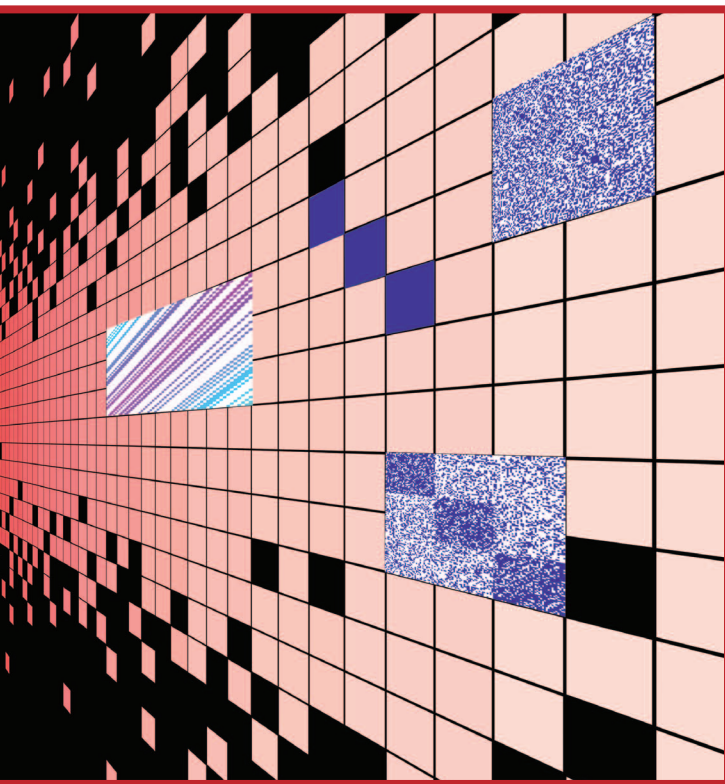Namrata Vaswani, Thierry Bouwmans,
Sajid Javed, and Praneeth Narayanamurthy

Robust Subspace Learning

Robust PCA, robust subspace tracking, and robust subspace recovery

©ISTOCKPHOTO.COM/SHUOSHU

Principal component analysis (PCA) is one of the most widely used dimension reduction techniques. A related easier problem is termed *subspace learning* or *subspace estimation*. Given relatively clean data, both are easily solved via singular value decomposition (SVD). The problem of subspace learning or PCA in the presence of outliers is called *robust subspace learning* (*RSL*) or *robust PCA* (*RPCA*). For long data sequences, if one tries to use a single lower-dimensional subspace to represent the data, the required subspace dimension may end up being quite large. For such data, a better model is to assume that it lies in a low-dimensional subspace that can change over time, albeit gradually. The problem of tracking such data (and the subspaces) while being robust to outliers is called *robust subspace tracking* (*RST*). This article provides a magazine-style overview of the entire field of RSL and tracking. In particular, solutions for three problems are discussed in detail:

- RPCA via sparse+low-rank (S+LR) matrix decomposition
- RST via S+LR
- robust subspace recovery (RSR).

RSR assumes that an entire data vector is either an outlier or an inlier. The S+LR formulation instead assumes that outliers occur on only a few data vector indices and, hence, are well modeled as sparse corruptions.

Introduction

PCA finds application in a variety of scientific and data analytics' problems ranging from exploratory data analysis and classification, e.g., image retrieval or face recognition, to a variety of modern applications such as video analytics, recommendation system design, and understanding social networks dynamics. PCA finds a small number of orthogonal basis vectors, called *principal components* (*PCs*), along which most of the variability of the data set lies. In most applications, only the subspace spanned by the PCs (principal subspace) is of interest. For example, this is all that is needed for dimension reduction. This easier problem is typically called *subspace learning* or *subspace estimation*: the two terms are sometimes used interchangeably. Given a matrix of clean data, PCA is easily

accomplished via SVD on the data matrix. The same problem becomes much harder if the data is corrupted by even a few outliers. The reason is that SVD is sensitive to outliers. We show an example of this in Figure 1. In today's big data age, since data is often acquired using a large number of inexpensive sensors, outliers are becoming even more common. They occur due to various reasons such as node or sensor failures, foreground occlusion of video sequences, or abnormalities or other anomalous behavior on certain nodes of a network. The harder problem of PCA or subspace learning for outlier corrupted data is called *RPCA or RSL*.

Since the term *outlier* does not have a precise mathematical meaning, the RPCA problem was, until recently, not well defined. Even so, many classical heuristics existed for solving it, e.g., see [1] and [2] and references therein. In recent years, there have been multiple attempts to qualify this term. Most popular among these is the idea of treating an outlier as an additive sparse corruption, which was popularized in the work of Wright and Ma [3]. This is a valid definition because it models the fact that outliers occur infrequently and allows them to have any magnitude. In particular, their magnitude can be much larger than that of the true data points. Using this definition, the recent work by Candès et al. [4] defined RPCA as the problem of decomposing a given data matrix, M, into the sum of a low-rank matrix, L, whose column subspace gives the PCs and a sparse matrix (outliers' matrix), S. This definition, which is often referred to as the *S+LR formulation*, has led to numerous interesting new works on RPCA solutions, many of which are provably correct under simple assumptions, e.g., [4]–[10]. A key motivating application is video analytics: decomposing a video into a slowly changing background video and a sparse foreground video [2], [4]; see an example in Figure 2(a). The background changes slowly, and the changes are usually dense (not sparse). It is thus well modeled as a dense vector that lies in the low-dimensional subspace of the original space [4]. The foreground usually consists of one or more moving objects, and is thus correctly modeled as the sparse outlier.

Often, for long data sequences, e.g., long surveillance videos, or long dynamic social network connectivity data sequences, if one tries to use a single lower-dimensional subspace to represent the data, the required subspace dimension may end up being quite large. For such data, a better model is to assume that it lies in a low-dimensional subspace that can change over time, albeit gradually. The problem of tracking a (slowly) changing subspace over time is often referred to as *subspace tracking* or *dynamic PCA*. The problem of tracking it in the presence of additive sparse outliers can thus be called either *RST* or *dynamic RPCA* [7], [11], [12].

Another way to interpret the word *outlier* is to assume that either an entire data vector is an outlier or it is an inlier. This a more appropriate model for outliers due to malicious users in recommendation system design or due to malicious participants in survey data analysis who enter all wrong answers in a survey

> **This article provides a magazine-style overview of the entire field of RSL and tracking.**

or in rating movies. For this problem setting, it is clearly impossible to recover the entire low-rank matrix L; it is only possible to correctly estimate its column subspace (and not the individual PCs). In modern literature, this problem is referred to as *RSR* [13].

Henceforth, the terms *RPCA* and *RST* refer only to the S+LR definitions while *RSR* is the aforementioned problem. See [14] for a detailed review of RPCA and RST via S+LR, and [15] for a review of RSR.

Two important extensions of RPCA are

- robust matrix completion (RMC) (or RPCA with missing data or low-rank matrix completion with sparse outliers) [16], [17]
- compressive or undersampled RPCA that involves RPCA from undersampled projected linear measurements [18]–[22].

This is useful in dynamic or functional magnetic resonance imaging (MRI) applications.

Applications
We describe some of the modern applications where the RPCA problem occurs.

Computer vision and video analytics
A large class of videos, e.g., surveillance videos, consists of a sparse foreground layer that contains one or more moving
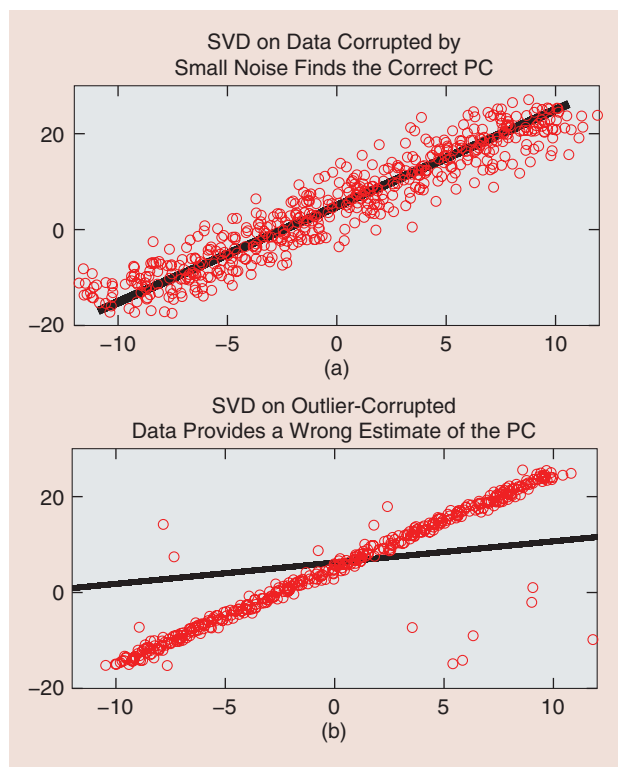

FIGURE 1. (a) PCA in small noise: the SVD solution works. The black line is the estimated PC computed using the observed data. (b) PCA in outliers: the SVD solution fails to correctly find the direction of largest variance of the true data. Instead its estimate is quite far from the true PC.

objects or people and a slow-changing background scene; see the example in Figure 2(a). Assume that all images are arranged as one-dimensional vectors. Then, the tth image forms the tth column, m_t, of the data matrix, M. If the image size is denoted by n and the total number of images in the video by d, then M is an $n \times d$ matrix. Assuming that the background scene changes slowly over time, the tth background image forms the tth column, ℓ_t, of the low-rank matrix L. Let r_L denote its rank. If the background never changes, then L will be rank-1. The low-rank assumption implies that the background changes depend on a much smaller number, r_L, of factors than either the number of images d or the image size n. Let $\mathcal{T}_t$ denote the support (set of indices of the nonzero pixels) of the foreground frame t. To get the $M = L + S$ formulation, we define s_t as a sparse vector with support $\mathcal{T}_t$ and with nonzero entries equal to the difference between foreground and background intensities on this support.

Being able to correctly solve the video-layering problem (separate a video into foreground and background layers) enables better solutions to many other video analytics and computer vision applications. For example, the foreground layer directly provides a video surveillance or object-tracking solution, while the background layer and its subspace estimate are useful in video editing or animation applications. Also, an easy low-bandwidth video conferencing solution would be to transmit only the layer of interest (usually the foreground). As another example, automatic video retrieval, used to look for videos of moving waters or other natural scenes, will become significant-ly easier if the retrieval algorithm is applied to only the background layer, whereas if the goal is to find a certain videos of a certain dog or cat breed, the algorithm can be applied to only the foreground layer. Finally, denoising and enhancement of very noisy videos becomes easier if the denoiser is applied to each layer separately (or to only the layer of interest). We show an example of this in Figure 3.

Dynamic and functional magnetic resonance imaging

The S+LR model is a good one for many dynamic MRI sequences. The changing region of interest (ROI) forms the sparse outlier for this problem while everything else that is slowly changing forms the low-rank component [22]. We show a cardiac sequence in Figure 2(b). The beating heart valves form the ROI in this case. This model can be used both to accurately recover a dynamic MRI sequence from undersampled data (solve the "compressive" MRI problem) and to correctly separate out the ROI. Similarly, in functional MRI-based brain activity imaging, only a sparse brain region is activated in response to given stimulus. This is the changing ROI (sparse outlier) for this problem. There is always some background brain activity in all the brain voxels. This is well modeled as being slowly changing and influenced by only a small number of factors, r_L.

Detecting anomalies in computer and social networks

Another application is in detecting anomalous connectivity patterns in social networks or in computer networks [23], [24].
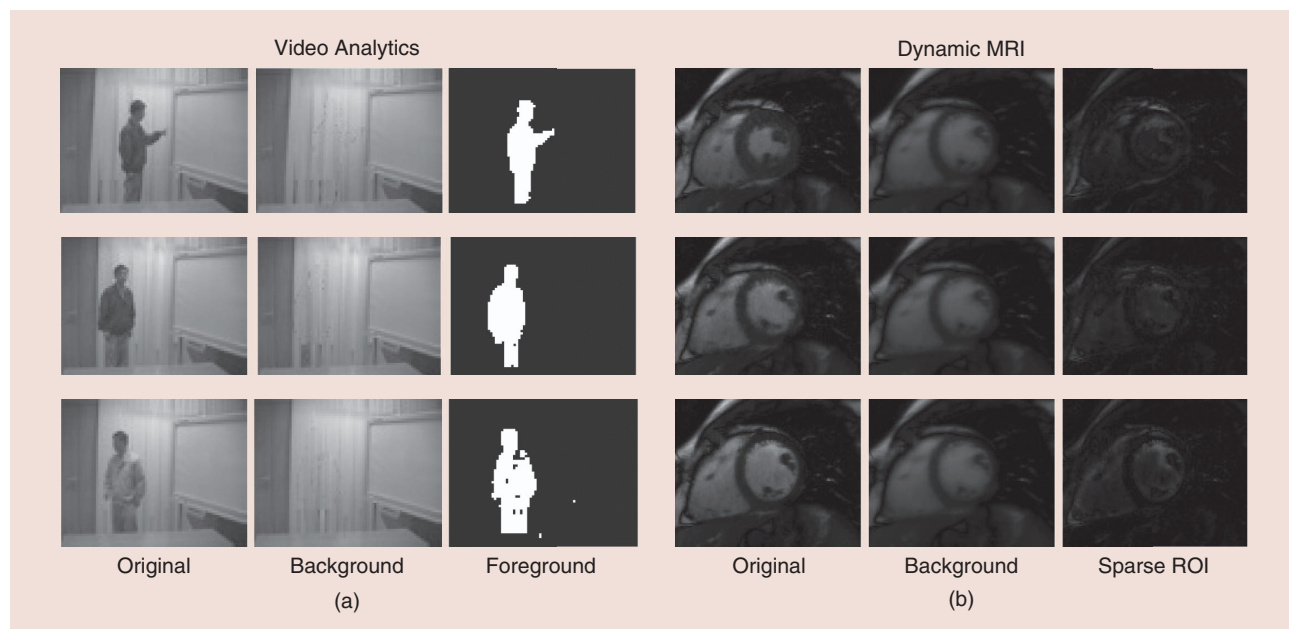

FIGURE 2. (a) A video analytics application. Video layering (foreground-background separation) in videos can be posed as an RPCA problem. This is often the first step to simplify many computer vision and video analytics tasks (for one such example, see Figure 3). We show three frames of a video in the first column. The background images for these frames are shown in the second column. Notice that they all look very similar and hence are well modeled as forming a low-rank matrix. The foreground support is shown in the third column. This clearly indicates that the foreground is sparse and changes faster than the background. (Result taken from [12]; code available at http://www.ece.iastate.edu/~hanguo/PracReProCS.html.) (b) Low-rank and sparse matrix decomposition for accelerated dynamic MRI [22]. The first column shows three frames of cardiac cine data. The second column shows the slow-changing background part of this sequence, while the third column shows the fast-changing sparse ROI. This is also called the *dynamic component*. These are the reconstructed columns obtained from eightfold undersampled data. They were reconstructed using undersampled stable PC pursuit [22].

This application is best solved using tensors rather than matrices; however, to explain the idea, we will just use the simpler matrix formulation. In this case, ℓ_t is the vector (or tensor) of network link "strengths" at time t when no anomalous behavior is present while s_t is the vector (or tensor) of outlier entries [24]. Outliers occur due to occasional anomalous behavior or due to node failures on a few edges. Hence, they are well modeled as being sparse.

Recommendation system design

The recommendation system design actually requires solving the RMC problem. To understand the problem, consider a specific example, say, the Netflix problem, which involves designing a movie recommendation system. Suppose there are n movies and d users. We use ℓ_t to denote the vector of true movie preferences of user t. The vector s_t would contain the outlier entries (if any). The matrix L is well modeled as being low rank under the assumption that user preferences are governed by only a few factors, r_L. The outliers s_t occur because some users enter some incorrect ratings due to laziness, malicious intent, or just typographical errors [4]. These are clearly sparse. The data matrix $M := L + S$ is also incomplete since any given user does not rate all movies. The goal in this case is to complete the true movie preferences' matrix L while being robust to the outliers. This is then used to recommend movies to the users.

Survey data analysis

A related application is in survey data analysis. Consider a survey with n questions given to d participants. If all participants answer questions truthfully and carefully (no malicious or typographical errors), then the survey answers of the tth participant form ℓ_t. This is a valid model under the assumption that the responses to all survey questions are governed by many fewer factors than n or d, and, hence, the resulting matrix formed by the true survey responses is low rank. The goal of the survey is, in fact, to find these few factors (PCs of this matrix). The problem becomes one of RPCA because some of the participants' entries will be wrong due to mistakes and typos. These are modeled as the sparse corruptions s_t. If the entries are wrong due to malicious participants, then the column-sparse model is a more appropriate one.

Desirable algorithm properties

In terms of algorithm design, the following are important questions to ask.

