Random Consensus Robust PCA

Daniel Pimentel-Alarcón & Robert Nowak
University *of* Wisconsin-Madison

Abstract

This paper presents R2PCA, a random consensus method for robust principal component analysis. R2PCA takes RANSAC's principle of using as little data as possible one step further. It iteratively selects small subsets of the data to identify *pieces* of the principal components, to then *stitch* them together. We show that if the principal components are in general position and the errors are sufficiently sparse, R2PCA will exactly recover the principal components with probability 1, in lieu of assumptions on coherence or the distribution of the sparse errors, and even under adversarial settings. R2PCA enjoys many advantages: it works well under noise, its computational complexity scales linearly in the ambient dimension, it is easily parallelizable, and due to its low sample complexity, it can be used in settings where data is so large it cannot even be stored in memory. We complement our theoretical findings with synthetic and real data experiments showing that R2PCA outperforms state-of-the-art methods in a broad range of settings.

1 Introduction

In many relevant applications one aims to find a low-dimensional subspace that approximates a large data matrix $\mathbf{M}$. Principal Component Analysis (PCA) is one of the most widely used techniques for this purpose. Unfortunately, a single grossly corrupted datum can severely compromise its performance. Hence there is a wide variety of approaches to make PCA robust. Examples include M-estimators [1], random sampling [2], influence function techniques [3], alternating minimization [4] and convex relaxations [5–10]. Other approaches use convex optimization methods on subsets of the data (e.g., full rows and full columns) to improve computational complexity [11, 12].

One of the most natural and widely used algorithms for robust estimation is random sample consensus (RANSAC) [2]. RANSAC is simple yet powerful. It is popular because it makes almost no assumptions about the data, and it does not require unrealistic conditions to succeed. It has theoretical guarantees, works well in practice, and has enjoyed many improvements since its inception, e.g., [13–15]. The RANSAC version of PCA iteratively selects a few columns in $\mathbf{M}$ to define a candidate subspace, until it identifies a subspace that agrees with other columns in $\mathbf{M}$. This will successfully identify the subspace that we are looking for, as long as $\mathbf{M}$ has enough uncorrupted columns.

But in many modern applications, such as image processing and networks data analysis, every column in $\mathbf{M}$ may have a few grossly corrupted entries. This makes all columns in $\mathbf{M}$ outliers, hence standard robust methods like RANSAC are no longer applicable. In this setting $\mathbf{M}$ can be better modeled as the sum of a low-rank matrix $\mathbf{L}$ and a sparse matrix $\mathbf{S}$ representing the corrupted entries. The goal is to identify the subspace U spanned by the columns in $\mathbf{L}$. This problem is often called robust PCA (RPCA) [6]. The last decades have seen great approaches to this problem [16], yet it remained unclear how to extend RANSAC's principles to this setting [3].

The main contribution of this paper is R2PCA: a random consensus algorithm for RPCA. R2PCA takes RANSAC's principle of using as little data as possible one step further. It iteratively selects small subsets of the entries in $\mathbf{M}$ to identify *pieces* of the subspace U. This process is repeated until we identify enough pieces to *stitch* them together and recover the whole subspace. The key idea behind R2PCA is that subspaces can be easily and efficiently recovered from a few of its canonical projections [17]. These canonical projections are the pieces that R2PCA aims to identify. See Figure 1 for some intuition.

Our main result shows that R2PCA will exactly recover the subspace that we are looking for with probabil-

Appearing in Proceedings of the 20^{th} International Conference on Artificial Intelligence and Statistics (AISTATS) 2017, Fort Lauderdale, Flordia, USA. JMLR: W&CP volume 54.

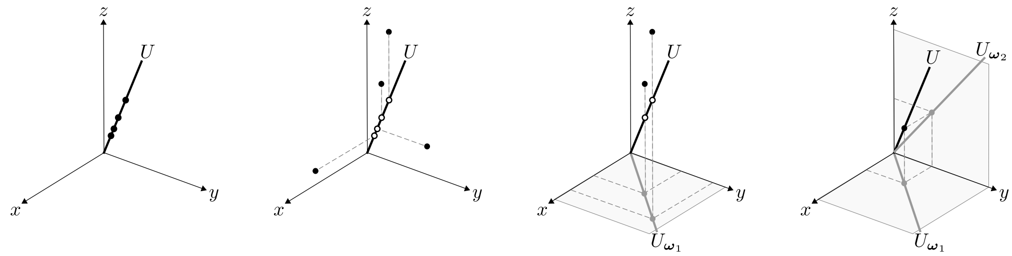

Figure 1: **First:** Each column in a rank-r matrix $\mathbf{L}$ corresponds to a point in an r-dimensional subspace U. In these figures, U is a 1-dimensional subspace (line) in general position, and the columns in $\mathbf{L}$ are drawn *generically* from U, that is, independently according to an absolutely continuous distribution with respect to the Lebesgue measure on U. For example, according to a gaussian distribution on U. **Second:** Adding $\mathbf{S}$ equates to corrupting some coordinates of some columns in $\mathbf{L}$. In this figure each point is corrupted in only one coordinate. As a result, the corrupted points no longer lie in U. So, how can we identify U from these corrupted points? **Third:** The key idea behind R2PCA is that since errors are sparse, if we only focus on a few coordinates of a few columns at a time, it is likely that the selected columns are uncorrupted on the selected coordinates. We can verify whether this is the case, because the projections of the selected columns onto the selected coordinates will agree if and only if the columns are uncorrupted in these coordinates. In this illustration, R2PCA only focused on the (x, y) coordinates and on two columns. The projections of both columns onto the (x, y) coordinates agree. Namely, they both lie in U_{ω_1}. Hence we can be sure that the (x, y) coordinates of these columns are uncorrupted, and that U_{ω_1} is actually equal to the projection of the subspace U that we aim to identify. **Last:** We can repeat this procedure for different sets of coordinates and columns, until we obtain enough projections to reconstruct the whole subspace.

ity 1, as long as $\mathbf{M}$ is generic, and the corrupted entries are sufficiently sparse. In contrast to popular convex relaxation methods (e.g., [5–12]), our results make no assumptions about coherence or the distribution of the sparse errors. In fact, our results hold even under adversarial settings where the errors are purposely located to complicate success. In its noisy variant, R2PCA can consistently estimate the desired subspace within the noise level. The computational complexity of R2PCA scales linearly in the ambient dimension. In addition, R2PCA enjoys many of RANSAC's advantages, and many of RANSAC's improvements can be easily adapted to R2PCA. For instance, R2PCA can run in parallel, with different computers searching for different pieces (canonical projections) of the subspace. This can greatly reduce computation time, which is of great interest in general, and paramount in real-time applications. Furthermore, R2PCA's principle of studying subspaces by pieces also improves computational and sample complexity. This is because R2PCA only uses small subsets of the data at a time. This is of particular importance in settings where $\mathbf{M}$ is so large it cannot even be stored in memory. We complement our theoretical findings with synthetic and real data experiments showing that R2PCA outperforms state-of-the-art methods, both in terms of speed and accuracy, in a broad range of settings.

2 Model and Main Results

Suppose we observe a $d \times n$ data matrix $\mathbf{M}$, given by

$$\mathbf{M} = \mathbf{L} + \mathbf{S}, \tag{1}$$

where $\mathbf{L}$ is rank-r and $\mathbf{S}$ is sparse. The goal is to identify the r-dimensional subspace U spanned by the columns of $\mathbf{L}$, or slightly more demanding, determine $\mathbf{S}$ and $\mathbf{L}$. Consider the following assumptions, where $\mathrm{Gr}(r, \mathbb{R}^d)$ denotes the Grassmannian manifold of r-dimensional subspaces in $\mathbb{R}^d$, and $\|\cdot\|_0$ denotes the ℓ_0-norm, given by the number of nonzero entries.

(**A1**) U is drawn according to an absolutely continuous distribution with respect to the uniform measure over $\mathrm{Gr}(r, \mathbb{R}^d)$.

