 (w/ L_1 norm):						
badWeather	0.86589	**0.98814**	0.01186	0.13411	0.54689	0.64618
baseline	0.84561	**0.99063**	**0.00937**	0.15439	**0.76709**	**0.79421**
cameraJitter	0.59694	**0.95928**	**0.04072**	0.40306	**0.55402**	**0.51324**
dynamicBackground	0.46324	**0.99677**	**0.00323**	0.53676	**0.65511**	**0.49254**
nightVideo	**0.83646**	0.87469	0.12531	**0.16354**	0.20992	0.29481
shadow	**0.76158**	**0.97612**	**0.02388**	**0.23842**	**0.64121**	**0.68493**
Overall	0.72829	**0.96427**	**0.03573**	0.27171	**0.56237**	**0.57099**
OMoGMF (Yong et al. 2018):						
badWeather	**0.86871**	0.98939	0.01061	**0.13129**	**0.57917**	**0.67214**
baseline	**0.89943**	0.98289	0.01711	**0.10057**	0.62033	0.72611
cameraJitter	**0.85954**	0.90739	0.09261	**0.14046**	0.30567	0.44235
dynamicBackground	**0.87655**	0.86383	0.13617	**0.12345**	0.08601	0.15012
nightVideo	0.75607	0.92372	0.07628	0.24393	**0.23252**	**0.31336**
shadow	0.55772	0.80276	0.03057	0.27562	0.40539	0.37450
Overall	**0.80300**	0.91166	0.06056	**0.16922**	0.37151	0.44643
ST_GRASTA (He, Balzano, and Lui 2011):						
badWeather	0.26555	0.98971	**0.01029**	0.73445	0.45526	0.30498
baseline	0.45340	0.98205	0.01795	0.54660	0.44009	0.42367
cameraJitter	0.51138	0.91313	0.08687	0.48862	0.23995	0.31572
dynamicBackground	0.41411	0.94755	0.05245	0.58589	0.08732	0.13736
nightVideo	0.42488	**0.97224**	**0.02776**	0.57512	**0.24957**	0.28154
shadow	0.44317	0.96681	0.03319	0.55683	0.42604	0.41515
Overall	0.41875	0.96192	0.03808	0.58125	0.31637	0.31307
RPCA_FPCP (Rodriguez and Wohlberg 2013):						
badWeather	0.82546	0.84424	0.15576	0.17454	0.09950	0.16687
baseline	0.73848	0.94733	0.05267	0.26152	0.29994	0.37900
cameraJitter	0.74452	0.84143	0.15857	0.25548	0.18436	0.29024
dynamicBackground	0.69491	0.80688	0.19312	0.30509	0.03928	0.07134
nightVideos	0.79284	0.85751	0.14249	0.20716	0.11797	0.19497
shadow	0.72132	0.90454	0.09546	0.27868	0.26474	0.36814
Overall :	0.75292	0.86699	0.13301	0.24708	0.16763	0.24509

of a single-threaded implementation as a function of the number of epochs per update. Clearly, the decrease in the residual is very slow beyond one epoch per update, due to the reasonable initialisation. On the other hand, there is a linear increase in per-iteration run-time with the number of epochs of coordinate descent per update. This motivated our choice of 1 epoch per update, which allows for real-time processing at 10 frames per second *without* parallelisation, which can further improve performance as suggested in Algorithm 2.

We have also conducted a number of experiments on instances from changedetection.net (Goyette et al. 2012), a benchmark often used to test low-rank approaches. There, short videos (1,000 to 9,000 frames) are supplemented with ground-truth information of what is foreground and what is background. These experiments have been run on a single 4-core workstation (Intel Core i7-4800MQ CPU, 16 GB of RAM, RedHat 7.6/64) and results have been deposited[2] in FigShare. In Tables 2 and 3, we summarise the results. In particular, we present the false positive rate (FPR), false negative rate (FNR), specificity, precision, recall, and the geometric mean of the latter two (F1) of our method and 6 other low-rank approaches, which have been used as reference methods recently (Bouwmans, Aybat, and Zahzah 2016). These reference methods are implemented in `LRSLibrary` (Sobral, Bouwmans, and Zahzah 2015; Bouwmans et al. 2015) and by the original authors of `OMoGMF` (Meng and Torre 2013; Yong et al. 2018), and have been used with their default settings. Out of these, OMoGMF (Yong et al. 2018) is the most recent and considered to be the most robust. Still, we can improve upon the results of OMoGMF by a considerable margin: the F1 score across the 6 categories is improved by 28% from 0.44643 to 0.57099, for example.

Further details and results are available in (Akhriev, Marecek, and Simonetto 2018). At http://changedetection.net/, a comparison against four dozen other methods is readily available, although one should like to discount methods tagged as "supervised", which are trained and tested on one and the same dataset. A further comparison against dozens of other methods is available in (Vaswani et al. 2018).