- *Provably correct and under what assumptions.* It is useful to know if an algorithm is guaranteed to be correct under simple and practically valid assumptions. We say that a guarantee is a complete correctness result or complete guarantee if it makes assumptions only on the algorithm input (the input data, parameters, and the initialization, if any) and proves that the algorithm output is within a small error of the true value of the quantity of interest. In all other situations, we say that the guarantee is a partial guarantee. This could mean many things: it could mean that the result makes assumptions on intermediate algorithm estimates; or that the guarantee only proves that the algorithm converges to a local minimum (or just to some stationary point) of a cost function. A partial guarantee by itself is usually not very useful. However, sometimes the proof technique used for obtaining it may be useful and may help obtain a complete correctness result in later work.
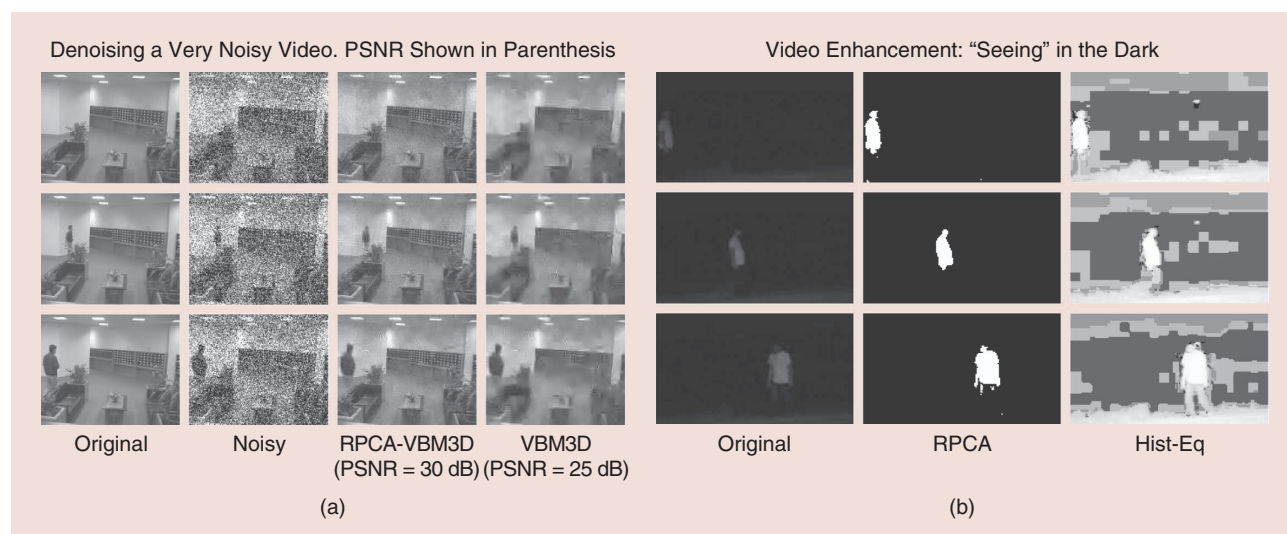

FIGURE 3. RPCA to enable video enhancement and denoising [12], [25]. (a) Denoising a very noisy video [video with Gaussian noise of standard deviation $\sigma = 70$ and hence peak signal-to-noise ratio (PSNR) 11 dB]: we show original and noisy videos in first two columns. The noisy video had PSNR 11 dB. RPCA-VM3D uses an RPCA algorithm (ReProCS) to first separate it into sparse and low-rank layers, and then apply the best denoiser from literature, VBM3D, on each layer. The last column directly applies VBM3D to the video. Notice that this gives a much more blurred denoised image. PSNR is noted below each figure too. (b) Video enhancement via RPCA: an application where an RPCA algorithm (ReProCS) helps to "see" in the dark. A standard approach for such low-light data is to do contrast enhancement using histogram equalization. As can be seen from the third column (Hist-Eq), this fails. The code for this experiment is downloadable from http://www.ece.iastate.edu/~hanguo/ReLD_Denoising.zip.

- *Practical accuracy.* Performance guarantees are often just sufficient conditions. Also the assumptions required can be hard to verify in practice. Hence, practical accuracy is an equally important measure of algorithm success. Commonly used metrics for RPCA include normalized recovery error of L or the error in recovering its column span. If the outlier matrix S is the "quantity" of interest, e.g., in case of foreground object tracking, then the error in recovering its support is often a more useful metric.

- *Time complexity.* Both order-wise computational complexity and actual experimental comparisons of time taken are important. The former is useful because it can take too long to do experimental comparisons for very large-sized problems. The latter is useful because order-wise complexity ignores constants that can sometimes be very large in practice. We discuss both types of time comparisons in this article.

- *Memory complexity and/or number of passes.* In today's big data age, this is often the most important concern. It can be the bottleneck even for applications for which real-time output is not needed and, hence, slow algorithms can be tolerated. We say that an algorithm is *memory optimal* if it needs the same order of memory as the algorithm output. So, for RPCA, an algorithm would be memory optimal if it needed memory of order $O(nr)$. It is nearly memory optimal if its memory complexity is within logarithmic factors of the optimal. If the data set needs to be loaded from a storage device while running the algorithm, another related concern is the number of passes required. This number then governs the number of times the data needs to be reloaded. For processing very large data sets, one needs algorithms that are either nearly memory optimal or need few passes through the data, ideally both.

- *Online (or recursive or causal) versus batch.* Online algorithms are algorithms for settings where inputs or data are arriving one at a time, and we need to make decisions on the fly or with small delays, without knowing what will happen in the future. In signal processing or controls literature, such an algorithm would be called *recursive*. The notion of an online algorithm also requires that the algorithm output quality improves as more data come in. For RPCA, this means that the subspace estimate quality improves over time. A batch method, on the other hand, needs to wait for all data to come in first.

An overview

This section provides an overview of the problem setting and key solutions for all the three problems discussed in this article along with three sample theoretical guarantees. It is a quick summary for the reader who is already somewhat familiar with the topic. In the two later sections "Robust Principal

> **Being able to correctly solve the video-layering problem (separate a video into foreground and background layers) enables better solutions to many other video analytics and computer vision applications.**

Component Analysis and Robust Subspace Tracking via Sparse+Low-Rank: Problem Setting" and "Robust Subspace Recovery" sections, we explain the solution approaches in detail. The code for all of the RPCA and RST solutions is available via the GitHub library of Andrew Sobral [26]: https://github.com/andrewssobral/lrslibrary.

In our guarantees and elsewhere, we use the letters C, c to denote different numerical constants in each use. To keep our statements simple, condition numbers are treated as numerical constants (and combined with C or c).

Robust principal component analysis and robust subspace tracking via sparse+low-rank: Problem setting

Static robust principal component analysis [4]
Given an $n \times d$ observed data matrix,

$$M := L + S + W,$$

where L is a low rank matrix (true data), S is a sparse matrix (outlier matrix), and W is small unstructured noise/modeling error, the goal is to estimate L and, hence, its column space. We use r_L to denote the rank of L. The maximum fraction of nonzeros in any row (column) of the outlier matrix S is denoted by max-outlier-frac-row (max-outlier-frac-col). The column space of $\hat{L}$ then gives the PCA solution.

Robust subspace tracking or dynamic robust principal component analysis [7]
At each time t, we get a data vector $m_t \in \mathbb{R}^n$ that satisfies

$$m_t := \ell_t + s_t + w_t, \text{ for } t = 1, 2, \ldots, d,$$

where w_t is small unstructured noise, s_t is the sparse outlier vector, and ℓ_t is the true data vector that lies in a fixed or slowly changing low-dimensional subspace of $\mathbb{R}^n$, i.e., $\ell_t = P_{(t)} a_t$, where $P_{(t)}$ is an $n \times r$ basis matrix (a matrix with mutually orthonormal columns) with $r \ll n$ and with $\left\| (I - P_{(t-1)} P_{(t-1)}') P_{(t)} \right\|$, small compared to $\left\| P_{(t)} \right\| = 1$. We use $\mathcal{T}_t$ to denote the support set of s_t. Given an initial subspace estimate, $\hat{P}_0$, the goal is to track $\text{span}(P_{(t)})$ and ℓ_t either immediately or within a short delay. A by-product is that ℓ_t, s_t, and $\mathcal{T}_t$ can also be tracked on the fly. The initial subspace estimate, $\hat{P}_0$, can be computed by using only a few iterations of any of the static RPCA solutions (described next), e.g., PCP [4] or alternating-projection (AltProj) [8], applied to the first t_{train} data samples $M_{[1, t_{\text{train}}]}$. Typically, $t_{\text{train}} = Cr$ suffices. Here, $[a, b]$ refers to all integers between a and b, inclusive, $[a, b) := [a, b - 1]$, and $M_{\mathcal{T}}$ denotes a submatrix of M formed by its columns indexed by entries in the set $\mathcal{T}$. Alternatively, in some applications, e.g., video surveillance, it is valid to assume that outlier-free data are available. In these situations, simple SVD can also be used.

Technically, dynamic RPCA is the offline version of the aforementioned problem. Define matrices L, S, W, M with $L = [\ell_1, \ell_2, \ldots \ell_d]$ and M, S, W similarly defined. The goal is to recover the matrix L and its column space with ϵ error.

We use $SE(\hat{P}, P) := \| (I - \hat{P}\hat{P}')P \|$ to measure the subspace error (distance) between the subspaces spanned by the columns of P and $\hat{P}$. This measures the sine of the maximum principal angle between the two subspaces. It is symmetric when the two subspaces have equal dimension. Here and elsewhere $\| \cdot \|$ denotes the induced l_2 norm and $'$ denotes matrix transpose.

Identifiability and other assumptions

The previously given problem definitions do not ensure identifiability since either L or S can be both low rank and sparse. For the unfamiliar reader, we explain this point in detail in the section "Understanding Identifiability Issues." One way to ensure that L is not sparse is by requiring that its left and right singular vectors are dense or "incoherent" with respect to a sparse vector [4].

Definition 2.1 (μ-incoherence/denseness)

We say that an $n \times r$ basis matrix (a matrix with mutually orthonormal columns) P is μ-incoherent if

$$\max_{i = 1, 2, \ldots, n} \| P^{(i)} \|^2 \leq \mu \frac{r_L}{n},$$

where $P^{(i)}$ denotes the ith row of P and $\mu \geq 1$ is the (in)coherence parameter. It quantifies the nondenseness of P.

One way to ensure that S is not low rank is by imposing upper bounds on max-outlier-frac-row and max-outlier-frac-col.

Consider the RST problem. If the r-dimensional subspaces change at each time, there are too many unknown parameters making the subspace tracking problem unidentifiable. One way to ensure identifiability is to assume that they are piecewise constant [7], [27], i.e., that

$$P_{(t)} = P_j \text{ for all } t \in [t_j, t_{j+1}), \quad j = 1, 2, \ldots, J,$$

with $t_0 = 1$ and $t_{J+1} = d$, and to lower bound $t_{j+1} - t_j$ by a number more than r. With this model, $r_L = rJ$ in general (except if subspace directions are repeated more than once).

Left and right incoherence of L and outlier fraction bounds on S are again needed. The union of the column spans of all the P_js is equal to the span of the left singular vectors of L. Thus, left incoherence is equivalent to assuming that the P_js are μ-incoherent. One can replace right incoherence by an independent identically distributed assumption on the a_ts, along with boundedness of each element of a_t. As explained in [27], the two assumptions are similar. The latter is more suited for RST since it involves online estimation. Since RST algorithms are online and often operate on mini-batches of data, we also

> **One way to ensure that L is not sparse is by requiring that its left and right singular vectors are dense or "incoherent" with respect to a sparse vector.**

need to redefine max-outlier-frac-row as the maximum nonzero fraction per row in any α-consecutive-column submatrix of S. We refer to it as *max-outlier-frac-row$^\alpha$*. Here α is the mini-batch size used by the RST algorithm.

Lastly, as we will see, if one also assumes a lower bound on outlier magnitudes (uses the model that outliers are large magnitude corruptions), it can help significantly relax the required upper bound on max-outlier-frac-row$^\alpha$, while also allowing for a fast, online, and memory-efficient algorithm.

Static robust principal component analysis: A summary

The first provably correct and polynomial complexity solution to RPCA was introduced in [4]. It involves solving a convex program that they called *PC pursuit* (*PCP*). The same program was introduced and studied in parallel [5] and later work [6] as an S+LR solution. While polynomial complexity is better than exponential, it is still too slow for today's big data sets. Moreover, the number of iterations needed for a convex program solver to get to within an ϵ ball of the true solution of the convex program is $O(1/\epsilon)$, and thus the typical complexity for a PCP solver is $O(nd^2/\epsilon)$ [8]. To address this issue, a provably correct alternating minimization (alt-min) solution called *Alt-Proj* was introduced in [8]. This has much lower time complexity and still works under the same assumptions as PCP. The following holds [8].

Theorem 2.2 (Alternating projection guarantee for robust principal component analysis)

Let $L \overset{\text{SVD}}{=} U\Sigma V'$. If 1) U, V are μ-incoherent, 2) max (max-outlier-frac-row, max-outlier-frac-col) $\leq \frac{c}{\mu r_L}$, 3) $\| W \|_F^2 \leq C\epsilon^2$, and 4) algorithm parameters are appropriately set, then AltProj returns $\hat{L}, \hat{S}$ with $\| \hat{L} - L \|_F \leq \epsilon$, and $\| \hat{S} - S \|_{\max} \leq \epsilon/\sqrt{md}$ in time $O(ndr_L^2 \log(1/\epsilon))$.

An even faster solution that relied on projected gradient descent (GD), called *RPCA-GD*, was proposed in [10]. If condition numbers are treated as constants, its complexity is only $O(ndr_L \log(1/\epsilon))$. This is comparable to that of vanilla r-SVD for simple PCA; however, this approach needs outlier fraction bounds to be $\sqrt{r_L}$ times tighter. Another algorithm that relies on the recursive projected compressive sensing (CS) (ReProCS) approach and has the same time complexity as RPCA-GD is called *nearly optimal RST via ReProCS* (*ReProCS-NORST*) [27], [28]. This is also online (after initialization); has much lower, and nearly optimal, memory complexity of $O(nr_L \log n \log(1/\epsilon))$; and has outlier tolerance that is better than even AltProj. But it needs two extra assumptions: lower bound on most outlier magnitudes, and fixed or slow subspace change. For a similar setting, a recent RMC solution, called *nearly optimal RMC* (*NO-RMC*), is also applicable [17]. It is the fastest RPCA solution with a time complexity of just $O(nr_L^2 \log^2 n \log^2(1/\epsilon))$, but it needs $d \approx n$. As we explain later, this is a strong assumption for video data. It achieves this speed-up by deliberately undersampling M and solving RMC.

Algorithm	Outlier Tolerance, Rank of (L)	Assumptions	Memory, Time,	# of Parameters
PCP(C) [4] (offline)	max-outlier-frac-row = $O(1)$ max-outlier-frac-col = $O(1)$ $r_L \leq \dfrac{c \min(n,d)}{\log^2 n}$	Uniform random outlier support	Mem: $O(nd)$ Time: $O\left(nd^2 \dfrac{1}{\epsilon}\right)$	Zero
AltProj [8] (offline)	max-outlier-frac-row = $O(1/r_L)$ max-outlier-frac-col = $O(1/r_L)$		Mem: $O(nd)$ Time: $O\left(ndr_L^2 \log \dfrac{1}{\epsilon}\right)$	Two
RPCA-GD [10] (offline)	max-outlier-frac-row = $O(1/r_L^{1.5})$ max-outlier-frac-col = $O(1/r_L^{1.5})$		Mem: $O(nd)$ Time: $O\left(ndr_L \log \dfrac{1}{\epsilon}\right)$	Five
PG-RMC [17] (offline)	max-outlier-frac-row = $O(1/r_L)$ max-outlier-frac-col = $O(1/r_L)$	$d \approx n$	Mem: $O(nd)$ Time: $O\left(nr_L^3 \log^2 n \log^2 \dfrac{1}{\epsilon}\right)$	Four
ReProCS-NORST [27], [28] (online) detects and tracks subspace change	max-outlier-frac-row = $O(1)$ max-outlier-frac-col = $O(1/r_L)$	Outlier magnitude lower bounded slow subspace change or fixed subspace first Cr samples: AltProj assumptions	Mem: $O\left(nr_L \log n \log \dfrac{1}{\epsilon}\right)$ Time: $O\left(ndr_L \log \dfrac{1}{\epsilon}\right)$ Track delay: $O\left(r \log n \log \dfrac{1}{\epsilon}\right)$	Four

We compare the theoretical guarantees for these solutions in Tables 1 and 2 and experimental performance (for video analytics) in Table 3. As can be seen, among the provable solutions, ReProCS has the best experimental performance and is the fastest. AltProj is about three times slower. GD and NO-RMC are, in fact, slower than AltProj.

Robust subspace tracking: A summary
In [7], [9], [12], [27], and [29], the novel online solution framework ReProCS was introduced to solve the RST or the dynamic RPCA problem. The best ReProCS guarantee is given next. This is for a ReProCS-based algorithm that we call *ReProCS-NORST* because its tracking delay is nearly optimal [27], [28]. At each time, it outputs $\hat{P}_{(t)}, \hat{\ell}_t, \hat{s}_t, \hat{\mathcal{T}}_t$. It also outputs $\hat{t}_j$ as the time at which it detects the jth subspace change.

Theorem 2.3 (Recursive projected compressive sensing—nearly optimal robust subspace tracking guarantee for robust subspace tracking)
Let $\alpha := Cr \log n$, $\Lambda := \mathbb{E}[a_1 a_1']$, and $\lambda^+ := \lambda_{\max}(\Lambda)$, $\lambda^- := \lambda_{\min}(\Lambda)$, and let $s_{\min} := \min_t \min_{i \in \mathcal{T}_t}(s_t)_i$ denote the minimum outlier magnitude. Pick an $\epsilon \leq \min(0.01, 0.4 \min_j \text{SE}(P_{j-1}, P_j)^2 / f)$. If
1) P_js are μ-incoherent; and a_ts are zero mean, mutually independent over time t, have identical covariance matrices, i.e., $\mathbb{E}[a_t a_t'] = \Lambda$, are element-wise uncorrelated (Λ is diagonal), are element-wise bounded (for a numerical constant η, $(a_t)_i^2 \leq \eta \lambda_i(\Lambda)$), and are independent of all outlier supports $\mathcal{T}_t$.
2) $\|w_t\|^2 \leq cr \|\mathbb{E}[w_t w_t']\|$, $\|\mathbb{E}[w_t w_t']\| \leq c\epsilon^2 \lambda^-$, zero mean, mutually independent, independent of s_t, ℓ_t
3) max-outlier-frac-col $\leq c_1 / \mu r$, max-outlier-frac-row$^\alpha \leq c_2$

4) subspace change: let $\Delta := \max_j \text{SE}(P_{j-1}, P_j)$
   a) $t_{j+1} - t_j > Cr \log n \log(1/\epsilon)$
   b) $\Delta \leq 0.8$ *and* $C_1 \sqrt{r \lambda^+} (\Delta + 2\epsilon) \leq s_{\min}$
5) *init*: $\text{SE}(\hat{P}_0, P_0) \leq 0.25$, $C_1 \sqrt{r \lambda^+} \text{SE}(\hat{P}_0, P_0) \leq s_{\min}$
6) algorithm parameters are appropriately set; then, with probability (w.p.) $\geq 1 - 10dn^{-10}$, $\text{SE}(\hat{P}_{(t)}, P_{(t)}) \leq$

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

where $K := C \log(1/\epsilon)$. Memory complexity is $O(nr \log n \log(1/\epsilon))$ and time complexity is $O(ndr \log(1/\epsilon))$.