(**A2**) The columns of $\mathbf{L}$ are drawn independently according to an absolutely continuous distribution with respect to the Lebesgue measure on U.

(**A3**) The nonzero entries in $\mathbf{S}$ are drawn independently according to an absolutely continuous distribution with respect to the Lebesgue measure on $\mathbb{R}^{\|\mathbf{S}\|_0}$.

(**A4**) $\mathbf{S}$ has at most $\frac{n-r}{2(r+1)^\alpha}$ nonzero entries per row and at most $\frac{d-r}{2(r+1)^{\alpha-1}}$ nonzero entries per column, with $\alpha \geq 1$.

A1 requires that U is a subspace in general position, and **A2** requires that the columns in $\mathbf{L}$ are drawn generically from this subspace. Together, **A1** and **A2** require that $\mathbf{L}$ is a generic rank-r matrix. Similarly, **A3** requires that $\mathbf{S}$ is a generic sparse matrix. See Section 4 for a further discussion about our assumptions and their relation to other typical assumptions from the literature.

A4 requires that $\mathbf{M}$ has at most $\mathcal{O}(n/r^{\alpha})$ corrupted entries per row and at most $\mathcal{O}(d/r^{\alpha-1})$ corrupted entries per column. Notice that since decomposing $\mathbf{M}$ is the same as decomposing $\mathbf{M}^{\mathsf{T}}$, assumption **A4** can be interchanged in terms of rows and columns. **A4** is a reasonable assumption because in most problems where this setting arises, $\mathbf{S}$ is sparse and $r \ll d, n$, whence **A4** allows a large number of corrupted entries in $\mathbf{M}$. The parameter α determines the sparsity level of $\mathbf{S}$, which in turn determines the computational complexity of R2PCA. The larger α, the sparser $\mathbf{S}$, and the faster R2PCA will succeed.

The main result of this paper is the next theorem. It states that R2PCA (Algorithm 1 below) will exactly recover U, $\mathbf{L}$ and $\mathbf{S}$ on $\mathcal{O}((d+n)2^{r^{2-\alpha}})$ iterations (linear in d and n). In the extreme case where $\mathbf{S}$ has too many errors ($\alpha = 1$), R2PCA could require exponential time in r. But if $\mathbf{S}$ is sufficiently sparse ($\alpha \geq 2$), R2PCA will succeed in linear time in r. This is true even in the adversarial setting where the errors are purposely located to complicate success. In other words, the computational complexity in Theorem 1 considers the worst-case scenario. So in practice, as shown in our experiments, R2PCA can be much faster and allow a much larger number of corrupted entries. The proof is given in Appendix A.

Theorem 1. *Let* **A1**-**A4** *hold. Let* $\hat{\mathbf{U}}$, $\hat{\mathbf{L}}$ *and* $\hat{\mathbf{S}}$ *be the output of* R2PCA. *Then* $\operatorname{span}\{\hat{\mathbf{U}}\} = U$, $\hat{\mathbf{L}} = \mathbf{L}$ *and* $\hat{\mathbf{S}} = \mathbf{S}$ *with probability* 1. *Furthermore, the expected number of iterations required by* R2PCA *is upper bounded by* $(d+n-r)2^{(r+1)^{2-\alpha}}$.

Remark 1. *Throughout the paper we assume that the rank r is known. If this is not the case, r can be estimated by iteratively selecting $(\tau+1)\times(\tau+1)$ minors in $\mathbf{M}$, and verifying their rank (or their singular value decomposition in the noisy setting). Under* **A1**-**A4**, *with probability* 1 *there will be a $(\tau+1)\times(\tau+1)$, rank-τ minor in $\mathbf{M}$ if and only if $r=\tau$. So we can start with $\tau = 1$. If we find a 2×2 minor in $\mathbf{M}$, we know that $r = 1$. If there exists no such minor, we know that $r \geq 2$. We can iteratively repeat this process until we find a $(\tau+1)\times(\tau+1)$ minor of rank-τ.* **A1**-**A4** *imply that if $r=\tau$, then for every $\boldsymbol{\omega} \subset \{1,\dots,d\}$ with $\tau+1$ entries, $\mathbf{M}_{\boldsymbol{\omega}}$ will contain a $(\tau+1)\times(\tau+1)$, rank-τ minor. Furthermore, using the same reasoning as in the proof of Theorem 1, one can show that on expectation, it would take no more than $\mathcal{O}(2^{r^{2-\alpha}})$ trials to find such a minor (recall that $\alpha \geq 1$ determines the sparsity level in $\mathbf{S}$).*

3 Algorithm

In this section we introduce R2PCA in its most basic setting (Algorithm 1). In Section 5 we discuss how to generalize it to noisy settings. From a high level perspective, R2PCA searches for small subsets of uncontaminated data in $\mathbf{M}$ to obtain *pieces* of U to then *stitch* them together. Once U is known, R2PCA searches for a few uncontaminated entries in each column of $\mathbf{M}$ to recover the coefficients of $\mathbf{L}$. Once $\mathbf{L}$ is known, $\mathbf{S}$ can be trivially recovered through (1).

The key idea behind R2PCA is that subspaces can be exactly and efficiently recovered from a few of its canonical projections [17]. So rather than attempting to identify U directly, we will aim to identify small projections of U, knowing that there is a simple way to *stitch* these projections together to recover U. More precisely, let $\boldsymbol{\Omega}$ be a $d \times N$ matrix with exactly $r+1$ nonzero entries per column, and let $\boldsymbol{\omega}_i \subset \{1,2,\dots,d\}$ index the nonzero entries in the i^{th} column of $\boldsymbol{\Omega}$. $\boldsymbol{\omega}_i$ indicates the canonical coordinates involved in the i^{th} projection that we will aim to identify. For any subspace, matrix or vector that is compatible with $\boldsymbol{\omega}_i$, we will use the subscript $\boldsymbol{\omega}_i$ to denote its restriction to the coordinates/rows in $\boldsymbol{\omega}_i$. For example, $\mathbf{M}_{\boldsymbol{\omega}_i} \in \mathbb{R}^{(r+1)\times n}$ denotes the restriction of $\mathbf{M}$ to the rows in $\boldsymbol{\omega}_i$, and $U_{\boldsymbol{\omega}_i} \subset \mathbb{R}^{r+1}$ denotes the projection of U onto the coordinates in $\boldsymbol{\omega}_i$.

Our goal is to identify a collection of projections $\{U_{\boldsymbol{\omega}_i}\}_{i=1}^N$ such that U can be recovered from these projections. Whether this is the case depends on the $\boldsymbol{\omega}_i$'s, i.e., on $\boldsymbol{\Omega}$. Fortunately, Theorem 1 in [17] specifies the conditions on $\boldsymbol{\Omega}$ to guarantee that U can be recovered from $\{U_{\boldsymbol{\omega}_i}\}_{i=1}^N$. To present this result, let us introduce the matrix $\mathbf{A}$. **A1** implies that with probability 1, $U_{\boldsymbol{\omega}_i}$ is a hyperplane, i.e., an r-dimensional subspace in $\mathbb{R}^{r+1}$. As such, it is characterized by its orthogonal direction, which we will call $\mathbf{a}_{\boldsymbol{\omega}_i}$. More precisely, let $\mathbf{a}_{\boldsymbol{\omega}_i} \in \mathbb{R}^{r+1}$ be a nonzero vector in $\ker U_{\boldsymbol{\omega}_i}$, and let $\mathbf{a}_i$ be the vector in $\mathbb{R}^d$ with the entries of $\mathbf{a}_{\boldsymbol{\omega}_i}$ in the locations of $\boldsymbol{\omega}_i$, and zeros elsewhere. Let $\mathbf{A}$ be the $d \times N$ matrix formed with $\{\mathbf{a}_i\}_{i=1}^N$ as columns. This way, $\mathbf{A}$ encodes the information of the projections $\{U_{\boldsymbol{\omega}_i}\}_{i=1}^N$. With this, we are ready to present Theorem 1 in [17], which we restate here as Lemma 1 with some adaptations to our context.