Conclusions

We have presented a tracking result for time-varying low-rank models of time-varying matrices, robust to both uniformly-distributed measurement noise and arbitrarily-distributed "sparse" noise. This improves upon prior work, as summarised by the recent special issues (Vaswani et al. 2018; Vaswani, Chi, and Bouwmans 2018).

Our analytical guarantees improve upon the state of the art in two ways. First, we provide a bound on the tracking error in estimation of the time-varying low-rank sub-space, rather than a result restricted to the off-line case. Second, we do not make restrictive assumptions on RIP properties, incoherence, identical covariance matrices, independence of all outlier supports, or initialisation. Broadly speaking, such analyses of *time-varying non-convex optimisation* (Liu et al. 2018; Tang et al. 2018; Fattahi et al. 2019; Massicot and Marecek 2019), seems to be an important direction for further research.

In practice, our use of randomised coordinate descent in alternating least-squares seems much better suited to high-volume (high-dimensional, high-frequency) data streams than spectral methods and other alternatives we are aware of. When the matrix $\mathbf{M}_k$ does not change quickly, performing a fixed number of iterations within an inexact step (4) upon arrival of a new sample makes it possible to spread the computational load over time, while still recovering a good background model. Also, our algorithm is easy to implement and optimize. It has very few hyper-parameters, and this simplifies tuning. Our results are hence practically relevant.

Acknowledgments

This research received funding from the European Union Horizon 2020 Programme (Horizon2020/2014-2020) under grant agreement number 688380 (project VaVeL).

References

Akhriev, A., and Marecek, J. 2019. Deep autoencoders with value-at-risk thresholding for unsupervised anomaly detection. In *IEEE International Symposium on Multimedia (ISM)*, to appear.

Akhriev, A.; Marecek, J.; and Simonetto, A. 2018. Pursuit of low-rank models of time-varying matrices robust to sparse and measurement noise. *arXiv preprint arXiv:1809.03550*. Full version.

Balzano, L., and Wright, S. J. 2013. On GROUSE and incremental SVD. In *2013 5th IEEE Int. Workshop on Comp. Advances in Multi-Sensor Adaptive Proc. (CAMSAP)*, 1–4.

Balzano, L.; Chi, Y.; and Lu, Y. M. 2018. Streaming PCA and subspace tracking: The missing data case. *Proceedings of the IEEE* 106(8):1293–1310.

Bhojanapalli, S.; Boumal, N.; Jain, P.; and Netrapalli, P. 2018. Smoothed analysis for low-rank solutions to semidefinite programs in quadratic penalty form. *Conference on Learning Theory (COLT)*.

Bhojanapalli, S.; Neyshabur, B.; and Srebro, N. 2016. Global optimality of local search for low rank matrix recovery. In *Advances in Neural Information Processing Systems 29*, 3873–3881.

Boumal, N.; Absil, P.-A.; and Cartis, C. 2018. Global rates of convergence for nonconvex optimization on manifolds. *IMA Journal of Numerical Analysis* 39(1):1–33.

Boumal, N.; Voroninski, V.; and Bandeira, A. 2016. The non-convex burer-monteiro approach works on smooth semidefinite programs. In *Advances in Neural Information Processing Systems 29*. 2757–2765.

Bouwmans, T.; Aybat, N. S.; and Zahzah, E.-h. 2016. *Handbook of robust low-rank and sparse matrix decomposition: Applications in image and video processing*. Chapman and Hall/CRC.

Bouwmans, T.; Sobral, A.; Javed, S.; Jung, S. K.; and Zahzah, E.-h. 2015. Decomposition into low-rank plus additive matrices for background/foreground separation: A review for a comparative evaluation with a large-scale dataset. *CoRR* abs/1511.01245.

[2] https://figshare.com/articles/AAAI2020_results_zip/10316696

Candès, E. J., and Recht, B. 2009. Exact matrix completion via convex optimization. *Foundations of Computational mathematics* 9(6):717.

Candès, E. J.; Li, X.; Ma, Y.; and Wright, J. 2011. Robust principal component analysis? *Journal of the ACM* 58(3):11.

Dutta, A., and Li, X. 2017. On a problem of weighted low-rank approximation of matrices. *SIAM Journal on Matrix Analysis and Applications* 38(2):530–553.

Dutta, A.; Li, X.; and Richtárik, P. 2017. A batch-incremental video background estimation model using weighted low-rank approximation of matrices. In *Proceedings of the IEEE International Conference on Computer Vision*, 1835–1843.

Fattahi, S.; Josz, C.; Mohammadi, R.; Lavaei, J.; and Sojoudi, S. 2019. Absence of spurious local trajectories in time-varying optimization. *arXiv preprint arXiv:1905.09937*.

Feng, J.; Xu, H.; Mannor, S.; and Yan, S. 2013. Online PCA for contaminated data. In Burges, C. J. C.; Bottou, L.; Welling, M.; Ghahramani, Z.; and Weinberger, K. Q., eds., *Advances in Neural Information Processing Systems 26*. Curran Associates, Inc. 764–772.