In the statement of Theorem 2.3, the condition number, $f := \lambda^+ / \lambda^-$ is treated as a numerical constant and ignored. Under the assumptions of the theorem, the following also hold:
1) $\|\hat{s}_t - s_t\| = \|\hat{\ell}_t - \ell_t\| \leq 1.2(\text{SE}(\hat{P}_{(t)}, P_{(t)}) + \epsilon)\|\ell_t\|$
2) at all times, t, $\hat{\mathcal{T}}_t = \mathcal{T}_t$
3) $t_j \leq \hat{t}_j \leq t_j + 2\alpha$
4) Offline-NORST: $\text{SE}(\hat{P}_{(t)}^{\text{off}}, P_{(t)}) \leq \epsilon$, $\|\hat{s}_t^{\text{off}} - s_t\| = \|\hat{\ell}_t^{\text{off}} - \ell_t\| \leq \epsilon\|\ell_t\|$ at all t.

Observe that ReProCS-NORST can automatically detect and track subspace changes with delay that is only slightly more than r (near optimal). Also, its memory complexity is within log factors of nr (memory needed to output the subspace estimate). The second key point is that, after the first Cr data samples (used for initialization) for α-sample mini-batches of data, it tolerates a constant maximum fraction of outliers per row without any assumption on outlier support. This is possible because it makes two other extra assumptions: fixed or slow subspace change and a lower bound on $s_{\min}$ (see last two assumptions of Theorem 2.3).

Table 2. Comparing all RPCA and RST solutions—both with and without guarantees. The methods are chronologically ordered.

Algorithm	Outlier Tolerance and Rank of L or How Well It Works in Practice	Assumptions if Guarantee Exists	Memory, Time Complexity, or Practical Speed	Mode (# of Parameters)
RSL [2]	Works well	No guarantee	Slower	Batch (zero)
i-RSL [35], [36]	Never works	No guarantee	Slower	Online (zero)
PCP(C) [4]	max-outlier-frac-row $\in O(1)$ max-outlier-frac-col $\in O(1)$ $r_L \leq \dfrac{c\min(n,d)}{\log^2 n}$	Uniform random support, Strong incoherence	Memory: $O(nd)$ Time: $O(nd^2/\epsilon)$	Batch (zero)
PCP(H) [5], [6],	max-outlier-frac-row $\in O(1/r_L)$ max-outlier-frac-col $\in O(1/r_L)$	No extra assumption	Memory: $O(nd)$ Time: $O(nd^2/\epsilon)$	Batch (two)
ReProCS [11], [12]	Works well	Partial guarantee	Fast	Online (four)
GRASTA [50]	Works	No guarantee	Faster	Online (zero)
Bilinear-Dec [46]	Never works	Partial guarantee		Online (one)
ORPCA [47]	Works well	Partial guarantee	Faster	Online (three)
GoDec [37]	Works	Partial guarantee	Fast	Batch (three)
pRoST [51]	Works	No guarantee	Slower	Online (two)
AltProj [8],	max-outlier-frac-row $\in O(1/r_L)$ max-outlier-frac-col $\in O(1/r_L)$	No extra assumption	Memory: $O(nd)$ Time: $O\left(ndr_L^2\log\frac{1}{\epsilon}\right)$	Batch (two)
Original-ReProCS [9], [44]	max-outlier-frac-row $\in O(1)$ max-outlier-frac-col $\in O(1/r)$	ReProCS-NORST assumptions, and outlier support change model, slow subspace change (strong)	Memory: $O(nr\log n/\epsilon^2)$ Time: $O\left(ndr\log\frac{1}{\epsilon}\right)$	Online (four)
Modified-PCP [30]	max-outlier-frac-row $\in O(1)$ max-outlier-frac-col $\in O(1)$ $r_L \leq \dfrac{c\min(n,d)}{\log^2 n}$	Uniform random support Strong incoherence (weaker) Slow subspace change (strong) Initial data: AltProj assumption	Memory: $O(nr\log^2 n)$ Time: $O\left(ndr\log^2 n\frac{1}{\epsilon}\right)$	Online (one)
APSM [49]		Partial guarantee		Online (three)
RPCA-GD [10]	max-outlier-frac-row $\in O(1/r_L^{1.5})$ max-outlier-frac-col $\in O(1/r_L^{1.5})$	No extra assumption	Memory: $O(nd)$ Time: $O\left(ndr_L\log\frac{1}{\epsilon}\right)$	Batch (five)
NO-RMC [17]	max-outlier-frac-row $\in O(1/r_L)$ max-outlier-frac-col $\in O(1/r_L)$	$d \approx n$	Memory: $O(nd)$ Time: $O\left(nr_L^3\log n\log\frac{1}{\epsilon}\right)$	Batch (five)
ReProCS-NORST and simple-ReProCS [27], [29]	max-outlier-frac-row $\in O(1)$ max-outlier-frac-col $\in O(1/r)$	Outlier magnitude lower bounded, Slow subspace change (mild) First Cr samples: AltProj assumptions	Memory: $O\left(nr\log n\log\frac{1}{\epsilon}\right)$ Time: $O\left(ndr\log\frac{1}{\epsilon}\right)$	Online (four)

Note: For algorithms that come without guarantees or with partial guarantees, we specify this in the "Assumptions" column. For all these algorithms in the "Outlier Tolerance" column, we list "works," "works well" or "never works." This information is based on the comparisons shown in [12] (for iRSL and bilinear-Dec) or in the "Experiments" section. In the "Memory, Time Complexity, or Practical Speed" column, we list "fast," "faster," or "slower." This information is again based on experiments described in the "Experiments" section. The dynamic RPCA solutions assume subspace at any time has dimension r and there are a total of J subspace changes. Thus $r_L = rJ$. Expansions of all algorithm acronyms are provided in the article text.

As long as the initial subspace estimate is accurate enough and the subspace changes slowly enough so that both Δ and SE($\hat{P}_0, P_0$) are $O(1/\sqrt{r})$, the lower bound just requires that $s_{\min}$ be of the order of $\sqrt{\lambda^+}$ or larger, which is reasonable. Moreover, as explained later in Remark 3.5, this can be relaxed further. The requirement on Δ and SE($\hat{P}_0, P_0$) is not restrictive either, because SE(.) is only measuring the largest principal angle. It still allows the chordal distance between the two subspaces (l_2 norm of the vector containing the sine of all principal angles) to be $O(1)$.

Slow subspace change is a natural assumption for static camera videos (with no sudden scene changes). The outlier lower bound is a mild requirement because, by definition, an *outlier* is a large magnitude corruption. The small magnitude

Table 3. The average F_1 score using the CDnet data set for the comparison of provable, heuristics, and heuristics with specific constraints methods for background subtraction. Time is shown for a video having 320 × 240 resolution of 8,000 frames. The best and second-best performing methods are shown in red and blue, respectively.

Provable Methods	Baseline	DB	Camera Jitter	Shadow	Thermal	IOM	Average	Time (s/frame)
PCP (batch); Figure 5(c); [4], [5]	0.75	0.69	0.62	0.73	0.65	0.48	0.65	4.19
AltProj (batch); [8]	**0.78**	**0.71**	0.60	**0.76**	0.69	0.58	**0.68**	2.38
NO-RMC (batch); Figure 5(o); [17]	0.71	0.64	0.64	0.66	**0.71**	0.50	0.64	2.85
RPCA-GD (batch); Figure 5(f); [10]	0.74	0.62	0.68	**0.75**	0.66	0.49	0.65	2.46
Simple-ReProCS (online); [9], [29]	**0.77**	**0.77**	**0.69**	0.71	**0.74**	**0.70**	**0.73**	**0.74**
Mod-PCP (online); [30]	0.75	0.64	**0.70**	0.65	0.69	**0.70**	**0.68**	**0.44**
Heuristics Methods	**Baseline**	**DB**	**Camera Jitter**	**Shadow**	**Thermal**	**IOM**	**Average**	**Time**
GRASTA (online); Figure 5(k); [50]	0.66	0.35	0.43	0.52	0.42	0.35	0.45	1.16
3TD (batch); Figure 5(g); [62]	**0.88**	0.75	0.72	0.68	**0.78**	0.55	0.72	2.17
2PRPCA (batch); Figure 5(u); [63]	**0.92**	**0.79**	**0.81**	**0.80**	0.76	**0.65**	**0.78**	1.63
GoDec (batch); Figure 5(d); [37]	0.77	0.58	0.48	0.51	0.62	0.38	0.55	1.56
OR-PCA (online); [47]	0.62	0.45	0.36	0.52	0.66	0.59	0.53	0.17
pROST (online); Figure 5(h); [51]	0.79	0.59	0.79	0.70	0.58	0.48	0.65	2.03
Probabilistic robust matrix factorization; Figure 5(e); [64]	**0.92**	0.77	**0.85**	**0.88**	**0.83**	0.48	**0.78**	2.40
Heuristics Methods with Specific Constraints	**Baseline**	**DB**	**Camera Jitter**	**Shadow**	**Thermal**	**IOM**	**Average**	**Time**
modified-ReProCS (online); Figure 5(l); [12]	0.80	0.76	0.72	0.75	0.77	**0.69**	**0.74**	**0.61**
incPCP (online); Figure 5(i); [52]	0.81	0.71	0.78	0.74	0.70	**0.75**	0.74	**0.41**
OR-PCA-illum (online); Figure 5(t); [74]	0.86	0.75	0.70	0.74	0.76	0.56	0.72	**0.22**
MSCL (batch); Figure 5(x); [65]	0.87	**0.85**	**0.83**	0.82	**0.82**	**0.80**	**0.83**	1.68
DECOLOR (batch); Figure 5(v); [66]	**0.92**	0.70	0.68	**0.83**	0.70	0.59	0.73	1.88
LSD (batch); Figure 5(p); [67]	**0.92**	0.71	0.78	0.81	0.75	0.67	0.77	1.43
TVRPCA (batch); Figure 5(m); [68]	0.84	0.55	0.63	0.71	0.69	0.57	0.66	1.48
SRPCA (batch); Figure 5(n); [69]	0.82	0.84	0.78	0.77	0.79	**0.80**	0.80	**0.59**
RMAMR (batch); Figure 5(j); [70]	0.89	0.82	0.75	0.73	0.75	0.66	0.76	1.32
GFL (batch); Figure 5(w); [71]	0.83	0.74	0.78	0.82	0.76	0.59	0.75	2.40
GOSUS (online); Figure 5(q); [53]	**0.90**	0.79	**0.82**	**0.84**	**0.80**	0.74	**0.81**	0.89
COROLA (online); Figure 5(s); [72]	0.85	**0.86**	**0.82**	0.78	**0.80**	0.71	0.80	**0.39**
OMoG+TV (online); Figure 5(r); [73]	0.85	0.76	0.78	0.68	0.70	0.71	0.74	**0.19**
Non-RPCA Methods	**Baseline**	**DB**	**Camera Jitter**	**Shadow**	**Thermal**	**IOM**	**Average**	**Time**
PAWCS; [75]	0.88	0.85	0.78	0.86	0.80	0.74	0.81	2.82
SuBSENSE; [76]	0.91	0.78	0.78	0.88	0.78	0.63	0.79	1.89
LOBSTER; [77]	0.87	0.74	0.80	0.72	0.79	0.72	0.75	3.7

ones get classified as small noise w_t. Moreover, as we explain later (see Remark 3.5), the requirement can be relaxed further. The looser requirement on outlier fractions per row means that ReProCS can tolerate slow moving or occasionally static objects better than other approaches (does not confuse these for the background).

Another approach that can solve RST is modified-PCP [30]. This was designed as a solution to the problem of RPCA with partial subspace knowledge [30]. It solves RST by using the previous subspace estimate as the "partial subspace knowledge."

Dynamic versus static robust principal component analysis
While RST or dynamic RPCA is a different problem than static RPCA (it assumes a time-varying subspace and slow subspace change), from the point of view of applications, especially those for which a natural time sequence exists, e.g., video analytics for videos from static camera(s) or dynamic MRI-based

ROI tracking [22], both models are equally applicable. For long videos from static cameras, dynamic is generally a better model to use. The same is true for videos involving slow-moving or occasionally static foreground objects since these result in a larger fraction of outliers per row of the data matrix. On the other hand, for videos involving sudden scene changes, the original RPCA problem is a better fit, since in this case the slow subspace change assumption absolutely does not hold.

For recommendation system design or survey data analysis, if the initial data set has a certain number of users, but as time goes on, more users get added to the system, the following strategy is best. Use a static RPCA solution on the initial data set. As more users get added, use an RST solution to both update the solution and detect when subspace changes occur. In both applications, subspace change would occur when the survey reaches a different previously unrepresented demographic.

Robust subspace recovery: Problem and summary

There is a long body of both old and recent work that studies the following problem: given a set of d points in $\mathbb{R}^n$, suppose that the inlier points lie in an unknown low-dimensional subspace, while outlier points are all points that do not. Thus, an entire data vector is either an inlier or an outlier. Also suppose that the fraction of outliers is less than 50% of all data points. Then, how does one find the low-dimensional subspace spanned by the inliers? In recent literature, this body of work has been referred to as *mean absolute deviation rounding* (*MDR*), [31] *outlier-RPCA* [32], *PCA with contaminated data* [33], or, more recently, *RSR* [13]. This is a harder problem to solve especially when the fraction of outliers is large since, in this case, entire data vectors get classified as outliers and are thrown away.

While many algorithms have been proposed in recent literature, complete guarantees under simple assumptions are hard to obtain. An exception is the outlier pursuit solution [33]. It solves this problem by using a nice extension of the PCP idea: it models the outliers as additive "column-sparse corruptions." Xu et al. [33] prove the following.

Theorem 2.4 (Outlier pursuit guarantee for robust subspace recovery)

Let γ denote the fraction of corrupted columns (outlier fraction). Outlier pursuit correctly recovers the column space of L and correctly identifies the outlier support if
- $\gamma \in O(1/r)$
- the matrix L is column-incoherent, i.e., the row norm of the matrix of its right singular vectors is bounded by $\mu r / ((1 - \gamma) n)$.

This is a nice and simple correctness guarantee, but it requires a tight bound on outlier fractions (just like the S+LR literature). Most other guarantees (described later) only prove asymptotic convergence of the iterates of the algorithm to either a stationary point or a local minimum of the chosen cost

> **Among the provable solutions, ReProCS has the best experimental performance and is the fastest. AltProj is about three times slower. GD and NO-RMC are, in fact, slower than AltProj.**

function; and/or lower bound some performance metric for the solution, e.g., explained variance or mean absolute deviation (MAD).

Robust principal component analysis and subspace tracking via sparse+low-rank matrix decomposition

Understanding identifiability issues

The S+LR formulation by itself does not ensure identifiability. For example, it is impossible to correctly separate M into $L + S$ if L is also sufficiently sparse in addition to being low rank or if S is also sufficiently low rank in addition to being sparse. In particular, there is a problem if S has rank that is equal to or lower than that of L or if L has support size that is smaller than or equal to that of S.

For example, if S is sparse with support that never changes across columns, then S will be a matrix with rank at most s (where s is the column support size). If, in addition, the non-zero entries of s_t are the same for all t (all columns), then, in fact, S will be a rank-1 matrix. In this case, without extra assumptions, there is no way to correctly identify S and L. This situation occurs in case of a video with a foreground object that never moves and whose pixel intensities are such that s_t never changes. As a less extreme example, if the object remains static for $b_0 d$ frames at a time before moving, and its pixel intensities are such that s_t itself is also constant for these many frames, then the rank of S will be $d/(b_0 d) = 1/b_0$. If this number is less than r_L, then, again, the problem is not identifiable without extra assumptions. Observe that, in this example, max-outlier-frac-row equals b_0. Thus max-outlier-frac-row needs to be less than $1/r_L$ for identifiability. By a similar argument, max-outlier-frac-col needs the same bound.

The opposite problem occurs if the left or right singular vectors of L are sparse. If the left singular vectors are sparse, L will be a row sparse matrix (some rows will be zero). If the right singular vectors are sparse, then it will be a column sparse matrix. If both left and singular vectors are sparse, then L will be a general S+LR matrix. In all these situations, if the singular vectors are sparse enough, it can happen that L is sparser than S, and then it becomes unidentifiable.

It is possible, though, to impose simple assumptions to ensure that neither of the aforementioned situations occur. We can ensure that S is not lower rank than L by requiring that the fraction of outliers in each row and each column be at most c/r_L with $c < 1$ [5], [6], [8]. Alternatively, one can assume that the outlier support is generated from a uniform distribution; with this assumption, just a bound on its size suffices [4].