Lemma 1 (Theorem 1 in [17])**.** *Let* **A1** *hold. With probability* 1, $U = \ker \mathbf{A}^{\mathsf{T}}$ *if and only if*

(i) *There is a matrix $\boldsymbol{\Omega}'$ formed with $d-r$ columns of $\boldsymbol{\Omega}$, such that every matrix formed with a subset of η columns in $\boldsymbol{\Omega}'$ has at least $\eta + r$ nonzero rows.*

There exist plenty of matrices $\mathbf{\Omega}$ satisfying (i). For example:

$$\mathbf{\Omega} \;=\; \underbrace{\begin{bmatrix} \ddots & & \mathbf{0} \\ & \mathbf{1} & \\ \mathbf{0} & & \ddots \end{bmatrix}}_{d-r} \begin{matrix} \}\, d-(r+1) \\ \\ \}\, r+1, \end{matrix}$$

where $\mathbf{1}$ and $\mathbf{0}$ denote blocks of all 1's and all 0's. One may easily verify that $\mathbf{\Omega}$ satisfies condition (i) by taking $\mathbf{\Omega}' = \mathbf{\Omega}$. Notice that $\mathbf{A}$ is sparse (it only has $r+1$ nonzero entries per column), so computing $\ker \mathbf{A}^\mathsf{T}$ can be done efficiently.

Lemma 1 implies that $N = d - r$ projections are necessary and sufficient to recover U. Furthermore, it tells us which projections to look for, and how to reconstruct U from these projections. Hence, our strategy will be to select a $d \times (d-r)$ matrix $\mathbf{\Omega}$ satisfying (i), then identify the projections $\{U_{\boldsymbol{\omega}_i}\}_{i=1}^{d-r}$, and finally construct $\mathbf{A}$ according to these projections to recover U as $\ker \mathbf{A}^\mathsf{T}$.

In principle, our strategy to identify each projection is very similar to RANSAC. Given $\boldsymbol{\omega}_i$, we iteratively select $r+1$ columns of $\mathbf{M}_{\boldsymbol{\omega}_i}$ uniformly at random. Let $\mathbf{M}'_{\boldsymbol{\omega}_i} \in \mathbb{R}^{(r+1)\times(r+1)}$ be the matrix formed with the selected columns. $\operatorname{span}\{\mathbf{M}'_{\boldsymbol{\omega}_i}\}$ defines a candidate projection of U onto $\boldsymbol{\omega}_i$. **A1-A3** imply that with probability 1, $\operatorname{span}\{\mathbf{M}'_{\boldsymbol{\omega}_i}\} = U_{\boldsymbol{\omega}_i}$ if and only if $\mathbf{M}'_{\boldsymbol{\omega}_i}$ has no corrupted entries. This will be the case if and only if $\operatorname{rank}(\mathbf{M}'_{\boldsymbol{\omega}_i}) = r$. We will thus verify whether $\operatorname{rank}(\mathbf{M}'_{\boldsymbol{\omega}_i}) = r$. If this is not the case, this candidate projection will be discarded, and we will try a different $\mathbf{M}'_{\boldsymbol{\omega}_i}$. On the other hand, if $\operatorname{rank}(\mathbf{M}'_{\boldsymbol{\omega}_i}) = r$, then we know that $\operatorname{span}\{\mathbf{M}'_{\boldsymbol{\omega}_i}\}$ is the projection $U_{\boldsymbol{\omega}_i}$ that we were looking for. In this case, we can construct $\mathbf{a}_i$ as before. This process is repeated for each column $\boldsymbol{\omega}_i$ in $\mathbf{\Omega}$ to obtain $\mathbf{A}$. Since $\mathbf{\Omega}$ satisfies (i), we know by Lemma 1 that at the end of this procedure we will have enough projections to reconstruct U as $\dim\ker \mathbf{A}^\mathsf{T}$.

At this point, we have already recovered U. Let $\mathbf{U} \in \mathbb{R}^{d\times r}$ be a basis of U. We will now estimate the coefficient matrix $\mathbf{\Theta} \in \mathbb{R}^{r\times n}$ such that $\mathbf{L} = \mathbf{U}\mathbf{\Theta}$. Let $\mathbf{m}$ be a column in $\mathbf{M}$, and let $\boldsymbol{\omega}$ be a subset of $\{1, 2, \ldots, d\}$ with exactly $r+1$ elements. **A1-A3** imply that with probability 1, $\mathbf{m}_{\boldsymbol{\omega}}$ will lie in $U_{\boldsymbol{\omega}}$ if and only if all entries in $\mathbf{m}_{\boldsymbol{\omega}}$ are uncorrupted. We will thus iteratively select a set $\boldsymbol{\omega}$ indexing $r+1$ random entries in $\mathbf{m}$ until we find an $\boldsymbol{\omega}$ such that $\mathbf{m}_{\boldsymbol{\omega}} \in U_{\boldsymbol{\omega}}$. Once we find such $\boldsymbol{\omega}$, the coefficient vector of the corresponding column in $\mathbf{L}$ will be given by $\boldsymbol{\theta} := (\mathbf{U}_{\boldsymbol{\omega}}^\mathsf{T}\mathbf{U}_{\boldsymbol{\omega}})^{-1}\mathbf{U}_{\boldsymbol{\omega}}^\mathsf{T}\mathbf{m}_{\boldsymbol{\omega}}$. This process will be repeated for every column in $\mathbf{M}$ to obtain the coefficient matrix $\mathbf{\Theta}$, which together with $\mathbf{U}$ determine $\mathbf{L}$ as $\mathbf{U}\mathbf{\Theta}$. Once $\mathbf{L}$ is known, one can trivially recover $\mathbf{S}$ through (1). R2PCA is summarized in Algorithm 1.

Algorithm 1: Random Robust PCA (R2PCA)

Input: Data matrix $\mathbf{M} \in \mathbb{R}^{d\times n}$, rank r,
  matrix $\mathbf{\Omega} \in \{0,1\}^{d\times(d-r)}$ satisfying (i).
PART 1: Estimate $\mathbf{U}$
  for $i = 1, 2, \ldots, d-r$ **do**
    $\boldsymbol{\omega}_i$ = indices of the $r+1$ nonzero rows of the i^{th} column in $\mathbf{\Omega}$.
    repeat
      $\mathbf{M}'_{\boldsymbol{\omega}_i} \in \mathbb{R}^{(r+1)\times(r+1)} = r+1$ columns of $\mathbf{M}_{\boldsymbol{\omega}_i}$, selected randomly.
    until $\operatorname{rank}(\mathbf{M}'_{\boldsymbol{\omega}_i}) = r$.
    $\mathbf{a}_{\boldsymbol{\omega}_i} \in \mathbb{R}^{r+1}$ = nonzero vector in $\ker \mathbf{M}'^{\mathsf{T}}_{\boldsymbol{\omega}_i}$.
    $\mathbf{a}_i \in \mathbb{R}^d$ = vector with the entries of $\mathbf{a}_{\boldsymbol{\omega}_i}$ in the locations of $\boldsymbol{\omega}_i$, and zeros elsewhere.
  $\mathbf{A} \in \mathbb{R}^{d\times(d-r)} = [\mathbf{a}_1\ \mathbf{a}_2\ \cdots \mathbf{a}_{d-r}]$.
  $\hat{\mathbf{U}} \in \mathbb{R}^{d\times r}$ = basis of $\ker \mathbf{A}^\mathsf{T}$.
PART 2: Estimate $\mathbf{\Theta}$
  for each column $\mathbf{m}$ in $\mathbf{M}$ **do**
    repeat
      $\boldsymbol{\omega}$ = subset of $\{1, 2, \ldots, d\}$ with $r+1$ elements, selected randomly.
    until $\mathbf{m}_{\boldsymbol{\omega}} \in \operatorname{span}\{\hat{\mathbf{U}}_{\boldsymbol{\omega}}\}$.
    $\hat{\boldsymbol{\theta}} = (\hat{\mathbf{U}}_{\boldsymbol{\omega}}^\mathsf{T}\hat{\mathbf{U}}_{\boldsymbol{\omega}})^{-1}\hat{\mathbf{U}}_{\boldsymbol{\omega}}^\mathsf{T}\mathbf{m}_{\boldsymbol{\omega}}$.
    Insert $\hat{\boldsymbol{\theta}}$ into $\hat{\mathbf{\Theta}}$.
Output: $\hat{\mathbf{U}}$, $\hat{\mathbf{L}} = \hat{\mathbf{U}}\hat{\mathbf{\Theta}}$, $\hat{\mathbf{S}} = \mathbf{M} - \hat{\mathbf{L}}$.