Goyette, N.; Jodoin, P.-M.; Porikli, F.; Konrad, J.; and Ishwar, P. 2012. Changedetection. net: A new change detection benchmark dataset. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2012 IEEE Computer Society Conference on*, 1–8. IEEE.

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

He, J.; Balzano, L.; and Lui, J. C. S. 2011. Online Robust Subspace Tracking from Partial Information. *arXiv e-prints* arXiv:1109.3827.

Huber, P. J. 1981. *Robust Statistics*. Wiley-Interscience.

Jain, P., and Kar, P. 2017. *Non-convex optimization for machine learning*, volume 10 of *Foundations and Trends® in Machine Learning*. Now Publishers.

Lerman, G., and Maunu, T. 2018. An overview of robust subspace recovery. *Proceedings of the IEEE* 106(8):1380–1410.

Lin, Z.; Liu, R.; and Su, Z. 2011. Linearized alternating direction method with adaptive penalty for low-rank representation. In Shawe-Taylor, J.; Zemel, R. S.; Bartlett, P. L.; Pereira, F.; and Weinberger, K. Q., eds., *Advances in Neural Information Processing Systems 24*. 612–620.

Liu, G.; Lin, Z.; Yan, S.; Sun, J.; Yu, Y.; and Ma, Y. 2013. Robust recovery of subspace structures by low-rank representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35(1):171–184.

Liu, J.; Marecek, J.; Simonetta, A.; and Takac, M. 2018. A coordinate-descent algorithm for tracking solutions in time-varying optimal power flows. In *2018 Power Systems Computation Conference (PSCC)*, 1–7.

Ma, S., and Aybat, N. S. 2018. Efficient optimization algorithms for robust principal component analysis and its variants. *Proceedings of the IEEE* 106(8):1411–1426.

Malistov, A. 2014. Estimation of background noise in traffic conditions and selection of a threshold for selecting mobile objects. *Actual issues of modern science* 4. In Russian with English abstract in pp. 12-13 at http://www.malistov.ru/docs/dissertation/abstract_malistov.pdf.

Mardani, M.; Mateos, G.; and Giannakis, G. B. 2013. Dynamic anomalography: Tracking network anomalies via sparsity and low rank. *IEEE Journal of Selected Topics in Signal Processing* 7(1):50–66.

Marecek, J.; Maroulis, S.; Kalogeraki, V.; and Gunopulos, D. 2018. Low-rank methods in event detection. *arXiv preprint arXiv:1802.03649*.

Marecek, J.; Richtarik, P.; and Takac, M. 2017. Matrix completion under interval uncertainty. *European Journal of Operational Research* 256(1):35–43.

Massicot, O., and Marecek, J. 2019. On-line non-convex constrained optimization. *arXiv preprint arXiv:1909.07492*.

Meng, D., and Torre, F. D. L. 2013. Robust matrix factorization with unknown noise. In *Proceedings of the 2013 IEEE International Conference on Computer Vision*, ICCV '13, 1337–1344. Washington, DC, USA: IEEE Computer Society.

Nesterov, Y. 2012. Efficiency of coordinate descent methods on huge-scale optimization problems. *SIAM Journal on Optimization* 22(2):341–362.

Oreifej, O.; Li, X.; and Shah, M. 2013. Simultaneous video stabilization and moving object detection in turbulence. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35(2):450–462.

Rodriguez, P., and Wohlberg, B. 2013. Fast principal component pursuit via alternating minimization. In *2013 IEEE International Conference on Image Processing*, 69–73.

Sawhney, H. S., and Ayer, S. 1996. Compact representations of videos through dominant and multiple motion estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 18(8):814–830.

Sobral, A.; Bouwmans, T.; and Zahzah, E.-h. 2015. Lrslibrary: Low-rank and sparse tools for background modeling and subtraction in videos. In *Robust Low-Rank and Sparse Matrix Decomposition: Applications in Image and Video Processing*. CRC Press, Taylor and Francis Group.

Tang, Y.; Dall'Anese, E.; Bernstein, A.; and Low, S. 2018. Running primal-dual gradient method for time-varying nonconvex problems. *arXiv preprint arXiv:1812.00613*.

Vaswani, N., and Narayanamurthy, P. 2018. Static and dynamic robust PCA and matrix completion: A review. *Proceedings of the IEEE* 106(8):1359–1379.

Vaswani, N.; Bouwmans, T.; Javed, S.; and Narayanamurthy, P. 2018. Robust subspace learning: Robust PCA, robust subspace tracking, and robust subspace recovery. *IEEE Signal Processing Magazine* 35(4):32–55.

Vaswani, N.; Chi, Y.; and Bouwmans, T. 2018. Rethinking PCA for modern data sets: Theory, algorithms, and applications [scanning the issue]. *Proceedings of the IEEE* 106(8):1274–1276.

Yong, H.; Meng, D.; Zuo, W.; and Zhang, L. 2018. Robust online matrix factorization for dynamic background subtraction. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 40(7):1726–1740.