We can ensure that L is not sparse by requiring that its left and right singular vectors be dense. This is called the *denseness* or, more generally, the *incoherence* assumption. The term *incoherence* is used to imply that this condition helps ensure that the matrix L is different enough from the sparse matrix

S (in the sense that the normalized inner product between the two matrices is not too large). Let $\boldsymbol{L} \stackrel{\text{SVD}}{=} \boldsymbol{U}\Sigma\boldsymbol{V}'$ be the reduced (rank-r_L) SVD of $\boldsymbol{L}$. Since the left and right singular vectors, $\boldsymbol{U}$ and $\boldsymbol{V}$, have unit Euclidean norm columns, the simplest way to ensure that the columns are dense (nonsparse) is to assume that the magnitude of each entry of $\boldsymbol{U}$ and $\boldsymbol{V}$ is small enough. In the best (most dense) case, all their entries would be equal and just differ in signs. Thus, all entries of $\boldsymbol{U}$ will have magnitude $1/\sqrt{n}$, and all those of $\boldsymbol{V}$ will have magnitude $1/\sqrt{d}$. A slightly weaker assumption than this, but one that suffices, is to assume an upper bound on the Euclidean norms of each row of $\boldsymbol{U}$ and of $\boldsymbol{V}$. Consider $\boldsymbol{U}$, which is $n \times r_L$. In the best (most dense) case, each of its rows will have norm $\sqrt{r_L/n}$. Incoherence or denseness means that each of its rows has a norm that is within a constant fraction of this number. Using the definition of the incoherence parameter μ given previously, this is equivalent to saying that $\boldsymbol{U}$ is μ-incoherent with μ being a numerical constant. All RPCA guarantees require that both $\boldsymbol{U}$ and $\boldsymbol{V}$ are μ-incoherent. One of the first two (parallel) guarantees for PCP [4] also made the following stronger assumption, which we will refer to as *strong incoherence*:

$$\max_{i=1,2,\ldots,n,j=1,2,\ldots,d}\left|(\boldsymbol{UV}')_{i,j}\right| \le \sqrt{\mu\frac{r_L}{nd}}. \qquad (1)$$

This says that the inner product between a row of $\boldsymbol{U}$ and a row of $\boldsymbol{V}$ is upper bounded. Observe that the required bound is $1/\sqrt{r_L}$ times what left and right incoherence would imply (by using Cauchy-Schwartz inequality).

Older robust principal component analysis solutions: Robust subspace learning [2] and variants

The S+LR definition for RPCA was introduced in [4]. However, even before this, many good heuristics existed for solving RPCA. The ones that were motivated by video applications did implicitly model outliers as sparse corruptions in the sense that a data vector (image) was assumed to contain pixel-wise outliers (occluded pixels). The most well known among these is the RSL [2] approach. In later work, other authors also tried to develop incremental versions of this approach [35], [36]. The main idea of all these algorithms is to detect the outlier data entries and either replace their values using nearby values [36] or weight each data point in proportion to its reliability (thus soft-detecting and down-weighting the detected outliers) [2], [35]. Outlier detection is done by projecting the data vectors orthogonal to the current subspace estimate and thresholding out large entries of the resulting vector. As seen from the exhaustive experimental comparisons shown in [12], the online heuristics fail in all experiments—both for simulated and real data. On the other hand, the original RSL method [2] remains a good practical heuristic.

Convex optimization-based solution: Principal component pursuit

The first solution to S+LR was introduced in parallel works by Candès et al. [4] (where they called it a solution to RPCA) and by Chandrasekharan et al. [5]. Both proposed to solve the following convex program, which was referred to as *PCP* in [4]:

$$\min_{\tilde{L},\tilde{S}}\|\tilde{\boldsymbol{L}}\|_* + \lambda\|\tilde{\boldsymbol{S}}\|_1 \text{ subject to } \boldsymbol{M} = \tilde{\boldsymbol{L}} + \tilde{\boldsymbol{S}}.$$

Here, $\|\boldsymbol{A}\|_1$ denotes the vector l_1 norm of the matrix $\boldsymbol{A}$ (sum of absolute values of all its entries), and $\|\boldsymbol{A}\|_*$ denotes the nuclear norm (sum of its singular values). The nuclear norm can be interpreted as the l_1 norm of the vector of singular values of the matrix. In other literature, it is also called the Schatten-1 norm. PCP is the first known polynomial time solution to RPCA that is also provably correct. Reference [4] both gave a simple correctness result and showed how PCP outperformed existing work at the time for the video analytics application.

Why it works
It is well known from CS literature (and earlier) that the vector l_1 norm serves as a convex surrogate for the support size (number of nonzero entries) of a vector (or vectorized matrix). In a similar fashion, the nuclear norm serves as a convex surrogate for the rank of a matrix. Thus, while the program that tries to minimize the rank of $\tilde{\boldsymbol{L}}$ and sparsity of $\tilde{\boldsymbol{S}}$ involves an impractical combinatorial search, the above program is convex and solvable in polynomial time.

Time and memory complexity
The complexity of solving a convex program depends on the iterative algorithm (solver) used to solve it. Most solvers have time complexity that is more than linear in the matrix dimension: a typical complexity is $O(nd^2)$ per iteration [8]. Also, they typically need $O(1/\epsilon)$ iterations to return a solution that is within ϵ error of the true solution of the convex program [8]. Thus the typical complexity is $O(nd^2/\epsilon)$. PCP operates on the entire matrix, so memory complexity is $O(nd)$.

Nonconvex solutions: Alternating minimization
To obtain faster algorithms, in more recent works, authors have tried to develop provably correct algorithms that rely on either alt-min or projected GD. Both alt-min and GD have been used for a long time as practical heuristics for trying to solve various nonconvex programs. The initialization either came from other prior information, or multiple random initializations were used to run the algorithm and the "best" final output was picked. The new ingredient in these provably correct solutions is a carefully designed initialization scheme that already outputs an estimate that is "close enough" to the true one.

For RPCA, the first provably correct nonconvex solution was AltProj [8]. This borrows some ideas from an earlier algorithm called *go decomposition* (*GoDec*) [37]. AltProj works by projecting the residual at each step onto the space of either sparse matrices or onto the space of low-rank matrices. The approach proceeds in stages with the first stage projecting onto the space of rank $\hat{r} = 1$ matrices, while increasing the support size of the sparse matrix estimate at each iteration in the stage. In the second stage, $\hat{r} = 2$ is used and so on for a total of r stages. Consider stage one. AltProj is initialized

by thresholding out the very large entries of the data matrix M to return the first estimate of the sparse matrix. Thus, $\hat{S}_0 = HT(M; \zeta_0)$, where HT denotes the hard thresholding operator and ζ_0 denotes the threshold used for it. After this, it computes the residual $M - \hat{S}_0$ and projects it onto the space of rank-1 matrices. Thus, $\hat{L}_1 = \mathcal{P}_1(M - \hat{S}_0)$, where $\mathcal{P}_1$ denotes a projection onto the space of rank-1 matrices. It then computes the residual $M - \hat{L}_1$ and projects it again onto the space of sparse matrices but using a carefully selected threshold ζ_1 that is smaller than ζ_0. Thus, $\hat{S}_1 = HT(M - \hat{L}_1; \zeta_1)$. This process is repeated until a halting criterion is reached. The algorithm then moves on to stage two. This stage proceeds similarly, but each low-rank projection is now onto the space of rank $\hat{r} = 2$ matrices. This process is repeated for r stages.

> **AltProj works by projecting the residual at each step onto the space of either sparse matrices or onto the space of low-rank matrices.**

Why this works
Once the largest outliers are removed, it is expected that projecting onto the space of rank-1 matrices returns a reasonable rank-1 approximation of L, $\hat{L}_1$. This means that the residual $M - \hat{L}_1$ is a better estimate of S than M. Because of this, it can be shown that $\hat{S}_1$ is a better estimate of S than $\hat{S}_0$ and so the residual $M - \hat{S}_1$ is a better estimate of L than $M - \hat{S}_0$. This, in turn, means $\hat{L}_2$ will be a better estimate of L than $\hat{L}_1$. The proof that the initial estimate of L is good enough relies on incoherence of left and right singular vectors of L and the fact that no row or column has too many outliers. These two facts are also needed to show that each new estimate is better than the previous.

Time and memory complexity
The time complexity of AltProj is $O(ndr^2 \log(1/\epsilon))$. It operates on the entire matrix, so memory complexity is $O(nd)$.

Nonconvex solutions: Projected gradient descent (Robust principal component analysis-gradient descent and nearly optimal-robust matrix completion)
While the time complexity of AltProj was lower than that of PCP, there was still scope for improvement in speed. In particular, the outer loop of AltProj that runs r times seems unnecessary (or can be made to run fewer times). Two more recent works [10], [17] try to address this issue. The question asked in [10] was can one solve RPCA with computational complexity that is of the same order as a single r-SVD? This has complexity $O(ndr(-\log \epsilon))$. The authors of [10] show that this is indeed possible with an extra factor of κ^2 in the complexity and with a tighter bound on outlier fractions. Here, κ is the condition number of L. To achieve this, they developed an algorithm that relies on projected GD called *RPCA-GD*. The authors of [17] use a different approach. They develop a projected GD solution for RMC and argue that, even in the absence of missing entries, the same algorithm provides a very fast solution to RPCA as long as the data matrix is nearly square. For solving RPCA, it deliberately undersamples the available full matrix M to speed up the algorithm. This algorithm is called *NO-RMC*.

Projected GD is natural heuristic for using GD to solve constrained optimization problems. To solve $\min_{x \in C} f(x)$, after each GD step, it projects the output onto the set C before moving on to the next iteration.

Robust principal component analysis-gradient descent [10]
Notice that the matrix L can be decomposed as $L = \tilde{U}\tilde{V}'$, where $\tilde{U}$ is an $n \times r$ matrix and $\tilde{V}$ is a $d \times r$ matrix. The algorithm alternately solves for $S, \tilde{U}, \tilde{V}$. Like AltProj, it also begins by first estimating the sparse component $\hat{S}_0$. Instead of hard thresholding, it uses a more complicated approach called *max-sorting-thresholding* for doing this. It then initializes $\hat{\tilde{U}}_0$ and $\hat{\tilde{V}}_0$ via r-SVD on $M - \hat{S}_0$ followed by "projecting onto the set of incoherent matrices" (we explain this in the next paragraph). After this, it repeats the following three steps at each iteration: 1) use "max-sorting-thresholding" applied to $M - \hat{L}_{i-1}$ to obtain $\hat{S}_i$; 2a) implement one GD step for minimizing the cost function $\mathcal{L}(\tilde{U}, \tilde{V}, S) := \|\tilde{U}\tilde{V}' + S - M\|_F^2 + 0.25\|\tilde{U}'\tilde{U} - \tilde{V}'\tilde{V}\|_F$ over $\tilde{U}$ while keeping $\tilde{V}, S$ fixed at their previous values, and 2b) obtain $\hat{\tilde{U}}_i$ by "projecting the output of step 2a onto the set of incoherent matrices"; and 3) obtain $\hat{\tilde{V}}_i$ in an analogous fashion.

The step "projecting onto the set of incoherent matrices" involves the following. Recall that incoherence (denseness) of left and right singular vectors is needed for the static RPCA solutions to work. To ensure that the estimate of $\tilde{U}$ after one step of GD satisfies this, RPCA-GD projects the matrix onto the "space of incoherent matrices." This is achieved by clipping: if a certain row has norm larger than $\sqrt{2\mu r/n}\|\hat{\tilde{U}}_0\|_2$, then each entry of that row is rescaled so that the row norm equals this value.

Nearly optimal-robust matrix completion [17]
The NO-RMC algorithm is designed to solve the more general RMC problem. Let Ω denote the set of observed entries, and let Π_Ω denote projection onto a subspace formed by matrices supported on Ω. Thus, $\Pi_\Omega(A)$ zeroes out entries of A whose indices are not in Ω and keeps other entries the same. The set Ω is generated uniformly at random. NO-RMC modifies AltProj as follows. First, instead of running the algorithm in r stages, it reduces the number of stages needed. In the qth outer loop, it projects onto the space of rank k_q matrices (instead of rank q matrices in case of AltProj). The second change is in the update step of $\hat{L}$, which now also includes a GD step before the projection. Thus, the ith iteration of the qth outer loop now computes $\hat{L}_i = \mathcal{P}_{k_q}(\hat{L}_{i-1} + (1/p)\mathcal{P}_\Omega(M - \hat{L}_{i-1} - \hat{S}_{i-1}))$. On first glance, the projection onto the space of rank k_q matrices should need $O(ndk_q)$ time. However, as the authors explain, because the matrix that is being projected is a sum of a low-rank matrix and a matrix with many zeroes (sparse matrix as used in fast matrix algorithms' terminology), the computational cost

for this step is actually only $O(|\Omega|k_q + (n+d)k_q + k_q^3)$ (see [17, p. 4]). The required number of observed entries is roughly $|\Omega| \approx nr^3 \log n$ with high probability. This is what results in a significant speed-up. However NO-RMC requires d to be of the same order as n. This is a stringent requirement for high-dimensional data sets for which n is large, e.g., it is hard to find videos that have as many frames as the image size n.

Nonconvex online solution for robust subspace tracking and robust principal component analysis: Recursive projected compressive sensing

The RST problem has received much less attention than RPCA [7], [9], [27]. In fact, even the easier subspace tracking or subspace tracking with missing data problems are not fully solved. Many algorithms exist, e.g., see [38]–[40]; however, all existing guarantees are asymptotic results for the statistically stationary setting of data being generated from a single unknown subspace; and most are partial guarantees. Of course, as explained previously, the most general nonstationary model that allows the subspace to change at each time is not even identifiable. The ReProCS approach [7], [11] described here resolves this issue by using the piecewise constant subspace change model described previously.

The simplest ReProCS algorithm, ReProCS-NORST [27], [28], starts with a "good" estimate of the initial subspace, which is obtained by $C(\log r)$ iterations of AltProj applied to $M_{[1, t_{\text{train}}]}$ with $t_{\text{train}} = Cr$. After this at each time, it first solves projected CS equivalently ("robust regression"; please see the section "Relating Projected Compressive Sensing and Robust Regression") to estimate the sparse outliers, s_t, and then ℓ_t as $\hat{\ell}_t = m_t - \hat{s}_t$; and then subspace update to update the subspace estimate $\hat{P}_{(t)}$. Projected CS (robust regression) proceeds as follows. At time t, let $\hat{P}_{(t-1)}$ denote the subspace estimate from the previous time instant, $(t-1)$. If this estimate is accurate enough, because of slow subspace change, projecting $m_t := s_t + \ell_t + w_t$ onto its orthogonal complement nullifies most of ℓ_t. To be precise, we compute $\tilde{m}_t := \Psi m_t$,

where $\Psi := I - \hat{P}_{(t-1)}\hat{P}'_{(t-1)}$. Thus, $\tilde{m}_t = \Psi s_t + b_t$, where $b_t := \Psi(\ell_t + w_t)$ and $\|b_t\|$, is small. Recovering s_t from $\tilde{m}_t$ is thus a regular CS/sparse recovery problem in small noise [41]. Notice that, even though Ψ is square, it is rank deficient: it has rank $n - r$. This is why we cannot just invert it to estimate s_t, but instead need to solve the CS problem. For solving the CS problem, any approach can be used. As an example, one can use l_1 minimization: compute $\hat{s}_{t,cs}$ using l_1 minimization, followed by thresholding-based support estimation to get $\hat{\mathcal{T}}_t$, and then a least squares (LS)-based debiasing step on $\hat{\mathcal{T}}_t$ to get $\hat{s}_t$. The reason we can accurately recover s_t by solving the CS problem is the following: denseness (incoherence) of the $P_{(t)}$s along with slow subspace change can be used to show that the matrix Ψ satisfies the restricted isometry property (RIP) [41] with a small enough RIP constant [7]. Once $\hat{s}_t$ is available, one can then recover ℓ_t by subtraction: thus, $\hat{\ell}_t = m_t - \hat{s}_t$.

The $\hat{\ell}_t$s are used in the subspace update step, which runs every α frames. In its simplest form, this involves 1) detecting subspace change and 2) obtaining progressively improved estimates of the changed subspace via K steps of r-SVD, each done with a new set of α frames of $\hat{\ell}_t$. Here, r-SVD means compute the top r left singular vectors of $\hat{L}_{t;\alpha} := [\hat{\ell}_{t-\alpha+1}, \hat{\ell}_{t-\alpha+2}, \ldots, \hat{\ell}_t]$. This step was designed assuming a piecewise constant subspace change model; however, the algorithm itself works even without this assumption (it works for real videos as well). A block diagram of the ReProCS framework is shown in Figure 4.

Why it works

It is not hard to see that the noise $b_t := \Psi(\ell_t + w_t)$ seen by the projected CS step is proportional to the error between the subspace estimate from $(t-1)$ and the current subspace. Using the RIP arguments given previously and the outlier magnitudes' lower-bound assumption, one can ensure that the CS step output is accurate enough and the outlier support $\mathcal{T}_t$ is correctly recovered. With this, it is not hard to see that $\hat{\ell}_t = \ell_t + w_t - e_t$, where $e_t := s_t - \hat{s}_t$ satisfies
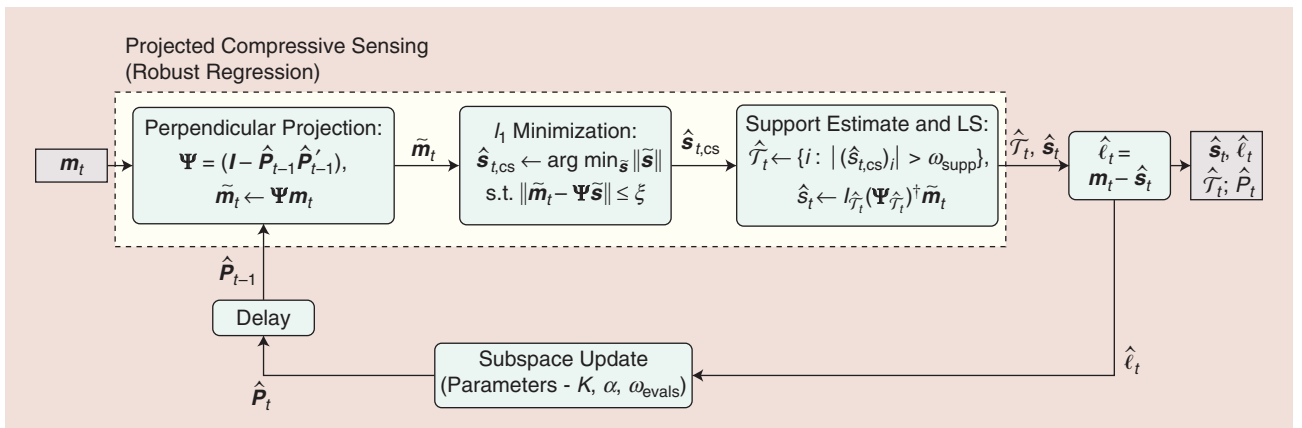

FIGURE 4. The ReProCS solution framework for RST or dynamic RPCA. The subspace update is either in "detect" phase or in "update" phase. Suppose that the jth change is detected at time $\hat{t}_j$. The update phase proceeds as follows. At time $t = \hat{t}_j + k\alpha$, for $k = 1, 2, \ldots, K$, it computes the jth estimate of P_j (denoted $\hat{P}_{j,k}$) as the top r singular vectors of $\hat{L}_{t;\alpha} := [\hat{\ell}_{t-\alpha+1}, \hat{\ell}_{t-\alpha+2}, \ldots, \hat{\ell}_t]$, and it sets $\hat{P}_t$ equal to this. At other times, it just sets $\hat{P}_t \leftarrow \hat{P}_{t-1}$. The detect phase uses the following idea: to detect the jth change, it checks if the maximum singular value of $(I - \hat{P}_{j-1}\hat{P}'_{j-1})\hat{L}_{t;\alpha}$ is larger than $\sqrt{\alpha}\omega_{\text{evals}}$.

$$e_t = I_{\mathcal{T}_t}(\Psi'_{\mathcal{T}_t} \Psi_{\mathcal{T}_t})^{-1} I'_{\mathcal{T}_t} \Psi' \ell_t$$

and $\|e_t\| \le C\|b_t\|$. Consider the subspace update. Every time the subspace changes, one can show that the change can be detected within a short delay. After that, the K SVD steps help get progressively improved estimates of the changed subspace. To understand this, observe that, after a subspace change but before the first update step, b_t is the largest, and, hence, e_t is also the largest for this interval. However, because of good initialization or because of slow subspace change and previous subspace correctly recovered (to error ϵ), neither is too large. Both are proportional to $(\epsilon + \Delta)$, or to the initialization error. Recall that Δ quantifies the amount of subspace change. For simplicity, suppose that $\mathrm{SE}(\hat{P}_0, P_0) = \Delta$. Using the idea presented next, we can show that we get a "good" first estimate of the changed subspace.