4 More about our Assumptions

Essentially, **A1-A3** require that $\mathbf{M}$ is a combination of a generic sparse matrix, and a generic low-rank matrix. This discards pathological cases, like matrices with identical columns or exact-zero entries. Examples of these cases could arise in unnatural, cartoon-like images.

However, **A1-A3** allow realistic cases, like natural images. For instance, backgrounds in natural images can be highly structured but are not perfectly constant, as there is always some degree of natural variation that is reasonably modeled by an absolutely continuous (but possibly highly inhomogeneous) distribution. For example, the sky in a natural image might be strongly biased towards blue values, but each sky pixel will have at least small variations that will make the sky not perfectly constant blue. So while these are structured images,these variations make them generic enough so that our theoretical results are applicable. This is confirmed in our real data experiments.

Furthermore, because absolutely continuous distributions may be strongly inhomogeneous, they can be used to represent highly coherent matrices (that is, matrices whose underlying subspace is highly aligned with the canonical axes). Previous theory and methods for RPCA cannot handle some of the highly coherent cases that our new theory covers and that our new algorithm handles well (as demonstrated in our experiments).

We point out that **A1-A3** do not imply coherence nor vice-versa. For example, coherence assumptions indeed allow some identical columns, or exact-zero entries. However, they rule-out cases that our theory allows. For example, consider a case where a few rows of $\mathbf{U}$ are drawn i.i.d. $\mathcal{N}(0,\sigma_1^2)$ and many rows of $\mathbf{U}$ are drawn i.i.d. $\mathcal{N}(0,\sigma_2^2)$, with $\sigma_1 \gg \sigma_2$. This is a good model for some microscopy and astronomical applications that have a few high-intensity pixels, and many low-intensity pixels. Such $\mathbf{U}$ would yield a highly coherent matrix, which existing theory and algorithms cannot handle, while our results can (this can be confirmed in our experiments).

To sum up, our assumptions are different, not stronger nor weaker than the usual coherence assumptions, and we believe they are also more reasonable in many practical applications.

5 Handling Noise

In practice, all entries in $\mathbf{M}$ may be noisy, even the ones that are not corrupted by gross errors. We can model this as

$$\mathbf{M} \;=\; \mathbf{L}+\mathbf{S}+\mathbf{W}, \tag{2}$$

where $\mathbf{L}$ and $\mathbf{S}$ are as before, and $\mathbf{W}$ represents a noise matrix. The goal is the same as before: determine $\mathbf{L}$ and $\mathbf{S}$ from $\mathbf{M}$.

Recall that R2PCA's goal is to identify the projections $\{U_{\boldsymbol{\omega}_i}\}_{i=1}^{d-r}$ to then reconstruct U. In the noiseless setting, we do this by iteratively selecting $(r+1)\times(r+1)$ matrices $\mathbf{M}'_{\boldsymbol{\omega}_i}$, and checking their rank. If $\text{rank}(\mathbf{M}'_{\boldsymbol{\omega}_i}) = r$, then we know that all entries in $\mathbf{M}'_{\boldsymbol{\omega}_i}$ are uncorrupted, whence $U_{\boldsymbol{\omega}_i}$ is given by $\text{span}\{\mathbf{M}'_{\boldsymbol{\omega}_i}\}$. But in the noisy setting, $\text{rank}(\mathbf{M}'_{\boldsymbol{\omega}_i}) = r+1$ in general, regardless of whether these columns are corrupted by gross errors. Hence we cannot determine directly whether the columns in $\mathbf{M}'_{\boldsymbol{\omega}_i}$ are uncorrupted by simply checking whether $\text{rank}(\mathbf{M}'_{\boldsymbol{\omega}_i}) = r$, as we did in the noiseless setting. Instead, we can check the $(r+1)^{\text{th}}$ singular value of $\mathbf{M}'_{\boldsymbol{\omega}_i}$, which we will denote as λ_{r+1}. If λ_{r+1} is above the noise level, it is likely that at least one entry in $\mathbf{M}'_{\boldsymbol{\omega}_i}$ is grossly corrupted. On the other hand, if λ_{r+1} is within the noise level, it is likely that $\mathbf{M}'_{\boldsymbol{\omega}_i}$ has no grossly corrupted entries, whence we can use the subspace spanned by the r leading singular vectors of $\mathbf{M}'_{\boldsymbol{\omega}_i}$ as an estimator of $U_{\boldsymbol{\omega}_i}$. Unfortunately, since $\mathbf{M}'_{\boldsymbol{\omega}_i}$ only has $r+1$ rows and columns, the singular values and vectors of $\mathbf{M}'_{\boldsymbol{\omega}_i}$ will have a large variance. This means that λ_{r+1} will be above the noise level for many uncorrupted matrices $\mathbf{M}'_{\boldsymbol{\omega}_i}$, and below the noise level for many corrupted ones. As a result, we could miss good estimators and use bad ones. Furthermore, even if $\mathbf{M}'_{\boldsymbol{\omega}_i}$ is uncorrupted, the subspace spanned by its r leading singular vectors could be far from $U_{\boldsymbol{\omega}_i}$. As a result, our estimate of U could be very inaccurate.

But this can be improved if we use a few more entries in $\mathbf{M}$ so that the noise cancels out. To this end, let $\boldsymbol{\kappa}_i$ be a subset of $\{1,2,\dots,d\}$ with $k>r$ elements containing $\boldsymbol{\omega}_i$, and let $\mathbf{M}'_{\boldsymbol{\kappa}_i} \in \mathbb{R}^{k\times k}$ be a matrix formed with k columns of $\mathbf{M}_{\boldsymbol{\kappa}_i}$. Define $\mathbf{V}_{\boldsymbol{\kappa}_i} \in \mathbb{R}^{k\times r}$ as the matrix formed with the r leading left singular vectors of $\mathbf{M}'_{\boldsymbol{\kappa}_i}$. Under mild, typical assumptions (e.g., finite second and fourth moments), if $\mathbf{M}_{\boldsymbol{\kappa}_i}$ is uncorrupted, as k grows, the $(r+1)^{\text{th}}$ singular value of $\mathbf{M}'_{\boldsymbol{\kappa}_i}$ converges to the noise level, and $\text{span}\{\mathbf{V}_{\boldsymbol{\kappa}_i}\}$ converges to $U_{\boldsymbol{\kappa}_i}$. In other words, the larger k, the better estimates of U we will obtain. On the other hand, as k grows, it is more likely that at least one entry in $\mathbf{M}'_{\boldsymbol{\kappa}_i}$ is grossly corrupted (because $\mathbf{M}'_{\boldsymbol{\kappa}_i}$ will have more entries, each of which may be grossly corrupted), contrary to what we want. We thus want k to be large enough such that $\mathbf{M}'_{\boldsymbol{\kappa}_i}$ can be used to accurately estimate $U_{\boldsymbol{\kappa}_i}$, but not so large that there are no matrices $\mathbf{M}'_{\boldsymbol{\kappa}_i}$ with uncorrupted entries. The fraction of corrupted entries in $\mathbf{M}$ determines how large k can be. Figure 3 in Section 6 shows the feasible range of k as a function of the fraction of corrupted entries in $\mathbf{M}$.