The input to the PCA step is $\hat{\ell}_t$, and the noise seen by it is e_t. Notice that e_t depends on the true data ℓ_t, and, hence, this is a setting of PCA in data-dependent noise [42], [43]. From [43], it is known that the subspace recovery error of the PCA step is proportional to the ratio between the time-averaged noise power plus time-averaged signal-noise correlation, $(\|\Sigma_t \mathbb{E}[e_t e'_t]\| + \|\Sigma_t \mathbb{E}[\ell_t e'_t]\|)/\alpha$, and the minimum signal space eigenvalue, λ^-. The instantaneous values of both noise power and signal-noise correlation are of order $(\Delta + \epsilon)$ times λ^+. However, using the fact that e_t is sparse with support $\mathcal{T}_t$ that changes enough over time so that max-outlier-frac-row$^\alpha$ is bounded, their time-averaged values are at least $\sqrt{\text{max-outlier-frac-row}^\alpha}$ times smaller. This follows using Cauchy-Schwartz. As a result, after the first subspace update, the subspace recovery error is below $3\sqrt{\text{max-outlier-frac-row}}\,(\lambda^+/\lambda^-)$ times $(\Delta + \epsilon)$. Since $9\text{max-outlier-frac-row}(\lambda^+/\lambda^-)^2$ is bounded by a constant $c_2 < 1$, this means that, after the first subspace update, the subspace error is below $\sqrt{c_2}$ times $(\Delta + \epsilon)$.

This, in turn, implies that $\|b_t\|$, and hence $\|e_t\|$, is also $\sqrt{c_2}$ times smaller in the second subspace update interval compared to the first. This, along with repeating the aforementioned argument, helps show that the second estimate of the changed subspace is $\sqrt{c_2}$ times better than the first, i.e., its error is $(\sqrt{c_2})^2$ times $(\Delta + \epsilon)$. Repeating the argument K times, the Kth estimate has error $(\sqrt{c_2})^K$ times $(\Delta + \epsilon)$. Since $K = C\log(1/\epsilon)$, this is an ϵ accurate estimate of the changed subspace.

Time and memory complexity

ReProCS memory complexity is $n\alpha$ in online mode and $Kn\alpha$ in offline mode. With the choices of K and α that suffice for its guarantee (see the ReProCS-NORST theorem given previously), its memory complexity is just $O(nr\log n)$ in online mode and $O(nr\log n\log(1/\epsilon))$ in offline mode. Both are within logarithmic factors of nr and thus nearly optimal: nr is the memory needed to output the estimate of an r-dimensional subspace in $\mathbb{R}^n$. The computational complexity is $O(ndr\log(1/\epsilon))$: this is the time needed to compute an r-SVD for an $n \times d$ matrix with good eigen-gap. Thus, the speed is as fast as it can get without deliberately undersampling the input data for algorithm speed-up.

Summary of recursive projected compressive sensing-based solutions

In [7] and [11], a novel solution framework ReProCS was introduced to solve the RST or the dynamic RPCA problem. In later works, this was shown to be provably correct under progressively weaker assumptions [9], [27], [29], [44]. Under two extra assumptions, the ReProCS framework provides an online (after initialization), fast, memory-efficient, and highly robust solution (see the ReProCS-NORST theorem given previously). By *highly robust* we mean that, after initialization, it can tolerate an order-wise larger maximum fractions of outliers in any row than other RPCA approaches. For the video application, this means that it tolerates slow-moving or occasionally static foreground objects better than other RPCA approaches. This is also evident from the experiments; see the column for intermittent object motion (IOM) category of videos in Table 3. ReProCS can also tolerate a larger fraction of outliers per column especially when $r_L \gg r$. The extra assumptions needed are simple and practically valid: 1) slow subspace change (or fixed subspace in case of static RPCA) and 2) outlier magnitudes are either all large enough, or most are large enough and the rest are very small. The first assumption is a natural assumption for static camera videos (with no sudden scene changes), and the second is also easy because, by definition, an *outlier* is a large magnitude corruption. The small magnitude ones get classified as w_t. In practice, intermediate magnitude outliers are the most problematic of course (for all approaches). We explain the improved outlier tolerance and these extra requirements in the next section. Also, as noted there, the outlier magnitude lower bound can be relaxed.

We have explained the latest and simplest version of ReProCS here [27]. Under slightly stronger assumptions on subspace change, for example, if it is known that, at each subspace change time, only one subspace direction change, the SVD step, can be replaced by a slightly faster approach called *projection-SVD* [7], [12], [29]. The simplest version of the resulting algorithm is simple-ReProCS studied in [29].

Looser bound on max-outlier-frac-row and outlier magnitudes' lower bound

As explained previously, if the allowed maximum outlier fraction in any row is more than $1/r_L$, the original RPCA problem becomes unidentifiable. ReProCS tolerates a constant maximum fraction per row because it uses extra assumptions. We explain these here. It recovers s_t first and then ℓ_t and does this at each time t. To recover s_t accurately, it does require that ℓ_t not be sparse (incoherence or denseness of the P_js ensures this). Moreover, it also exploits "good" knowledge of the subspace of ℓ_t (either from initialization or from the previous subspace's estimate and slow subspace change). But it has no way to deal with the residual error, $b_t := (I - \hat{P}_{(t-1)}\hat{P}'_{(t-1)})\ell_t$ in the subspace knowledge. Since the individual vector b_t does not have any structure that can exploited, the error in recovering s_t cannot be lower than $C\|b_t\|$. Thus, to enable correct recovery of the support of s_t, $s_{\min}$ needs to be larger than $C\|b_t\|$. This is where the $s_{\min}$ lower bound comes from. Correct support

recovery is needed to ensure that the subspace estimate can be improved with each update. This step also requires element-wise boundedness of the $\boldsymbol{a}_t$s along with mutual independence and identical covariances (these together are similar to right incoherence of $\boldsymbol{L}$; see [27]).

Remark 3.5 (Relax outlier magnitudes lower bond assumption)

The outlier magnitudes, lower bound assumption of the ReProCS-NORST theorem given in the "An Overview" section can be relaxed to lower bound on most outlier magnitudes. In particular, the following suffices: assume that $\boldsymbol{s}_t$ can be split into $\boldsymbol{s}_t = (\boldsymbol{s}_t)_{\text{small}} + (\boldsymbol{s}_t)_{\text{large}}$ that are such that, in the kth subspace update interval, $\|(\boldsymbol{s}_t)_{\text{small}}\| \le 0.3^{k-1}(\epsilon + \Delta)\sqrt{r\lambda^+}$ and the smallest nonzero entry of $(\boldsymbol{s}_t)_{\text{large}}$ is larger than $C0.3^{k-1}(\epsilon + \Delta)\sqrt{r\lambda^+}$.

If there were a way to bound the element-wise error of the CS step (instead of the l_2 norm error), the above requirement could be relaxed further.

Relating projected compressive sensing and robust regression

Projected CS is equivalent to solving the robust regression problem with a sparsity model on the outliers. To understand this simply, let $\hat{\boldsymbol{P}} = \hat{\boldsymbol{P}}_{(t-1)}$. The exact version of robust regression assumes that the data vector $\boldsymbol{m}_t$ equals $\hat{\boldsymbol{P}}\boldsymbol{a}_t + \boldsymbol{s}_t$, while its approximate version assumes that this is only approximately true. Since $\hat{\boldsymbol{P}}$ is only an approximation to $\boldsymbol{P}_{(t)}$, even in the absence of unstructured noise $\boldsymbol{w}_t$, approximate robust regression is the correct problem to solve for our setting. Its unconstrained version involves $\min_{\boldsymbol{a},\boldsymbol{s}} \lambda \|\boldsymbol{s}\|_1 + \|\boldsymbol{m}_t - \hat{\boldsymbol{P}}\boldsymbol{a} - \boldsymbol{s}\|^2$. Observe that one can solve for $\boldsymbol{a}$ in closed form to get $\hat{\boldsymbol{a}} = \hat{\boldsymbol{P}}'(\boldsymbol{m}_t - \boldsymbol{s})$. Substituting this, the minimization simplifies to $\min_{\boldsymbol{s}} \|\boldsymbol{s}\|_1 + \|(\boldsymbol{I} - \hat{\boldsymbol{P}}\hat{\boldsymbol{P}}')(\boldsymbol{m}_t - \boldsymbol{s})\|^2$. This is exactly the same as the Lagrangian version of projected CS. The version for which we obtain guarantees solves $\min_{\boldsymbol{s}} \|\boldsymbol{s}\|_1 \text{ s.t. } \|(\boldsymbol{I} - \hat{\boldsymbol{P}}\hat{\boldsymbol{P}}')(\boldsymbol{m}_t - \boldsymbol{s})\|^2 \le \xi^2$, but we could also have used other CS results and obtained guarantees for the Lagrangian version with minimal changes.

Online solution 2: Modified-principal component pursuit for robust principal component analysis with partial subspace knowledge and for dynamic robust principal component analysis

A simple extension of PCP, called *modified-PCP*, provides a nice solution to the problem of RPCA with partial subspace knowledge [30]. This problem often occurs in face recognition-type applications: the training data set may be outlier-free, but, due to limited training data, it may not contain all possible directions of variation of the face subspace. To understand the solution idea, let $\boldsymbol{G}$ denote the basis matrix for the partial subspace knowledge. If $\boldsymbol{G}$ is such that the matrix $(\boldsymbol{I} - \boldsymbol{GG}')\boldsymbol{L}$ has rank smaller than r_L, then the following is a better idea:

$$\min_{\tilde{\boldsymbol{L}},\tilde{\boldsymbol{S}}} \|(\boldsymbol{I} - \boldsymbol{GG}')\tilde{\boldsymbol{L}}\|_* + \lambda \|\tilde{\boldsymbol{S}}\|_1 \text{ subject to } \tilde{\boldsymbol{L}} + \tilde{\boldsymbol{S}} = \boldsymbol{M}.$$

This solution was called *modified-PCP* because it was inspired by a similar idea, called *modified-CS* [45], that solves

the problem of CS (or sparse recovery) when partial support knowledge is available. More generally, even if the approximate rank of $(\boldsymbol{I} - \boldsymbol{GG}')\boldsymbol{L}$ is much smaller, i.e., suppose that $\boldsymbol{L} = \boldsymbol{GA} + \boldsymbol{W} + \boldsymbol{L}_{\text{new}}$, where $\boldsymbol{L}_{\text{new}}$ has rank $r_{\text{new}} \ll r_L$ and $\|\boldsymbol{W}\|_F \le \epsilon$ is small, the following simple change to the same idea works:

$$\min_{\tilde{\boldsymbol{L}}_{\text{new}},\tilde{\boldsymbol{S}},\tilde{\boldsymbol{A}}} \|\tilde{\boldsymbol{L}}_{\text{new}}\|_* + \lambda \|\tilde{\boldsymbol{S}}\|_1 \text{ s.t. } \|\boldsymbol{M} - \tilde{\boldsymbol{S}} - \boldsymbol{G}\tilde{\boldsymbol{A}}\|_F \le \epsilon$$

and output $\hat{\boldsymbol{L}} = \boldsymbol{G}\hat{\boldsymbol{A}} + \hat{\boldsymbol{L}}_{\text{new}}$. To solve the tracking problem using modified-PCP, the subspace estimate from the previous set of α frames serves as the partial subspace knowledge $\boldsymbol{G}$ for the current set. It is initialized using PCP.

The modified-PCP guarantee was proved using ideas borrowed from [4] for PCP [PCP(C)]. It shows that, when $r_{\text{new}} \ll r$, modified-PCP needs a much weaker version of the strong incoherence condition needed by PCP. However, like PCP (and unlike ReProCS), it also needs uniformly randomly generated support sets, which is a strong requirement. But, also like PCP (and unlike ReProCS), it does not need a lower bound on outlier magnitudes. However, its slow subspace change assumption is unrealistic. Also, it does not detect the subspace change automatically, i.e., it assumes t_j is known. This last limitation can be removed by using the ReProCS approach to detecting subspace change [9], [29].

Summary discussion of all simple (and provably correct) approaches

A summary is provided in Table 1. The results for PCP(C) and modified-PCP both require uniform random support (impractical requirement), but, because of that, both allow a constant outlier fraction in any row or column of the data matrix. The PCP(H) result as well as all follow up works (AltProj, ReProCS, GD, NO-RMC) does not require any outlier support generation model, but, because of that, they require a tighter bound on the outlier fractions in any row or column. AltProj and NO-RMC allow this to be $O(1/r_L)$ while GD requires this to be $O(1/r_L^{1.5})$. ReProCS has the best outlier fraction tolerance: for the first Cr frames it needs the fractions to be $O(1/r)$; after that it needs the fraction in any row of a α sample mini-batch to only be $O(1)$, while it needs the fraction in any column to be $O(1/r)$. The row bound is better than the information theoretic upper bound on what can be tolerated without extra assumptions. ReProCS uses fixed or slow subspace change and most outlier magnitudes lower bounded to obtain this advantage. These two extra assumptions (and a different way to ensure right incoherence) also help get an online algorithm that has near-optimal memory complexity and near-optimal tracking delay.

In terms of time complexity, PCP is the slowest because it needs to solve a convex program; AltProj is much faster. GD and ReProCS are faster than AltProj, and NO-RMC is the fastest. Up to log factors, its time complexity is order nr_L^2. For small r_L, this is almost linear in the number of unknowns. However, to achieve this, it deliberately undersamples the full data matrix and then solves an RMC problem; and for this to work correctly, it requires the data matrix to be nearly square.

This is a strong requirement. Also, it should be clarified that this discussion does not incorporate the undersampling complexity. At least, the simple way to generate a Bernoulli support (for the undersampling) will incur a cost that is linear in the matrix size, i.e., $O(nd)$.

In terms of practical performance, for a video analytics (foreground-background separation) application, ReProCS has the best performance among all of the provable methods, and it is also among the fastest methods. In fact, for the dynamic background (DB) and IOM category of videos, it has the best performance compared to all methods that do not use more assumptions. The IOM videos are examples of large outlier fractions per row, while the DB videos are examples of large r_L (but r may be smaller). In terms of MATLAB time taken, modified-PCP and ReProCS are the fastest. The others are at least three times slower. Of these, AltProj is the third fastest while both GD and NO-RMC are even slower than it experimentally. One reason RMC is slow is that undersampling is expensive. All approaches need a form of left and right incoherence as discussed previously.

Other online heuristics for robust principal component analysis and dynamic robust principal component analysis (Robust subspace tracking)

After the ReProCS algorithm was introduced in 2010 [11], [18], many other online algorithms for RPCA were proposed. None of these come with complete guarantees.

Bilinear decomposition [46]

Mateos and Giannakis proposed a RPCA approach using bilinear decomposition (Bilinear-Dec) with sparsity control. Their approach uses a least-trimmed squares (LTS) PCA estimator that is closely related to an estimator obtained from an l_0-norm regularized criterion, adopted to fit a low-rank bilinear factor analysis model that explicitly incorporates an unknown sparse vector of outliers per datum. Efficient approximate solvers are employed by surrogating the l_0-norm of the outlier matrix with its closest convex approximation. This leads to an M-type PCA estimator.

Go decomposition [37]

GoDec can be understood most easily as a precursor to the Alt-Proj algorithm described in detail previously. Its initialization is a little different: the first step involves low-rank projection of the data matrix instead of first removing large outliers. Also, it does not proceed in stages.

Stochastic optimization [47]

In [47], an online algorithm was developed to solve the PCP convex program using stochastic optimization. This relies the fact that the nuclear norm of a rank r matrix L can be computed as the minimizer of the squared sum of the Frobenius norms of matrices $\tilde{U}$ and $\tilde{V}$ under the constraint $L = \tilde{U}\tilde{V}'$.

We list online RPCA (OR-PCA) as a heuristic because it only comes with a partial guarantee: the guarantee assumes that the subspace estimate outputted at each time t is full rank.

Adaptive projected subgradient method [48], [49]

For the adaptive projected subgradient method (APSM) algorithm, at each time instant, a cost function is defined based on the incoming data. This cost function scores a zero loss for a nonempty set of points/possible solutions. The aim is to find a point that belongs to the intersection of all the sets associated with the incoming data. The time instant at which the data contain outlier noise is identified. CS matching pursuit (CoSAMP) is used to estimate the sparse outlier vector.

Grassmannian robust adaptive subspace tracking algorithm [50]

This borrows the main idea of ReProCS— projected CS (also known as *approximate robust regression*) followed by subspace update using the estimated true data vectors. But it replaces both the steps by different and approximate versions. The Grassmannian robust adaptive subspace tracking algorithm (GRASTA) solves the exact version of robust regression, which involves recovering a_t as $\arg\min_a \| m_t - \hat{P}_{(t-1)}a \|_1$. This formulation ignores the fact that $\hat{P}_{(t-1)}$ is only an approximation to the current subspace $P_{(t)}$. This is why, in experiments, GRASTA fails when there are significant subspace changes: it ends up interpreting the subspace tracking error as an outlier. In its subspace update step, the SVD or projected-SVD used in different variants of ReProCS [12], [18], [27], are replaced by a faster but approximate subspace tracking algorithm called *GROUSE* [40].

l_p norm robust online subspace tracking [51] and ROSETA

Both these algorithms modify the GRASTA approach and hence, indirectly rely on the basic ReProCS framework of alternating projected sparse recovery (robust regression) and subspace update. pROST replaces l_1 minimization in the robust regression step by nonconvex l_0-surrogates (l_p norm minimization for $p < 1$). Therefore, the authors call their algorithm l_p *norm robust online subspace tracking* (*pROST*). ROSETA uses the same framework, but projection coefficients and sparse outliers are computed using ADMM and the subspace is updated using a proximal point iteration with adaptive parameter selection.

Heuristics that use extra application-specific spatial and/or temporal constraints

The algorithms described previously solve only the RPCA or dynamic RPCA problems, without using extra application specific constraints. Using such constraints can significantly improve algorithm performance when they are valid. There has been a lot of work in this area, and it is hard to discuss or even mention all of it in this review article. We mention a few approaches as examples; all of these are motivated by the

> A simple extension of PCP, called *modified-PCP*, provides a nice solution to the problem of RPCA with partial subspace knowledge.

video analytics application. The incPCP algorithm is an RP-CA-based solution for video analytics that is online and near real time; it uses extra heuristics to deal with camera jitter and to handle panning and camera motion [52]. Grassmannian online subspace updates with structured sparsity (GOSUS) [53] is another incremental algorithm that uses structured sparsity of the outlier terms in conjunction with a GRASTA-like (or an ReProCS-like) algorithm.

Finally, two useful modifications of the ReProCS idea exploit the fact that, for many video applications, the foreground (sparse outlier) support changes in a correlated fashion over time. The first, called *modified-ReProCS* [12], [18], simply assumes slow support change of the sparse outliers. Thus, instead of recovering the outlier vector and its support via l_1 minimization (or CoSaMP, etc.), it uses the modified-CS idea [45]. Modified-CS is designed to exploit partial support knowledge in sparse signal recovery. The estimate of the outlier support from the previous time instant serves as the partial support knowledge. Replacing modified-CS by weighted-l_1 further helps in certain settings. The support-predicted modified-ReProCS approach (modified-ReProCS) [54] generalizes this basic idea further to include a support prediction step based on a simple object motion model and a Kalman filter to track the object's motion and velocity. Modified-ReProCS [12] is a truly practically useful algorithm since it can be used without any assumptions on how many moving objects, or regions, are there in a video; it only assumes that the foreground support does not change drastically. Its practical version uses a simple heuristic to decide when to use modified-CS and when to stick with simple l_1 minimization. We compare it in Table 3.

Robust subspace recovery

History

The robustness of PCA methods was first addressed in the fields of statistics and neural networks. In the statistics literature in 1980s, the predominant approach consists in replacing the standard estimation of the covariance matrix with a robust estimator of the covariance matrix [55], [56]. This formulation weights the mean and the outer products that form the covariance matrix. Calculating the eigenvalues and eigenvectors of this robust covariance matrix gives eigenvalues that are robust to sample outliers. The result is more robust but unfortunately is limited to relatively low-dimensional data. The second approach to RPCA uses this idea along with projection pursuit techniques [57]. In the old neural network literature in the 1990s, Xu and Yuille [1] first addressed the problem of RPCA by designing a neural network that relied on self-organizing rules based on statistical physics. The PCA energy function proposed by Oja was generalized by adding a binary decision field with a given prior distribution to take into account the outliers. The binary variables are zero when a data sample is considered an outlier.

> **The CDnet 2012 data set is the real-world region-level benchmark obtained by human experts.**

Low-rank plus column-sparse interpretation: Outlier pursuit

Recall that RSR assumes that an entire data vector is either an inlier or an outlier. Borrowing ideas from the S+LR literature, one way to reformulate this problem is as follows. The data matrix M can be split as $M = L + S_C$, where S_C is a column-sparse matrix (many columns are zero and some are nonzero) [58]. With this, an easy extension of the PCP idea can solve this problem. This was called *outlier pursuit* in [58] and solves

$$\min_{\tilde{L},\tilde{S}} \| \tilde{L} \|_* + \lambda \| \tilde{S}_C \|_{2,1} \text{ subject to } M = \tilde{L} + \tilde{S}_C.$$

Here, $\| \tilde{S} \|_{2,1}$ is the l_1 norm of the vector of column-wise l_2 norms. It is a commonly used convex surrogate for promoting group sparsity in the CS literature. The same program was also proposed in almost parallel work by McCoy and Tropp [31] where it was called *low-leverage decomposition*. The guarantee given by Xu et. al [58] for outlier pursuit is given previously in Theorem 2.4.

Maximizing mean absolute deviation via mean absolute deviation rounding

In [31], McCoy and Tropp developed a randomized algorithm called *MDR* to approximate the robust first PC. This is defined as $v_{MDR} := \max_{v:\|v\|=1} \| M'v \|_1 := \max_{\|v:\|=1} \sum_i \| v'm_i \|$. This cost is called *MAD* and is a classical metric for robust estimation known since the work of Huber going back to the 1980s. The later PCs can be computed by projecting the data vectors orthogonal to the subspace spanned by the previous ones (the projection pursuit PCA idea of Huber). The MDR solution is a computationally efficient algorithm to approximate v_{MDR}. It first solves a semidefinite programming (SDP) relaxation of the original maximizing MAD problem. The SDP can be solved in polynomial time, whereas the original problem is nonconvex. Next, it uses the output of the SDP in a randomization procedure to generate K candidate guesses of v_{MDR}. The algorithm then picks the "best" one (the one that maximizes the MAD) and outputs that as an estimate of v_{MDR}. The authors show that the MDR algorithm is guaranteed to provide an estimate of the first robust PC whose MAD is at least $(1 - \epsilon)$ times that of v_{MDR} with high probability The probability is high enough when K is large enough.

Outlier-robust principal component analysis [32]

Given a mix of authentic and corrupted points, Xu et al. [32] proposed the high-dimensional RPCA (HR-PCA) approach. This attempts to find a low-dimensional subspace that captures as much variance of the authentic points as possible. As explained in [32], HR-PCA alternates between a PCA and a "random removal" step in each iteration. For the PCA step, a candidate subspace is found via PCA on the "clean points" identified by the previous iteration. Its quality is measured by computing its trimmed variance (TV) which is defined as $TV(\hat{P}) := \sum_{j=1}^{r} \sum_{k=1}^{r} \left| (\hat{P})'_j m \right|_{(k)}^2 \Big/ r$. Here $\left| (\hat{P})'_j m \right|_{(k)}$ is the

kth order statistic (kth largest entry) among the "clean" data points in the current iteration $|(\hat{P})'_j \boldsymbol{m}_i|, i = 1, 2, \ldots, d$ and $(\hat{P})_j$ denotes the jth column of $\hat{P}$. In the first iteration, all points are treated as "clean points." In consecutive iterations, this number is reduced by one by using "random removal" done as follows: the probability of a point $\boldsymbol{m}_i$ being removed is proportional to $\Sigma^r_{j=1}((\hat{P})'_j \boldsymbol{m}_i)^2$. In the end, the algorithm compares the subspace estimates from all iterations and picks the one with largest TV. The guarantee given for HR-PCA provides a lower bound on the expressed variance of the computed PCs. The obtained bound is hard to parse. As explained by the authors, it shows that the expressed variance is greater than zero (the solution is not meaningless) whenever the fraction of outliers is less than 50%. On the other extreme, if the fraction of outliers goes to zero with n, then asymptotically the expressed variance goes to one (this corresponds to perfect recovery).

Solutions for robust supspace recovery

RSR as addressed by Lerman et al. [13] aims to recover the underlying subspace, P, from a data set consisting of inlier and outlier points by using the following robust M-estimator:

$$\min_{P: P'P = I} \sum_{i=1}^{d} \| \boldsymbol{m}_i - \mathcal{P}_P \boldsymbol{m}_i \| = \sum_{i=1}^{d} \| \mathcal{P}_{P_\perp} \boldsymbol{m}_i \|.$$

Here, $P_\perp$ is the orthogonal complement of P, and $\mathcal{P}_P$ and $\mathcal{P}_{P_\perp}$ denote orthogonal projections onto P and $P_\perp$, respectively. Since one needs to minimize over all matrices P with orthonormal columns, the above is a nonconvex problem. However, notice that $\Sigma^d_{i=1} \| \mathcal{P}_{P_\perp} \boldsymbol{m}_i \| = \Sigma^d_{i=1} \| P_\perp P_\perp' \boldsymbol{m}_i \|$. The matrix $Q := P_\perp P_\perp'$ is a projection matrix with properties $Q = Q', Q^2 = Q$, and $tr(Q) = n - r$. This intuition leads to the following convex relaxation-based approach. The geometric median subspace (GMS) algorithm [13] proceeds as follows: define the set $\mathbf{H} = \{ Q \in \mathbb{R}^{n \times n} : Q = Q', tr(Q) = 1 \}$, and then solve

$$\hat{Q} = \arg \min_{Q \in \mathbf{H}} F(Q) := \sum_{i=1} d \| Q \boldsymbol{m}_i \|.$$

For the noiseless case, the subspace P is estimated as the eigenvectors corresponding to the smallest r eigenvalues of $\hat{Q}$. In later work, Lerman et al. [59] suggested a tighter convex relaxation of the original problem. Their approach, called *REAPER*, suggests solving the following:

$$\hat{B} = \arg \min_B \sum_{i=1}^{d} \| \boldsymbol{m}_i - B \boldsymbol{m}_i \| \text{ s.t. } 0 \preceq B \preceq I, \ tr(B) = r.$$

This is the tightest convex relaxation because its constraint set is the convex hull of the set of basis matrices. An estimate of P is then obtained as the top r eigenvectors of $\hat{B}$.

Both GMS and REAPER work in the batch setting and, therefore, do not scale to big data. To address these limitations, Lerman et al. [60] provided three stochastic approximation algorithms that combined GMS and REAPER with the stochastic GD (SGD) idea. Each improved algorithm presents less running time and space complexity requirements than the batch GMS and REAPER.

Online principal component analysis with contaminated data [33]

Feng et al. [33] addressed the case of online PCA where outliers have no structural assumptions on them and data vectors come in sequentially. They developed an online algorithm that employed a probabilistic admiting/rejection procedure when a new sample comes in. Its guarantee makes assumptions on the initialization step (which is not specified), and the guarantee itself is asymptotic. The assumption on outlier fraction is very mild: anything less than 50% corruption works.

Parameter setting and experimental comparisons

We first explain how to set the parameters for the key approaches. After this, we provide a quantitative performance evaluation of these and many other RPCA methods for foreground-background separation (background subtraction) using the change detection (CDnet) 2012 data set [61]. For all the videos, the authors' code with their final parameter settings was used. No extra tuning was done for any approach.

Parameter setting

Parameter setting for principal component pursuit [4]
There is only one parameter λ in the PCP convex program. A smaller value of λ allows the solution $\hat{S}$ to be less sparse. Thus, its value should be chosen based on the expected number of outliers in the entire data matrix. However, the PCP convex program itself requires a solver (an iterative algorithm that solves it to within ϵ error in a finite number of iterations). The solver has various other parameters. The selection of which solver to use can itself be another parameter.

Parameter setting for alternating projection [8]
The AltProj algorithm requires two parameters, and a third to indicate the desired tolerance ϵ. The first is the rank. No guidance is provided on how to set this except cross-validation. The second, referred to as a *thresholding parameter*, is β, which controls the rate at which the sparse outliers are detected. A larger value of β indicates that after each "inner loop," the smallest nonzero value of the outlier is larger. A larger value of β indicates that the algorithm converges quickly if the outliers are very large/small. To detect outliers that have intermediate range requires a smaller value for β. The authors show that setting $\beta = 1/n$ works well in video experiments.

Parameter setting for robust principal component analysis-gradient descent [10]
There are five parameters to be tuned for RPCA-GD. It needs an estimate of the rank, r_L, the corruption fraction (the maximum fraction of outliers per each row and column), α_{GD}, a "safe-bound" on the estimate of corruption fraction, γ, the step size for the projected GD subroutine, η, and the number of iterations, T. In [10], the authors show that the step size can be set as $O(1/\sigma_1(\boldsymbol{U}_0 \boldsymbol{V}_0'))$, where $\boldsymbol{U}_0$ and $\boldsymbol{V}_0$ denote the first estimate of the factorized low-rank estimate, i.e., $\boldsymbol{M} - \hat{\boldsymbol{S}}_0 = \boldsymbol{U}_0 \boldsymbol{V}_0'$. No intuition is given on how to set the other parameters; please see their code.

Parameter setting for nearly optimal-robust matrix completion [17]

The parameter setting here is very similar to that of AltProj. However, in addition to these parameters, NO-RMC also requires an estimate of the incoherence parameter μ, and an estimate of the largest singular value of L, σ. Both of these estimates are required to set the number of iterations for the inner loop and ensure that the rank k_q estimate is "good enough." For default parameters, please see their code.

Parameter setting for recursive projected compressive sensing [27], [29]

The first step for setting algorithm parameters for ReProCS-based approaches is to obtain an initialization. There are two approaches that can be used for this. If there is outlier-free training data (e.g., for long surveillance videos obtained from closed-circuit television cameras), one can use simple SVD to obtain the initial subspace estimate: compute the top singular vectors that contain at least a certain percentage, e.g., 95%, of the total energy (sum of squares of all singular values). Another approach is to use a few iterations of a batch-RPCA method such as PCP or AltProj applied to the first $t_{\mathrm{train}} = Cr$ frames of data. PCP requires no tuning parameters but is slower. AltProj is faster but needs to know r. By cross-validation, we conclude that $r = 40$ and $t_{\mathrm{train}} = 400 = 10r$ suffices in all of our video experiments. We set $K = C\log(1/\epsilon) = 3$, and $\alpha = Cr\log n = 60$ as required by the guarantee. Since the dependence of α on n is only logarithmic, for all practically used values of n ($n = 1{,}000$ to $n = 320*240$) $\alpha = 60$ suffices. The parameter λ^+ is computed as the square of the largest singular value of $\hat{L}_{\mathrm{init}}/\sqrt{t_{\mathrm{train}}}$; and λ^- is the square of its rth largest singular value. The two parameters for the projected CS step are ξ and ω_{supp}. The required theoretical values of these depend on $S_{\min}$: the theorem requires $\xi = s_{\min}/15$ and $\omega_{\mathrm{supp}} = s_{\min}/2$. One way to estimate $s_{\min}$ is to let it be time varying and set it as $\min_{i \in \hat{\mathcal{T}}_{t-1}}|(\hat{s}_{t-1})_i|$ at t. An approach that works better for the video application is to set $\xi_t = \|\Psi\hat{\ell}_{t-1}\|$, $\omega_{\mathrm{supp},t} = \sqrt{\|m_t\|^2/n}$. The former relies on the idea that ξ_t needs to be a bound $\|\Psi\ell_t\|$. The latter is the root mean square value of the measurement vector at time t. If there are not too many outliers, most outlier magnitudes will be larger than this threshold (the problematic outliers being large magnitude corruptions), and, hence, this is a useful support estimation threshold. Finally the subspace change detection threshold ω_{evals} can be set equal to a very small fraction of λ^+, $\omega_{\mathrm{evals}} = 0.0007\lambda^+$ is used in all video experiments.

Experimental comparisons for video layering (foreground-background separation)

We evaluated the performance of the current state-of-the-art RPCA-based methods for foreground-background separation (background subtraction) using the CDnet 2012 data set [61]. We compared a total of 26 existing methods comprising 16 batch methods and eight online methods. These methods can be classified into the following three main categories:

- *Provable methods* comprise PCP [4], [5], nonconvex AltProj-based RPCA [8], NO-RMC [17], RPCA-GD [10], ReProCS [7], [12], simple ReProCS (simple-ReProCS) [29], and modified-PCP [30].
- *Heuristics methods* include GRASTA [50], OR-PCA [47], three-term decomposition (3TD) [62], two-pass RPCA (2PRPCA) [63], GoDec [37], pROST [51], and PRMF [64].
- *Heuristics methods with application-specific constraints* consist of modified-ReProCS [12], incremental PCP (incP-CP) [52], motion-assisted spatiotemporal clustering of low rank (MSCL) [65], detecting contiguous outliers in the low-rank representation (DECOLOR) [66], low-rank structured-sparse decomposition (LSD) [67], total variation RPCA (TVRPCA) [68], spatiotemporal RPCA (SRPCA) [69], robust motion-assisted matrix restoration (RMAMR) [70], generalized fused lasso (GFL) [71], GOSUS [53], contiguous outliers representation via online low-rank approximation (COROLA) [72], online mixture of Gaussians for matrix factorization with total variation (OMoGMF+TV) [73], and OR-PCA with illumination constraints (OR-PCA-illum) [74].