Since $\mathbf{M}'_{\boldsymbol{\kappa}_i}$ has $k>r$ rows, if $\mathbf{M}'_{\boldsymbol{\kappa}_i}$ is believed to be uncorrupted, we can use it to estimate several projections of U (as many as $k-r$). To see this, let $\boldsymbol{\upsilon}_i$ be a subset of $\boldsymbol{\omega}_i$ with exactly r elements. Let $j \in \boldsymbol{\kappa}_i \backslash \boldsymbol{\upsilon}_i$, and let $\boldsymbol{\omega}_{ij} := \boldsymbol{\upsilon}_i \cup j$. Observe that, $\mathbf{V}_{\boldsymbol{\omega}_{ij}} \in \mathbb{R}^{(r+1)\times r}$ gives an estimate of $U_{\boldsymbol{\omega}_{ij}}$ through $\text{span}\{\mathbf{V}_{\boldsymbol{\omega}_{ij}}\}$. As before, we will store this information in the matrix $\mathbf{A}$. More precisely, for each $j \in \boldsymbol{\kappa}_i \backslash \boldsymbol{\upsilon}_i$, we will take a nonzero vector $\mathbf{a}_{\boldsymbol{\omega}_{ij}} \in \ker \mathbf{V}^{\mathsf{T}}_{\boldsymbol{\omega}_{ij}}$, and we will construct the vector $\mathbf{a}_{ij} \in \mathbb{R}^d$ with the entries of $\mathbf{a}_{\boldsymbol{\omega}_{ij}}$ in the locations of $\boldsymbol{\omega}_{ij}$. This time, $\mathbf{A}$ will be the matrix with the $\mathbf{a}_{ij}$'s as columns. Since $\boldsymbol{\omega}_i = \boldsymbol{\omega}_{ij}$ for some j, Lemma 1 suggests that the projections encoded in $\mathbf{A}$ should be enough to reconstruct $\mathbf{U}$. We can thus use the matrix $\hat{\mathbf{U}} \in \mathbb{R}^{d\times r}$ formed with the last r left singular vectors of $\mathbf{A}$ (which approximates $\ker \mathbf{A}^{\mathsf{T}}$) to estimate of U.

Similarly, in the second part of R2PCA we can estimate the coefficients of $\mathbf{L}$ using k entries of each column in $\mathbf{M}$. More precisely, for each column $\mathbf{m}$ in $\mathbf{M}$, we can iteratively select a set $\boldsymbol{\kappa}$ indexing k random entries in $\mathbf{m}$ until we find a $\boldsymbol{\kappa}$ such that $\mathbf{m}_{\boldsymbol{\kappa}}$ is close to $\text{span}\{\hat{\mathbf{U}}_{\boldsymbol{\kappa}}\}$

(within the noise level). If this is the case, it is likely that the entries in $\mathbf{m}_{\boldsymbol{\kappa}}$ are uncorrupted, and that $\hat{\boldsymbol{\theta}} = (\hat{\mathbf{U}}_{\boldsymbol{\kappa}}^\mathsf{T}\hat{\mathbf{U}}_{\boldsymbol{\kappa}})^{-1}\hat{\mathbf{U}}_{\boldsymbol{\kappa}}^\mathsf{T}\mathbf{m}_{\boldsymbol{\kappa}}$ is a good estimate of the coefficient we are looking for. We repeat this process to obtain an estimate $\hat{\boldsymbol{\Theta}}$ of $\boldsymbol{\Theta}$. Finally, our estimate of $\mathbf{L}$ is given by $\hat{\mathbf{U}}\hat{\boldsymbol{\Theta}}$, which in turn gives an estimate of $\mathbf{S}$ through (1). The noisy variant of R2PCA is summarized in Algorithm 2 in Appendix B.

6 Experiments

In this section we present a series of experiments to study the performance of R2PCA and compare it with the augmented Lagrange multiplier method for robust PCA(RPCA-ALM) [18, 19]. We found, consistent with previous reports [9, 16, 20], that the RPCA-ALM algorithm typically performed as well or better than several others (e.g., singular value thresholding [7], alternating direction method [21], accelerated proximal gradient [22] and dual method [22]).

Synthetic Data. We will first use simulations to study the performance of R2PCA in the noiseless setting, as a function of the percentage of grossly corrupted entries per row p, and the coherence of $\mathbf{L}$, defined as $\mu := \frac{d}{r}\max_{1\leq i\leq d}\|\mathbf{P}_U\mathbf{e}_i\|_2^2$, where $\mathbf{P}_U$ denotes the projection operator onto U, and $\mathbf{e}_i$ the i^{th} canonical vector in $\mathbb{R}^d$. Intuitively, μ parametrizes how aligned is U with respect to the canonical axes. In all our experiments, $\mathbf{L}$ was a $d\times n$, rank-r matrix, with $d=n=100$ and $r=5$.

In our simulations, we first generated a $d\times r$ random matrix $\mathbf{U}$ with $\mathcal{N}(0,1)$ i.i.d. entries to use as basis of U. To obtain matrices with a specific coherence parameter, we simply increased the magnitude of a few entries in $\mathbf{U}$, until it had the desired coherence. We then generated an $r\times(r+1)(d-r)$ random matrix $\boldsymbol{\Theta}$, also with $\mathcal{N}(0,1)$ i.i.d. entries, to use as coefficient vectors. With this, we constructed $\mathbf{L}=\mathbf{U}\boldsymbol{\Theta}$. Next, we generated a $d\times r$ matrix $\mathbf{S}$ with p percent of nonzero entries per row, selected uniformly at random. The nonzero entries in $\mathbf{S}$ are i.i.d. $\mathcal{N}(0,10)$. Finally, we obtained $\mathbf{M}$ as in (1). We repeated this experiment 100 times for each pair (p,μ), and recorded the fraction of trials that $\mathbf{L}$ was exactly recovered. We declared a success if the normalized error (using Frobenius norm) was below 10^{-10} after at most 10^3 seconds. The results are summarized in Figure 2. As predicted by our theory, R2PCA performs perfectly as long as $\mathbf{S}$ is sufficiently sparse, regardless of coherence. In contrast, other approaches rely on low coherence (e.g., [5–12]), and one can see that their performance quickly decays as coherence increases. On the other hand, as p grows, and $\mathbf{S}$ becomes less sparse, the likelihood of finding uncorrupted blocks in $\mathbf{M}$ quickly decays. In turn, it takes more time to identify projections of U, up to the point where R2PCA is unable to identify enough projections to reconstruct U. Our astronomy and real-data experiments below illustrate the importance of coherent matrices and non uniformly distributed errors.

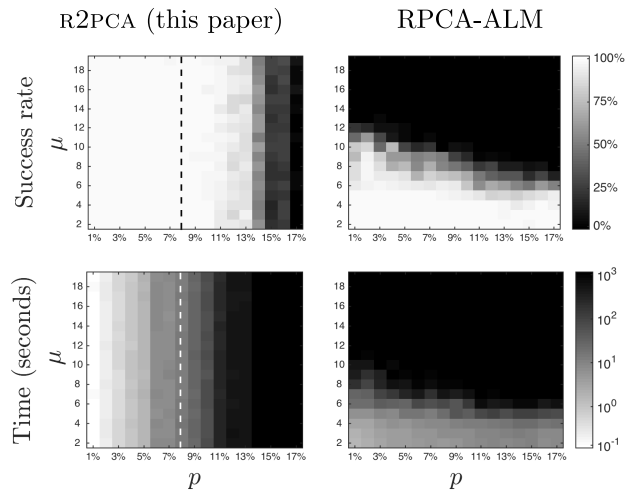

Figure 2: Transition diagrams of the success rate (**top row**) and time (**bottom row**) for exact recovery of $\mathbf{L}$ as a function of the percentage of grossly corrupted entries per row p, and the coherence parameter $\mu\in[1, d/r]$. The color of each (p,μ) pixel indicates the average over 100 trials (the lighter the better). Notice that as p grows, so does the time required to find projections, up to the point where R2PCA is unable to find enough projections to reconstruct U. Theorem 1 shows that if $\mathbf{M}$ has at most $p=7.9\%$ corrupted entries per row (dashed line), then R2PCA can exactly recover $\mathbf{L}$. We point out that our results hold regardless of coherence, as opposed to other algorithms, whose performance quickly decays as coherence increases.