We used the authors' implementations of TVRPCA, 2PRPCA, GRASTA, GoDec, DECOLOR, 3TD, LSD, SRPCA, GFL, GOSUS, RMAMR, OMoGMF+TV, simple-ReProCS, modified-ReProCS, AltProj, NO-RMC, and RPCA-GD, where we reported the results that were obtained in their respective studies for the remaining methods. The implementation of all the methods is also available in the low-rank and sparse library [26]. The execution times required by all of the algorithms were compared on a machine with a 3.0 GHz Intel core i5 processor and 4GB of random access memory.

The CDnet 2012 data set [61] is the real-world region-level benchmark obtained by human experts. This data set contains almost 31 video sequences that are divided into the following six different video categories:

- *baseline*
- *DB*
- *IOM*
- *thermal*
- *camera jitter*
- *shadows*.

The resolution of the videos also varies from 320×240 to 480×720 with hundreds to thousands of frames.

Visual results were reported using 15 challenging sequences from CDnet data set for comparison purpose. This contained

- two sequences *highway* and *office* from the *baseline* category
- three sequences *canoe*, *boats*, and *overpass* from the *DB* category
- two sequences *traffic* and *badminton* from the *camera jitter* category
- three sequences *winterDriveway, sofa*, and *streetLight* from the *IOM* category
- three sequences *backdoor, copyMachine*, and *cubicle* from the *shadows* category

- two sequences *library* and *lakeside* from the *thermal* category.

Figure 5 provides qualitative results and comparisons of 22 current state-of-the-art RPCA-based methods on 15 sequences.

We used ground truth-based metrics computed from the true positive (TP), true negative (TN), false positive (FP), and false negative (FN) for quantitative validation. FP and FN refer to pixels misclassified as foreground (FP) or background (FN), while TP and TN account for accurately classified pixels, respectively, as foreground and background. Then, we computed the metrics used in the CD.net [61] data set evaluation such as the recall, the precision, and the F_1-measure score. Recall gives the percentage of correct pixels classified as background when compared with the total number of background pixels in the ground truth as Recall $= TP/TP + FN$. Precision gives the percentage of correct pixels classified as background as compared at the total pixels classified as background by the method as Precision $= TP/TP + FP$. A good performance is obtained when the detection rate, also known as *recall,* is high without altering the precision. Based on these metrics, we computed the F_1-measure (or effectiveness measure) as $F_1 = 2 \times \text{Recall} \times \text{Precision}/\text{Recall} + \text{Precision}$. The F-measure characterizes the performance of classification in precision-recall space. The aim is to maximize F_1-measure closed to one. Table 3 shows the quantitative results in terms of average F_1 measure score of all of the compared categories reviewed in this study.

On average, among all methods that do not use extra constraints, PRMF, 2PRPCA, and simple-ReProCS had the best performance with F_1 scores of 74–78%. On average for all data sets, only two of the methods that use extra constraints—MSCL and GOSUS—were better and only by a little—achieved 83% and 81% scores, respectively.

We also reported the computational time in seconds for all of the compared methods. We selected a very large video sequence known as *boats* from the *DB* category for this purpose. The boats video contains about 8,000 image sequences of 320×240 image resolution. We divided this sequence into temporal segments for most of the batch methods because of memory issue. Table 3 also presents the computational time for all of the compared methods. Simple-ReProCS and modified-PCP are the fastest methods in the provable methods category. Of these, simple-ReProCS also has the best performance. From the heuristics methods, OR-PCA-illum and GRASTA are even faster, but have bad performance.
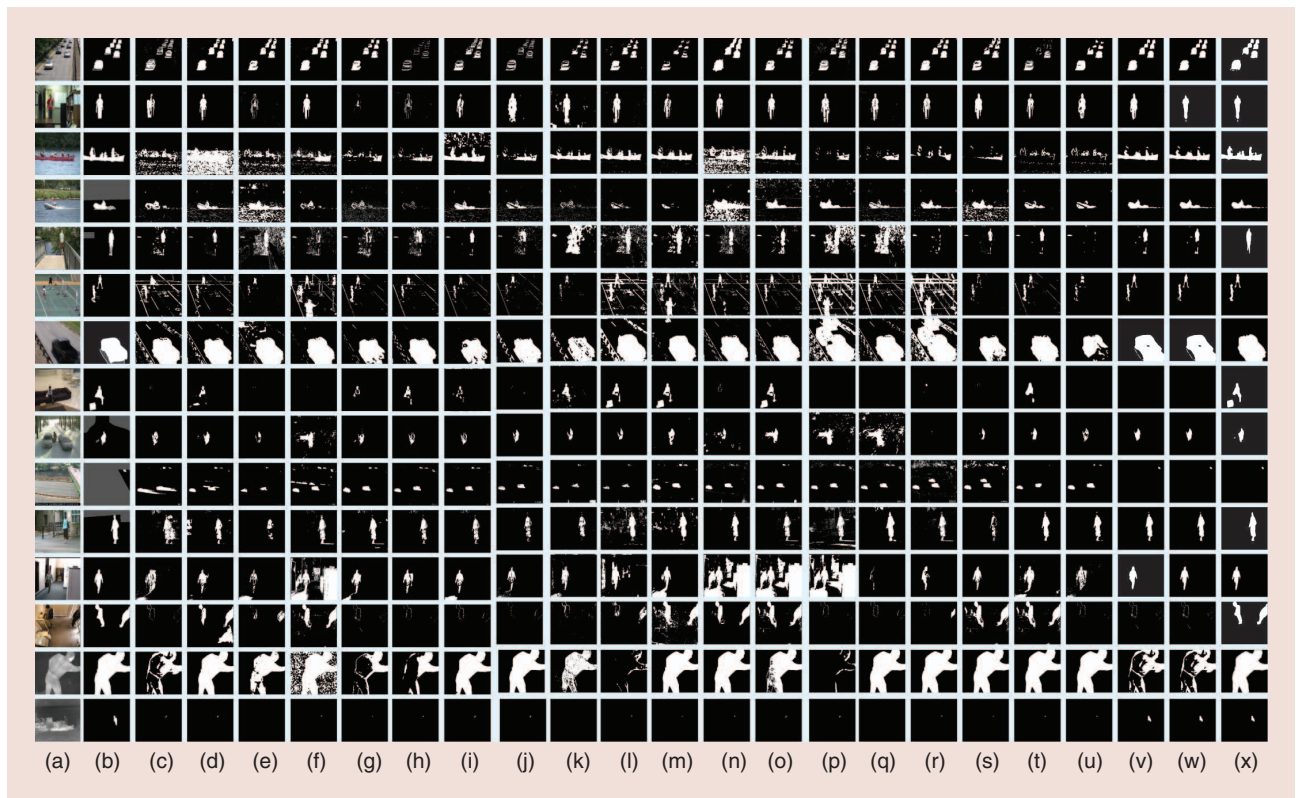

FIGURE 5. Comparison of the visual results obtained by the current state-of-the-art RPCA-based methods for background subtraction. From left to right: comparison of the qualitative results of the 15 input images from change detection data set are shown. (a) A set of 15 input images. (b) The ground truth of foreground objects. (c) Background subtraction estimated by RPCA via the PCP method. (d) GoDec. (e) RPMF. (f) RPCA-GD. (g) 3TD. (h) pROST. (i) incPCP. (j) RMAMR. (k) GRASTA. (l) ReProCS. (m) TVRPCA. (n) SRPCA. (o) NO-RMC. (p) LSD. (q) GOSUS. (r) OMoGMF+TV. (s) COROLLA. (t) OR-PCA-illum. (u) 2PRPCA. (v) DECOLOR. (w) GFL. (x) MSCL. From top to bottom: Rows 1 and 2: The sequences *highway* and *office* from the category *baseline*. Rows 3–5: The sequences *canoe, boat,* and *overpass* from the category *DB*. Rows 6 and 7: The sequences *badminton* and *traffic* from the category *camera jitter*. Rows 8–10: The sequences *sofa, winterDriveway,* and *streetLight* from the category *IOM*. Rows 11–13: The sequences *BackDoor, cubicle,* and *copyMachine* from the category *shadow*. Rows 14 and 15: The sequences *library* and *lakeside* from the category *thermal*.

The category *baseline* contained four simple videos where the background was always static in all of the video frames. Figure 5 (rows 1 and 2) showed that most of the compared methods in each category produced a good quality of foreground objects for these sequences. The *DB* category comprised six challenging videos (three of them were shown in rows 3–5 in Figure 5) depicting outdoor scenes. This was the most difficult among all categories for mounted camera object detection, which contained sequences exhibiting DB motions because of rippling of water surfaces and swaying of bushes. One scheme in the provable methods category (ReProCS), three in the heuristics methods category (3TD, 2PRPCA, PRMF), and many in the heuristics methods with specific constraints category estimated a better quality of foreground objects than all of the compared methods in these categories. Table 3 shows an average F_1 score of close to 80% and more than 80% for these methods because of the oversmoothing and spatiotemporal constraints enforced on these methods. In contrast, all other methods in these categories generated a noisy foreground segments because of highly DB regions. This experiment demonstrates that ReProCS can tolerate more background changes (larger rank r) for a given maximum outlier fraction per column (maximum support size of any foreground image).

The IOM category included six videos (three were shown in rows 8–10 in Figure 5), which contained ghosting artifacts in the detected motion. In these sequences, the moving foreground objects were motionless most of the time. This is a setting of large outlier fractions per row. As explained previously, all static RPCA methods that do not exploit slow subspace change will fail for this setting. This is indeed observed in our experiment. Most compared methods categories were not able to handle this challenge and obtained a low F_1 score (shown in Table 3). ReProCS and ReProCS-provable achieved the best performance among all methods (provable or not) that do not exploit extra problem-specific assumptions. It had an F_1 score of 70% since ReProCS does exploit the subspace dynamics. Only two methods in heuristics with additional constraints category (SRPCA and MSCL) were better than ReProCS, because they include specific heuristics to detect and remove motionless frames and they use spatiotemporal regularization in the low-rank background model.

The *shadows* category comprised six videos (some of them were shown in rows 11–13 in Figure 5) exhibiting both strong and faint shadows. For most of the compared methods in the provable and heuristics methods categories, these videos posed a great challenge (see Table 3). Provable methods such as RPCA-GD and ReProCS, heuristics methods such as PRMF, and many heuristic methods with additional constraints achieved promising performance as compared to other methods. We observed that some hard shadows on the ground were still a major limitation of the top performing algorithms. The *thermal* category comprised five sequences captured by

> **The RPCA problem has been extensively studied during the last seven to ten years. Dynamic RPCA or RST has received significant attention much more recently, and there are many unsolved important questions.**

the far-infrared camera as shown in rows 14 and 15 in Figure 5. Color saturation was the main challenge in this category, which degraded the performance of majority of the compared methods in each category (F_1 score less than 80% in Table 3). Only one provable method (ReProCS), and eight heuristics methods (ReProCS, 3TD, 2PRPCA, PRMF, OR-PCA-illum, MSCL, GOSUS, and COROLLA) were able to discriminate the background-foreground pixels effectively in the presence of color saturation. The *camera jitter* category comprised one indoor and three outdoor videos (rows 6 and 7 in Figure 5) where the background scene undergoes jitter induced motion throughout the video frames.

Comparison with nonrobust principal component analysis methods

We evaluated the RPCA methods against the top methods in the CDnet 2012 data set. We excluded the supervised methods based on deep learning because RPCA methods are unsupervised ones. We selected three top performing unsupervised methods called SuBSENSE [76], PAWCS [75], and LOBSTER [77]. For a fair comparison, we retested these methods with their source code publicly available either on the author's respective webpages or BGSLibrary, and we also reported the time taken by these methods on the same machine as the one used for the RPCA methods. From our comparisons, the F scores of SUBSENSE, PAWCS, and LOBSTER are a little bit lower than the ones reported in the CDnet 2012 data set. This is likely because we used the same set of parameters for all of the categories and the videos of the data set, while the authors of SuBSENSE, PAWCS, and LOBSTER have optimized the set of parameters for each case. In addition, the computational time is reported for these methods by including the training time. This is again done to keep comparisons with RPCA-based methods fair. On average, SuBSENSE, PAWCS, and LOBSTER reached an F-score equal to 0.79, 0.81, and 0.75, respectively, which is better than many RPCA methods. But they are also slower—the time taken is 2.8, 1.9, and 3.7 s/frame, respectively. This is much slower than many of the RPCA methods that also work well (are among the top five methods), e.g., ReProCS takes only 0.7 s/frame.

The reasons for both the accuracy and slow speed are that 1) SuBSENSE, PAWCS, and LOBSTER used many more color and texture features than RPCA methods; and 2) they use multiple cues (use additional process to deal with shadows and/or sudden illumination changes). SuBSENSE uses an advanced texture features called *spatiotemporal binary features* in addition to the color feature. For PAWCS, a background dictionary is iteratively updated using a local descriptor based on different low-level features. For LOBSTER, it is also based on a texture feature called *local binary patterns* to cope with sudden lighting variations in the background scenes.

Future directions

The RPCA problem has been extensively studied during the last seven to ten years. Dynamic RPCA or RST has received significant attention much more recently, and there are many unsolved important questions. The first question is: Can we obtain a guarantee for it under even weaker assumptions—can the lower bound on outlier magnitudes be relaxed? Two important extensions of RPCA are RMC and undersampled RPCA. While these have been studied, their dynamic extensions have received almost no attention. These remain important questions for very long data sets where a changing subspace assumption is a more appropriate one. There is an algorithm and a partial guarantee for the undersampled case in [18] and [78], but a complete correctness result still does not exist; and careful experimental evaluations of the proposed techniques on real dynamic MRI data sets are missing too.

> **An open question is: How can robust and dynamic RPCA ideas be successfully adapted to solve other more general related problems?**

A question motivated by video analytics is how can an RPCA and an RST solution for data from moving cameras be developed? To understand the problem, suppose that the camera motion corresponds to just x-y translation. Then, this means that the video data matrix M itself is not S+LR, but one needs to motion-compensate the video images in such way that the resulting new matrix $\tilde{M}$ is accurately modeled as S+LR. While many heuristics exist to address this issue, a simple provably correct and useful approach does not exist. Another related issue of practical importance is how can one reliably detect sudden subspace changes? For the video application, this would occur due to sudden scene changes (the camera being turned around, for example). Slow subspace change is easy to detect and reliably estimate (as explained previously), but sudden change results in a complete loss of track and hence the low-rank component also gets misclassified as outlier. One heuristic that works in practice is to use a sudden significant increase in outlier support size as an indicator of sudden scene change. It would be interesting to study if this is indeed a reliable indicator of sudden change.

Another interesting and practically relevant question is how can one develop a simple and provable RST solution that is streaming, i.e., it requires only one pass through the data and needs just enough memory to only store the output subspace estimate (an $n \times r$ matrix)? A streaming RPCA solution has been developed in a 2016 arXiv preprint, but it works only for one-dimensional RPCA. On the other hand, ReProCS is a nearly memory-optimal solution to dynamic RPCA [27], [29], but it requires more than one pass through the data (enough passes to solve SVD up to the error level at a given iteration).

An open question is: How can robust and dynamic RPCA ideas be successfully adapted to solve other more general related problems? One such problem is subspace clustering, which involves clustering a given data set into one of K different low-dimensional subspaces. This can be interpreted as a generaliza-tion of PCA that tries to represent a given data set using a single low-dimensional subspace. Another completely open question is whether one can solve the phaseless RPCA or S+LR problem. In many applications, such as ptychography, subdiffraction imaging, or astronomy, one can just acquire magnitude-only measurements. If the unknown signal or image sequence is well modeled as S+LR, can this modeling be exploited to recover it from undersampled phaseless measurements? Two precursors to this problem, low-rank phase retrieval [79] and phase retrieval for a single outlier-corrupted signal [80], have been recently studied. Finally, this article does not review the literature on deep learning-based approaches to RPCA, e.g., [81], nor does it overview the recent work on robust or dynamic RPCA for tensor data, e.g. [24]. All of these are active research areas with many open questions, but will hopefully be the foci of other review papers.

Authors

Namrata Vaswani (namrata@iastate.edu) received her B.Tech. degree from the Indian Institute of Technology (IIT-Delhi) in 1999 and her Ph.D. degree in 2004 from the University of Maryland, College Park. She is a professor of electrical and computer engineering, and (by courtesy) of mathematics, at Iowa State University. Her research interests lie at the intersection of statistical machine learning/data science, computer vision, and signal processing. Her recent research has focused on practicably useful and provably correct solutions for dynamic high-dimensional data problems such as dynamic compressive sensing (CS) and robust principal component analysis. She received the 2014 IEEE Signal Processing Society Best Paper Award for her modified-CS work that appeared in *IEEE Transactions on Signal Processing* in 2010. She is the special issues area editor of *IEEE Signal Processing Magazine*, has served twice as an associate editor of *IEEE Transactions on Signal Processing*, and is the lead guest editor of a *Proceedings of the IEEE Special Issue on Rethinking PCA for Modern Data Sets*.

Thierry Bouwmans (tbouwman@univ-lr.fr) obtained his B.Sc. degree in electronics from the University of Villeneuve d'Ascq, France, in 1990. He then obtained his master's degree in electronics in 1992. He received his Ph.D. degree in the field of three-dimensional computer vision in 1997 from the University of Littoral Côte d'Opale, Calais, France. Since 2000, he has been an associate professor at the University of La Rochelle, France. He obtained the diploma of HDR (Habilitation à Diriger des Recherches) for a full professor position in 2014. His research interests consist mainly in the detection of moving objects in challenging environments. He coauthored *Background Modeling and Foreground Detection for Video Surveillance and Robust Low-Rank* and *Sparse Matrix Decomposition: Applications in Image and Video Processing* (CRC Press). His research investigates the use of robust principal component analysis in video surveillance in particular.

He is also the main organizer of the Robust Subspace Learning and Applications in Computer Vision workshops hosted at the International Conference on Computer Vision in 2015 and 2017. He is a reviewer for many international journals and top-level conferences.