In Section 5 we presented a noisy variant of R2PCA that iteratively selects k rows of k columns of $\mathbf{M}$ to estimate U and $\boldsymbol{\Theta}$. Its performance depends on the choice of k. If k is too small, our estimates could be very inaccurate. If k is too large, $\mathbf{M}$ may not contain enough $k\times k$ uncorrupted blocks to obtain an estimate. The feasible range of k depends on the percentage of corrupted entries p. In our next experiment we study the performance of the noisy variant of R2PCA as a function of p and k. To obtain a noisy $\mathbf{M}$ according to (2), we generated matrices $\mathbf{L}$ and $\mathbf{S}$ as described before, and then added a $d\times n$ random matrix $\mathbf{W}$ with $\mathcal{N}(0,\sigma^2)$ i.i.d. entries. To measure accuracy we recorded the error of the estimated $\mathbf{L}$ after at most 10^3 seconds (using normalized Frobenius norm). We repeated this experiment 100 times for each pair (p,k) with $\sigma=10^{-3}$ fixed. The results, summarized in Figure 3, show the feasible range of k.

In our next simulation, we selected $k=2r$, known from our previous experiment to produce reasonable results for a wide range of p, and used it to test the perfor-

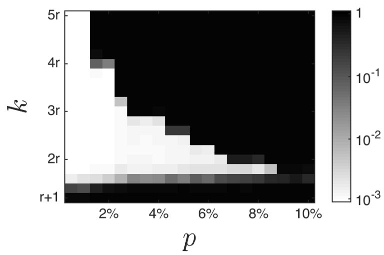

Figure 3: Transition diagram of the estimation error of $\mathbf{L}$ (using the noisy variant of R2PCA) as a function of the percentage of grossly corrupted entries per row p and the parameter k, with noise level $\sigma = 10^{-3}$. The color of each (p, k) pixel indicates the average over 100 trials (the lighter the better). If k is too small, our estimate could be very inaccurate. If k is too large, it is less likely to find $k \times k$ uncorrupted matrices to obtain an estimate. This figure shows the feasible range of k (white region, where R2PCA can recover $\mathbf{L}$ within the noise level), which depends on the percentage of corrupted entries p.

mance of R2PCA as a function of noise and coherence, with fixed $p = 5\%$. We repeated this experiment 100 times for each pair (σ, μ). The results, summarized in Figure 4, show that R2PCA can consistently estimate $\mathbf{L}$ within the noise level, as long as $\mathbf{S}$ is sufficiently sparse, regardless of coherence (as opposed to other algorithms).

Astronomy and Correlated Errors. In a video, the background can be modeled as approximately low-rank, and the foreground objects (like cars or people) can be modeled as sparse errors (as they typically take only a fraction of each frame). So the sparse plus low-rank model is a natural fit for this problem. Here $\mathbf{M}$ is the matrix containing the vectorized images in the video, and the goal is to decompose it into the sparse foreground $\mathbf{S}$ and the low-rank background $\mathbf{L}$. We now

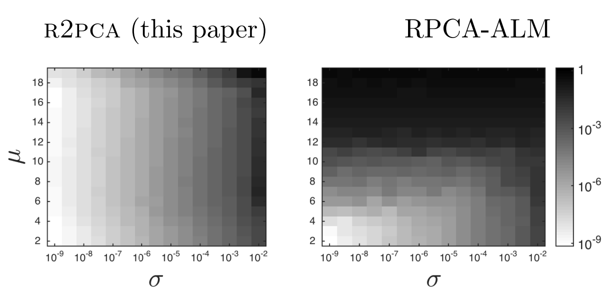

Figure 4: Transition diagram of the estimation error of $\mathbf{L}$ as a function of the noise level σ and the coherence parameter μ, with $p = 5\%$ grossly corrupted entries. The color of each (σ, μ) pixel indicates the average error over 100 trials (the lighter the better). This shows that R2PCA can consistently estimate $\mathbf{L}$ within the noise level, as long as $\mathbf{S}$ is sufficiently sparse, regardless of coherence. Other algorithms can also estimate $\mathbf{L}$ within the noise level, but only for a restricted range of matrices with bounded coherence.

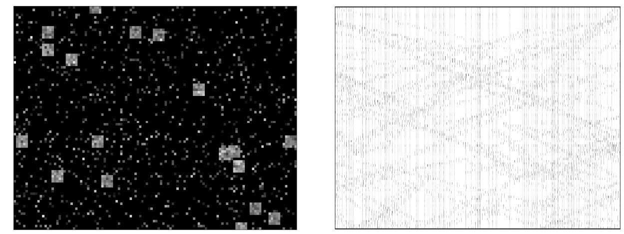

Figure 5: **Left:** One frame of a simulation of an astronomical video, composed of a background formed with ν twinkling stars and ρ moving objects. Each object (block) moves in a random direction at a random speed over the video. **Right:** Each frame is vectorized to form the matrix $\mathbf{M}$, which is shown negated and transposed for display purposes (i.e., we see $1 - \mathbf{M}^\mathsf{T}$). The vertical lines correspond to the pixels of the stars. All other points correspond to the moving objects. These points are highly correlated, as is the location of an object in consecutive frames.

present an experiment where highly coherent matrices and highly correlated sparse errors arise in a very natural way: background segmentation of astronomy videos.

In this experiment we simulated astronomy videos with a background with ν twinkling stars and ρ moving objects (see Figure 5). To this end we first generated a $d \times r$ matrix $\mathbf{U}$, and an $r \times N$ matrix $\mathbf{\Theta}$, with $d = 90 \cdot 120 = 10800$, $r = 5$ and $N = 100$. We selected ν rows in $\mathbf{U}$ uniformly at random, and populated them with the absolute values of i.i.d. $\mathcal{N}(0, 100)$ random variables. These entries represent the twinkling stars. All other entries in $\mathbf{U}$ were populated with the absolute values of i.d.d. $\mathcal{N}(0, 1)$ random variables. Similarly, we populated $\mathbf{\Theta}$ with the absolute value of i.d.d. $\mathcal{N}(0, 1)$ random variables. Next we constructed $\mathbf{L} = \mathbf{U\Theta}$, and bounded its entries by 1 (i.e., we divided $\mathbf{L}$ by its maximum value). Each column of $\mathbf{L}$ represents the vectorized background of a 90×120 frame of a video.

For each of the ρ moving objects, we selected uniformly at random: one starting point on the edge of a 90×120 frame, one starting time between $\{1, 2, \ldots, 100\}$, one direction, and one speed ranging from 1 to 5 pixels per frame. With this information, we created ρ objects moving across a dark background over 100 frames. Each moving object consisted of an $r \times r$ block with $\mathcal{N}(0, 1)$ entries. We vectorized the frames to obtain a 10800×100 matrix, whose entries we normalized between 0 and 1 to obtain $\mathbf{S}$. Finally, we replaced the zero entries in $\mathbf{S}$ with the corresponding entries in $\mathbf{L}$ to obtain $\mathbf{M}$. This way, all entries in $\mathbf{M}$ are between 0 and 1, such that the brightest star shines at a maximum intensity of 1, and so does the brightest pixel of all moving objects.

We repeated this experiment 100 times for each pair

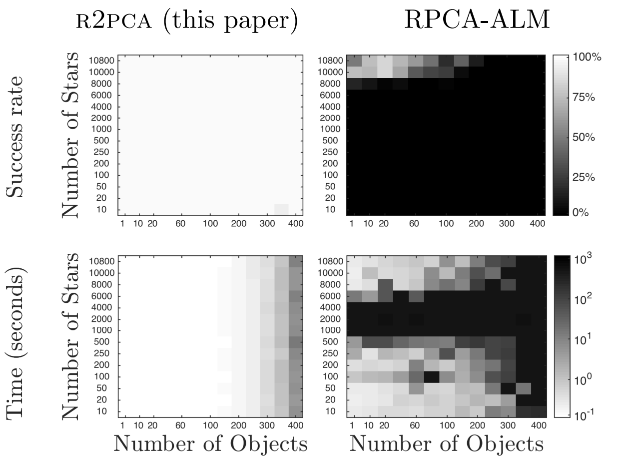

Figure 6: Transition diagrams of the success rate (**top row**) and time (**bottom row**) for exact recovery of $\mathbf{L}$ as a function of the number of moving objects ρ and the number of twinkling stars ν for the experiment described in Figure 5. The larger ρ, the larger proportion of corrupted entries p. The larger ν, the lower coherence μ. The color of each (ρ, ν) pixel indicates the average over 100 trials (the lighter the better). The results are consistent with the experiments in Figure 2.

(ν, ρ). The results, summarized in Figure 6, show that R2PCA has almost perfect performance handling highly coherent matrices and highly correlated errors (as opposed to other algorithms).

Real Data: Microscopy and Surveillance. Finally, we evaluate the background segmentation performance of R2PCA on real data. To this end we used several microscopy videos from the Internet [25], and videos from two widely used datasets: the Wallflower dataset [23] and the I2R dataset [24]. Figure 7 shows several examples, with more in Appendix C.

We point out that many cases of the Wallflower and the I2R datasets have low coherence. In these cases, the performance of R2PCA and RPCA-ALM is very similar. Consistent with our theory, the advantage of R2PCA becomes more evident in highly coherent cases, like our microscopy and astronomy experiments.

Remark 2. *Notice that in all of our background experiments,* R2PCA *can handle a much larger fraction of gross errors than the allowed by Theorem 1. This is because Theorem 1 holds even under the worst-case scenario where the errors are purposely located to complicate success. In many applications, as in background segmentation, errors are often grouped, which tends to leave more uncorrupted blocks. This facilitates* R2PCA*'s success.*

7 Conclusions

In this paper we present R2PCA, a novel algorithm for robust PCA. We show that under reasonable assumptions, R2PCA will succeed with probability 1 in linear time, in lieu of assumptions on coherence or the distribution of the sparse errors. The algorithm is parallelizable and can be used in large scale settings where the dataset is too large to even store in memory. Our experiments show that R2PCA consistently outperforms state-of-the-art methods both in terms of speed and accuracy in a broad range of settings, particularly on high coherence cases.

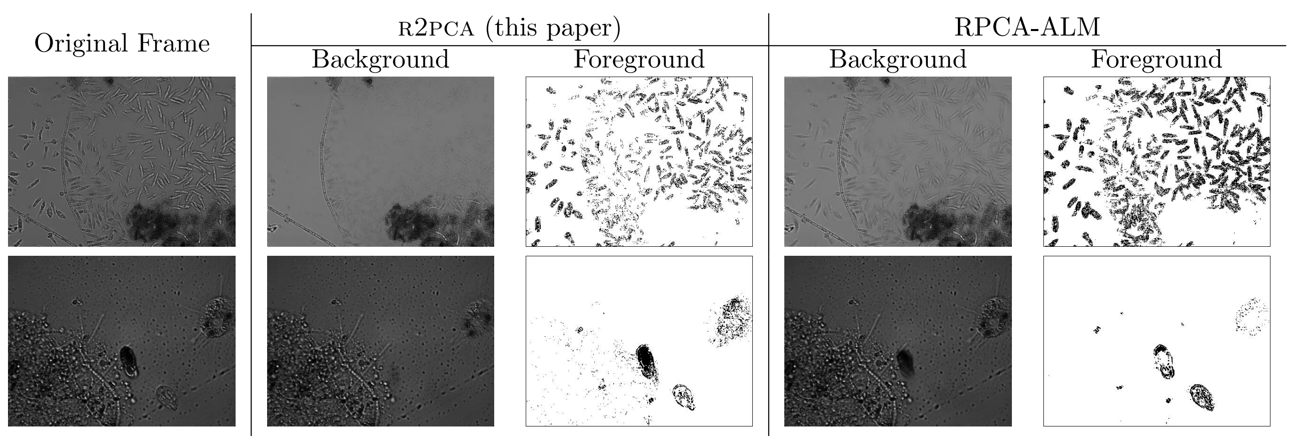

Figure 7: Sparse (foreground) plus low-rank (background) decomposition of several microscopy videos [25] using R2PCA and RPCA-ALM [18, 19]. Notice that the background obtained by RPCA-ALM contains foreground objects, while the background obtained by R2PCA is much cleaner. This is because it these videos the background is mostly dark with a few bright regions (which implies a highly coherent subspace) and the location of the errors is highly correlated (the location of an object in consecutive frames is very similar). In contrast to other optimization methods [5–12, 18, 19], we make no assumptions about coherence or the distribution of the sparse errors, and so this does not affect our results.

References

[1] R. Maronna, *Robust M-estimators of multivariate location and scatter*, The Annals of Statistics, 1976.

[2] M. Fischler and R. Bolles, *Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography*, Communications of the ACM, 1981.

[3] F. De La Torre and M. Black, *A framework for robust subspace learning*, International Journal of Computer Vision, 2003.

[4] Q. Ke and T. Kanade, *Robust L_1 norm factorization in the presence of outliers and missing data by alternative convex programming*, IEEE Conference on Computer Vision and Pattern Recognition, 2005.

[5] E. Candès and J. Romberg, *Sparsity and incoherence in compressive sampling*, Inverse Problems, 2007.

[6] J. Wright, A. Ganesh, S. Rao, Y. Peng and Y. Ma, *Robust principal component analysis: Exact recovery of corrupted low-rank matrices via convex optimization*, Advances in Neural Information Processing Systems, 2009.

[7] J. Cai, E. Candès and Z. Shen, *A singular value thresholding algorithm for matrix completion*, SIAM Journal on Optimization, 2010.

[8] H. Xu, C. Caramanis and S. Sanghavi, *Robust PCA via outlier pursuit*, Advances in Neural Information Processing Systems, 2010.

[9] E. Candès, X. Li , Y. Ma and J. Wright, *Robust principal component analysis?*, Journal of the ACM, 2011.

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

[11] L. Mackey, A. Talwalkar and M. Jordan, *Divide-and-conquer matrix factorization*, Advances in Neural Information Processing Systems, 2011.

[12] M. Rahmani and G. Atia, *A subspace learning approach for high dimensional matrix decomposition with efficient column/row sampling*, International Conference on Machine Learning, 2016.

[13] O. Chum, J. Matas and J. Kittler, *Locally optimized* RANSAC, Pattern Recognition, 2003.

[14] O. Chum and J. Matas, *Optimal randomized* RANSAC, IEEE Transactions on Pattern Analysis and Machine Intelligence, 2008.

[15] A. Hast, J. Nysjö and A. Marchetti, *Optimal* RANSAC *- Towards a repeatable algorithm for finding the optimal set*, Journal of WSCG, 2013.

[16] T. Bouwmans and E. Zahzah, *Robust PCA via principal component pursuit: a review for a comparative evaluation in video surveillance*, Computer Vision and Image Understanding, 2014.

[17] D. Pimentel-Alarcón, N. Boston and R. Nowak, *Deterministic conditions for subspace identifiability from incomplete sampling*, IEEE International Symposium on Information Theory, 2015.

[18] Z. Lin, M. Chen, L. Wu, and Y. Ma, *The augmented Lagrange multiplier method for exact recovery of corrupted low-rank matrices*, University of Illinois at Urbana-Champaign Technical Report, 2009.

[19] Z. Lin, R. Liu and Z. Su, *Linearized alternating direction method with adaptive penalty for low rank representation*, Advances in Neural Information Processing Systems, 2011.

[20] Y. Ma, *Low-rank matrix recovery and completion via convex optimization*, avaiable at `http://perception.csl.illinois.edu/matrix-rank/home.html`.

[21] X. Yuan and J. Yang, *Sparse and low-rank matrix decomposition via alternating direction methods*, available at `http://www.optimization-online.org/DB_HTML/2009/11/2447.html`, 2009.