Sajid Javed (S.Javed.1@warwick.ac.uk) obtained his B.Sc. (honors) degree in computer science from University of Hertfordshire, United Kingdom, in 2010. He then joined the Virtual Reality Laboratory of Kyungpook National University, Republic of Korea, in 2012, where he completed his combined master's and doctoral degrees in computer science under the supervision of Prof. Soon Ki Jung and cosupervision of Prof. Thierry Bouwmans from MIA Lab, France, in 2017. He has been a postdoctoral research fellow in the Department of Computer Science, University of Warwick, United Kingdom, since October 2017. He actively works on computer vision and image analytics for cancer under the supervision of world leader Prof. Nasir Rajpoot at the Tissue Image Analytics Lab. He has coauthored approximately 30 publications including several journals and international conferences publications in the area of robust principal component analysis for background-foreground modeling. His other research interests are salient object detection, visual object tracking, semantic segmentation, subspace clustering, and social analysis of cancer cells.

Praneeth Narayanamurthy (pkurpadn@iastate.edu) obtained his B.Tech. degree in electrical and electronics engineering from the National Institute of Technology Karnataka, Surathkal, India, in 2014 and subsequently worked as a project assistant in the Department of Electrical Engineering of the Indian Institute of Science. Currently, he is a Ph.D. student in the Department of Electrical and Computer Engineering at Iowa State University, Ames. His research interests include the algorithmic and theoretical aspects of high-dimensional statistical signal processing, and machine learning. He is a Student Member of the IEEE.

References

[1] L. Xu and A. Yuille, "Robust principal component analysis by self-organizing rules based on statistical physics approach," *IEEE Trans. Neural Networks*, vol. 6, no. 1, pp. 131–143, Jan. 1995.

[2] F. De La Torre and M. J. Black, "A framework for robust subspace learning," *Int. J. Comput. Vis.*, vol. 54, no. 1–3, pp. 117–142, 2003.

[3] J. Wright and Y. Ma, "Dense error correction via l1-minimization," *IEEE Trans. Inform. Theory*, vol. 56, no. 7, pp. 3540–3560, 2010.

[4] E. J. Candès, X. Li, Y. Ma, and J. Wright, "Robust principal component analysis?" *J. ACM*, vol. 58, no. 3, p. 11, 2011.

[5] V. Chandrasekaran, S. Sanghavi, P. A. Parrilo, and A. S. Willsky, "Rank-sparsity incoherence for matrix decomposition," *SIAM J. Optim.*, vol. 21, no. 2, pp. 572–596, 2011.

[6] D. Hsu, S. M. Kakade, and T. Zhang, "Robust matrix decomposition with sparse corruptions," *IEEE Trans. Inform. Theory*, vol. 57, no. 11, pp. 7221–7234, Nov. 2011.

[7] C. Qiu, N. Vaswani, B. Lois, and L. Hogben, "Recursive robust PCA or recursive sparse recovery in large but structured noise," *IEEE Trans. Inform. Theory*, vol. 60, no. 8, pp. 5007–5039, Aug. 2014.

[8] P. Netrapalli, U. N. Niranjan, S. Sanghavi, A. Anandkumar, and P. Jain, "Non-convex robust PCA," in *Proc. Conf. Neural Information Processing Systems*, 2014, pp. 1107–1115.

[9] J. Zhan, B. Lois, H. Guo, and N. Vaswani, "Online (and offline) robust PCA: Novel algorithms and performance guarantees," in *Proc. Int. Conf. Artificial Intelligence and Statistics*, 2016, pp. 1488–1496.

[10] X. Yi, D. Park, Y. Chen, and C. Caramanis, "Fast algorithms for robust PCA via gradient descent," in *Proc. Conf. Neural Information Processing Systems*, 2016, pp. 4152–4160.

[11] C. Qiu and N. Vaswani, "Real-time robust principal components' pursuit," in *Proc. Allerton Conf. Communication Control and Computing*, 2010, pp. 591–598.

[12] H. Guo, C. Qiu, and N. Vaswani, "An online algorithm for separating sparse and low-dimensional signal sequences from their sum," *IEEE Trans. Signal Processing*, vol. 62, no. 16, pp. 4284–4297, 2014.

[13] T. Zhang and G. Lerman, "A novel m-estimator for robust PCA," *J. Mach. Learn. Res.*, vol. 54, no. 1, pp. 749–808, 2014.

[14] N. Vaswani and P. Narayanamurthy, "Robust principal component analysis (PCA) and matrix completion: A review," in *Proc. IEEE*, to be published.

[15] G. Lerman and T. Maunu, "An overview of robust subspace recovery," submitted for publication.

[16] Y. Chen, H. Xu, C. Caramanis, and S. Sanghavi, "Robust matrix completion and corrupted columns," in *Proc. Int. Conf. Machine Learning*, 2011, pp. 873–880.

[17] Y. Cherapanamjeri, K. Gupta, and P. Jain, "Nearly-optimal robust matrix completion," in *Proc. Int. Conf. Machine Learning*, 2016.

[18] C. Qiu and N. Vaswani, "Recursive sparse recovery in large but correlated noise," in *Proc. 48th Allerton Conf. Communication Control and Computing*, 2011, pp. 752–759.

[19] A. E. Waters, A. C. Sankaranarayanan, and R. G. Baraniuk, "SPARCS: Recovering low-rank and sparse matrices from compressive measurements," in *Proc. Neural Information Processing Systems*, 2011, pp. 1089–1097.

[20] J. Wright, A. Ganesh, K. Min, and Y. Ma, "Compressive principal component pursuit," arXiv Preprint, arXiv:1202.4596, 2013.

[21] M. Mardani, G. Mateos, and G. B. Giannakis, "Recovery of low-rank plus compressed sparse matrices with application to unveiling traffic anomalies," *IEEE Trans. Inform. Theory*, vol 59, no. 8, pp. 5186–5205, 2013.

[22] R. Otazo, E. Candès, and D. K. Sodickson, "Low-rank and sparse matrix decomposition for accelerated dynamic MRI with separation of background and dynamic components," *Magn. Reson. Med.*, vol. 73, no.3, pp. 1125–1136, 2013.

[23] M. Mardani, G. Mateos, and G. B. Giannakis, "Dynamic anomalography: Tracking network anomalies via sparsity and low rank," *IEEE J. Select. Topics Signal Processing*, vol 7, no. 1, pp. 50–66, Feb. 2013.

[24] A. Ozdemir, E. M. Bernat, and S. Aviyente, "Recursive tensor subspace tracking for dynamic brain network analysis," *IEEE Trans. Signal Inform. Processing over Networks*, vol 3, no. 4, pp. 669–682, 2017.

[25] H. Guo and N. Vaswani, "Video denoising via online sparse and low-rank matrix decomposition," in *Proc. IEEE Statistical Signal Processing Workshop*, 2016, pp. 1–5.

[26] A. Sobral, T. Bouwmans, and E. H. Zahzah, "LRS library: 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*, Eds. Boca Raton, FL: CRC, 2016, ch. 27.

[27] P. Narayanamurthy and N. Vaswani, "Nearly optimal robust subspace tracking," arXiv Preprint, arxiv:1712.06061, 2018.

[28] P. Narayanamurthy and N. Vaswani, "A fast and memory-efficient algorithm for robust PCA (MERoP)," in *Proc. IEEE Int. Conf. Acoustics Speech Signal Processing*, 2018.

[29] P. Narayanamurthy and N. Vaswani, "Provable dynamic robust PCA or robust subspace tracking," arXiv Preprint, arXiv:1705.08948, 2018.

[30] J. Zhan and N. Vaswani, "Robust PCA with partial subspace knowledge," *IEEE Trans. Signal Processing*, vol. 63, no. 12, pp. 3332–3347, July 2015.

[31] M. McCoy and J. Tropp, "Two proposals for robust PCA using semidefinite programming," *Electron. J. Statist.*, vol. 5, pp. 1123–1160, 2011.

[32] H. Xu, C. Caramanis, and S. Mannor, "Outlier-robust PCA: The high-dimensional case," *IEEE Trans. Inform. Theory*, vol. 59, no. 1, pp. 546–572, 2013.

[33] J. Feng, H. Xu, S. Mannor, and S. Yan, "Online pca for contaminated data," in *Proc. Conf. Advanced Neural Information Processing Systems*, 2013, pp. 764–772

[34] M. Hubert, P. Rousseeuw, and K. Vanden Branden, "ROBPCA: A new approach to robust principal component analysis," *Technometrics*, vol. 47, no. 1, pp. 64–79, Feb. 2005.

[35] Y. Li, L. Xu, J. Morphett, and R. Jacobs, "An integrated algorithm of incremental and robust PCA," in *Proc. IEEE Int. Conf. Image Processing*, 2003, pp. 245–248.

[36] D. Skocaj and A. Leonardis, "Weighted and robust incremental method for subspace learning," in *Proc. IEEE Int. Conf. Computer Vision*, Oct. 2003, vol. 2, pp. 1494–1501.

[37] T. Zhou and D. Tao, "GoDec: Randomized low-rank and sparse matrix decomposition in noisy case," in *Proc. Int. Conf. Machine Learning*, 2011.

[38] B. Yang, "Projection approximation subspace tracking," *IEEE Trans. Signal Processing*, vol. 43, no. 1, pp. 95–107, 1995.

[39] Y. Chi, Y. C. Eldar, and R. Calderbank, "Petrels: Parallel subspace estimation and tracking by recursive least squares from partial observations," *IEEE Trans. Signal Processing*, vol. 61, no. 23, pp. 5947–5959, Dec. 2013.

[40] L. Balzano, B. Recht, and R. Nowak, "Online identification and tracking of subspaces from highly incomplete information," in *Proc. Allerton Conf. Communication Control and Computing*, 2010, pp. 704–711.

[41] E. Candes, "The restricted isometry property and its implications for compressed sensing," *C. R. Math. Acad. Sci. Paris Ser. I*, vol. 235, no. 9–10, pp. 589–592, 2008.

[42] N. Vaswani and H. Guo, "Correlated-pca: Principal components' analysis when data and noise are correlated," in *Proc. Advanced Neural Information Processing Systems*, 2016, pp. 1768–1776.

[43] N. Vaswani and P. Narayanamurthy, "Finite sample guarantees for PCA in non-isotropic and data-dependent noise," in *Proc. Allerton Conf.*, 2017, 783–789.

[44] B. Lois and N. Vaswani, "Online matrix completion and online robust PCA," in *Proc. IEEE Int. Symp. Information Theory*, 2015, pp. 1826–830.

[45] N. Vaswani and W. Lu, "Modified-CS: Modifying compressive sensing for problems with partially known support," *IEEE Trans. Signal Processing*, vol. 58, no. 9, pp. 4595–4607, Sept. 2010.

[46] G. Mateos and G. Giannakis, "Robust PCA as bilinear decomposition with outlier-sparsity regularization," *IEEE Trans. Signal Processing*, vol. 60, no. 19, pp. 5176–5190, Oct. 2012.

[47] J. Feng, H. Xu, and S. Yan, "Online robust PCA via stochastic optimization," in *Proc. Neural Information Processing Systems*, 2013, pp. 404–412.

[48] S. Chouvardas, Y. Kopsinis, and S. Theodoridis, "An adaptive projected subgradient based algorithm for robust subspace tracking," in *Proc. IEEE Int. Conf. Acoustics, Speech, Signal Processing*, 2014, pp. 5497–5501.

[49] S. Chouvardas, Y. Kopsinis, and S. Theodoridis, "Robust subspace tracking with missing entries: A set–theoretic approach," *IEEE Trans. Signal Processing*, vol. 63, no. 19, pp. 5060–5070, 2015.

[50] J. He, L. Balzano, and A. Szlam, "Incremental gradient on the Grassmannian for online foreground and background separation in subsampled video," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2012, pp. 1568–1575.

[51] F. Seidel, C. Hage, and M. Kleinsteuber, "pROST: A smoothed lp-norm robust online subspace tracking method for realtime background subtraction in video," arXiv Preprint, arXiv:1302.2073, 2013.

[52] P. Rodriguez and B. Wohlberg, "Incremental principal component pursuit for video background modeling," *J. Math. Imaging Vis.*, vol. 55, no. 1, pp. 1–18, 2016.

[53] J. Xu, V. Ithapu, L. Mukherjee, J. Rehg, and V. Singh, "GOSUS: Grassmannian online subspace updates with structured-sparsity," in *Proc. Int. Conf. Computer Vision*, 2013, pp. 3376–3383.

[54] C. Qiu and N. Vaswani, "Support predicted modified-cs for recursive robust principal components' pursuit," in *Proc. IEEE Int. Symp. Inform. Theory*, 2011, pp. 668–672.

[55] N. Campell, "Robust procedures in multivariate analysis I: Robust covariance estimations," *Appl. Statist.*, vol. 29, pp. 231–237, 1980.

[56] P. Huber, *Robust Statistics*, New York: Wiley, 1981.

[57] C. Croux and A. Ruiz-Gazen, "A fast algorithm for robust principal components based on projection pursuit," in *Proc. Computational Statistics*, 1996, pp. 211–217.

[58] H. Xu, C. Caramanis, and S. Sanghavi, "Robust PCA via outlier pursuit," *IEEE Trans. Inform. Theory*, vol. 58, no. 5, pp. 2496–2504, May 2012.

[59] G. Lerman, M. B. McCoy, J. A. Tropp, and T. Zhang, "Robust computation of linear models by convex relaxation," *Found. Comput. Math.*, vol. 15, no. 2, pp. 363–410, 2015.

[60] J. Goes, T. Zhang, R. Arora, and G. Lerman, "Robust stochastic principal component analysis," in *Proc. Int. Conf. Artificial Intelligence and Statistics*, 2014, vol. 33, pp. 266–274.

[61] Y. Wang, P. M. Jodoin, F. Porikli, J. Konrad, Y. Benezeth, and P. Ishwar, "CDnet 2014: An expanded change detection benchmark dataset," in *Proc. IEEE Computer Vision and Pattern Recognition Workshops*, 2014, pp. 393–400.

[62] O. Oreifej, X. Li, and M. Shah, "Simultaneous video stabilization and moving object detection in turbulence," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 35, no. 2, pp. 450–462, 2013.

[63] Z. Gao, L. F. Cheong, and Y. X. Wang, "Block-sparse RPCA for salient motion detection," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 36, no. 10, pp. 1975–1987, 2014.

[64] N. Wang, T. Yao, J. Wang, and D. Y. Yeung, "A probabilistic approach to robust matrix factorization," in *Proc. European Conf. Computer Vision*, 2012, pp. 126–139.

[65] S. Javed, A. Mahmood, T. Bouwmans, and S. K. Jung, "Background-foreground modeling based on spatiotemporal sparse subspace clustering," *IEEE Trans. Image Processing*, vol 26, no. 12, pp. 5840–5854, 2017.

[66] X. Zhou, C. Yang, and W. Yu, "Moving object detection by detecting contiguous outliers in the low-rank representation," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 35, no. 3, pp. 597–610, 2013.

[67] X. Liu, G. Zhao, J. Yao, and C. Qi, "Background subtraction based on low-rank and structured sparse decomposition," *IEEE Trans. Image Processing*, vol. 24, no. 8, pp. 2502–2514, 2015.

[68] X. Cao, L. Yang, and X. Guo, "Total variation regularized RPCA for irregularly moving object detection under dynamic background," *IEEE Trans. Comput.*, vol. 46, no. 4, pp. 1014–1027, 2016.

[69] S. Javed, A. Mahmood, T. Bouwmans, and S. K. Jung, "Spatiotemporal low-rank modeling for complex scene background initialization," *IEEE Trans. Circuits Syst. Video Technol.*, 2016.

[70] X. Ye, J. Yang, X. Sun, K. Li, C. Hou, and Y. Wang, "Foreground-background separation from video clips via motion-assisted matrix restoration," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 25, no. 11, pp. 1721–1734, 2015.

[71] B. Xin, Y. Tian, Y. Wang, and W. Gao, "Background subtraction via generalized fused Lasso foreground modeling," in *Proc. Computer Vision and Pattern Recognition*, Boston, MA, 2015, pp. 4676–4684.

[72] S. Moein and Z. Hong, "COROLA: A sequential solution to moving object detection using low-rank approximation," *Comput. Vis. Image Understanding*, vol. 146, pp. 27–39, 2016.

[73] H. Yong, D. Meng, W. Zuo, and L. Zhang, "Robust online matrix factorization for dynamic background subtraction," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PP, no. 99, 2017.

[74] S. Javed, S. H. Oh, T. Bouwmans, and S. K. Jung, "Robust background subtraction to global illumination changes via multiple features-based online robust principal components analysis with Markov random field," *J. Electron. Imaging*, vol. 24, no. 4, pp. 043011–043011, 2015.

[75] S. Charles and R. Bergevin, "Universal background subtraction using word consensus models," *IEEE Trans. Image Processing*, vol. 25, no. 10, pp. 4768–4781, 2016.

[76] P.-L. St-Charles, G.-A. Bilodeau, and R. Bergevin, "Subsense: A universal change detection method with local adaptive sensitivity," *IEEE Trans. Image Processing*, vol. 24, no. 1, pp. 359–373, 2015.

[77] P.-L. St-Charles and G.-A. Bilodeau, "Improving background subtraction using local binary similarity patterns," in *Proc. IEEE Winter Conf. Applications Computer Vision*, 2014, pp. 509–515.

[78] B. Lois, N. Vaswani, and C. Qiu, "Performance guarantees for undersampled recursive sparse recovery in large but structured noise," in *Proc. IEEE Global Conf. Signal and Information Processing*, 2013, pp. 1061–1064.

[79] N. Vaswani, S. Nayer, and Y. C. Eldar, "Low rank phase retrieval," *IEEE Trans. Signal. Processing*, Aug. 2017, vol. 65, no. 15, pp. 4059–4074.

[80] H. Zhang, Y. Chi, and Y. Liang, "Provable non-convex phase retrieval with outliers: Median truncated wirtinger flow," arXiv Preprint, arXiv:1603.03805, 2016.

[81] P. P. Brahma, Y. She, S. Li, J. Li, and D. Wu, "Reinforced robust principal component pursuit," *IEEE Trans. Neural Networks Learning Syst.*, vol. 29, no. 5, pp. 1525–1538, 2017.

SP