[22] Z. Lin, A. Ganesh, J. Wright, L. Wu, M. Chen and Y. Ma, *Fast convex optimization algorithms for exact recovery of a corrupted low-rank matrix*, Computational Advances in Multi-Sensor Adaptive Processing, 2009.

[23] K. Toyama, J. Krumm, B. Brumitt and B. Meyers, *Wallflower: principles and practice of background maintenance*, IEEE International Conference on Computer Vision, 1999. Dataset available at: `http://research.microsoft.com/en-us/um/people/jckrumm/wallflower/testimages.htm`

[24] L. Li, W. Huang, I. Gu and Q. Tian, *Statistical modeling of complex backgrounds for foreground object detection*, IEEE Transactions on Image Processing, 2004. Dataset available at: `http://perception.i2r.a-star.edu.sg/bk_model/bk_index.html`

[25] Microscopy videos downloaded from the Internet (YouTube), available at: `https://www.youtube.com/watch?v=iTswesT8zeo`, `https://www.youtube.com/watch?v=-sxrIvbqBGg`, `https://www.youtube.com/watch?v=sX6PTD4t-Xs`, `https://www.youtube.com/watch?v=Oocе_x-SoJA`, `https://www.youtube.com/watch?v=R99T08FjTkc`, `https://www.youtube.com/watch?v=6wNd-sHU7r8`, `https://www.youtube.com/watch?v=gbrZhYzk480`, `https://www.youtube.com/watch?v=1xlNgeBc3ek`.

Random Consensus Robust PCA

Supplementary Material

Daniel Pimentel-Alarcón & Robert Nowak
University *of* Wisconsin-Madison

A Proof of Theorem 1

In this section we give the proof of Theorem 1. Recall that $\mathbf{\Omega}$ is a $d \times (d-r)$ matrix, and that $\boldsymbol{\omega}_i \subset \{1, \dots, d\}$ indexes the $r+1$ nonzero entries in the i^{th} column of $\mathbf{\Omega}$. Each $\boldsymbol{\omega}_i$ indicates the coordinates of a projection of U that we aim to identify. Since R2PCA selects a matrix $\mathbf{\Omega}$ satisfying (i), Lemma 1 implies that if we find the projections of U onto the coordinates indicated in $\mathbf{\Omega}$, then we can reconstruct U from these projections. Hence we need to show that under the assumptions of Theorem 1, R2PCA can find the projections of U onto the $\boldsymbol{\omega}_i$'s in $\mathbf{\Omega}$. To this end, we will show that R2PCA can potentially find the projection onto *any* $\boldsymbol{\omega} \subset \{1, 2, \dots, d\}$ with exactly $r+1$ elements.

So let $\boldsymbol{\omega}$ be given. As discussed in Section 2, finding the projection $U_{\boldsymbol{\omega}}$ equates to finding $r+1$ uncorrupted columns in $\mathbf{M}_{\boldsymbol{\omega}}$. So R2PCA can potentially find $U_{\boldsymbol{\omega}}$ as long as there are $r+1$ uncorrupted columns in $\mathbf{M}_{\boldsymbol{\omega}}$. Since $\mathbf{M}_{\boldsymbol{\omega}}$ only contains $r+1$ rows, **A4** implies that there are at most $\frac{(n-r)(r+1)}{2(r+1)^{\alpha}}$ corrupted entries in $\mathbf{M}_{\boldsymbol{\omega}}$. In the worst-case scenario, each of these corrupted entries is located in a different column. It follows that $\mathbf{M}_{\boldsymbol{\omega}}$ has at most $\frac{n-r}{2(r+1)^{\alpha-1}}$ corrupted columns. Then

$$\mathsf{P}\big(i^{\text{th}} \text{ column in } \mathbf{M}'_{\boldsymbol{\omega}} \text{ is uncorrupted}\big) \geq 1 - \frac{1}{2(r+1)^{\alpha-1}},$$

which corresponds to the case where the first r columns in $\mathbf{M}'_{\boldsymbol{\omega}}$ are uncorrupted, whence the ratio of uncorrupted columns $((n-r) - \frac{n-r}{2(r+1)^{\alpha-1}})$ versus total remaining columns $(n-r)$ is smallest. It follows that

$$\begin{aligned}
&\mathsf{P}\big(\text{all columns in } \mathbf{M}'_{\boldsymbol{\omega}} \text{ are uncorrupted}\big) \\
&\quad\geq \left(1 - \frac{1}{2(r+1)^{\alpha-1}}\right)^{r+1} \\
&\quad= \left(1 - \frac{1/2}{(r+1)^{\alpha-1}}\right)^{(r+1)^{1+(\alpha-1)-(\alpha-1)}} \\
&\quad= \left(1 - \frac{1/2}{(r+1)^{\alpha-1}}\right)^{(r+1)^{(\alpha-1)}(r+1)^{2-\alpha}} \\
&\quad= \left(\left(1 - \frac{1/2}{(r+1)^{\alpha-1}}\right)^{(r+1)^{(\alpha-1)}}\right)^{(r+1)^{2-\alpha}} \\
&\quad\geq (1/2)^{(r+1)^{2-\alpha}}.
\end{aligned} \tag{1}$$

This implies that on expectation, R2PCA will require at most $2^{(r+1)^{2-\alpha}}$ iterations to find a set of $r+1$ uncorrupted columns in $\mathbf{M}_{\boldsymbol{\omega}}$. This is true for every $\boldsymbol{\omega}$. Since R2PCA only searches over the $\boldsymbol{\omega}_i$'s in $\mathbf{\Omega}$, and since $\mathbf{\Omega}$ has exactly $d-r$ columns, it follows that on expectation, R2PCA will require at most $(d-r)2^{(r+1)^{2-\alpha}}$ iterations to find the projections of U onto the canonical coordinates indicated by $\mathbf{\Omega}$. Since $\mathbf{\Omega}$ satisfies condition (i), we know by Lemma 1 that U is given by $\ker \mathbf{A}^{\mathsf{T}}$.

Now that U is known, let us show that R2PCA can recover $\mathbf{L}$. Let $\mathbf{U}$ be an arbitrary basis of U. We will show that R2PCA can determine the matrix $\mathbf{\Theta}$ containing the coefficients of $\mathbf{L}$ in this basis, such that in the end, $\mathbf{L}$ will be given by $\mathbf{U\Theta}$. To this end, let $\mathbf{m}$ be a column in $\mathbf{M}$. Observe that R2PCA can potentially find the coefficients of the corresponding column of $\mathbf{L}$ as long as there is a set $\boldsymbol{\omega} \subset \{1, 2, \dots, d\}$ with $r+1$ elements such that $\mathbf{m}_{\boldsymbol{\omega}} \in U_{\boldsymbol{\omega}}$. **A1-A3** imply that with probability 1, this will be the case if and only there are at least $r+1$ uncorrupted entries in $\mathbf{m}$. By **A4**, there are at most $\frac{d-r}{2(r+1)^{\alpha-1}}$ corrupted entries in $\mathbf{m}$. It follows that

$$\mathsf{P}\big(i^{\text{th}} \text{ entry in } \mathbf{m}_{\boldsymbol{\omega}} \text{ is uncorrupted}\big) \geq 1 - \frac{1}{2(r+1)^{\alpha-1}},$$

which corresponds to the case where the first r entries in $\mathbf{m}_{\boldsymbol{\omega}}$ are uncorrupted, whence the ratio of uncorrupted entries ($(d-r) - \frac{d-r}{2(r+1)^{\alpha-1}}$) versus total remaining entries $(d-r)$ is smallest. It follows that

$$\mathsf{P}\big(\text{all entries in } \mathbf{m}_{\boldsymbol{\omega}} \text{ are uncorrupted}\big) \geq \left(1 - \frac{1}{2(r+1)^{\alpha-1}}\right)^{r+1} \geq (1/2)^{(r+1)^{2-\alpha}},$$

where the last inequality follows by the same arithmetic manipulations as in (1). This implies that on expectation, R2PCA will require at most $2^{(r+1)^{2-\alpha}}$ iterations to find a set of $r+1$ uncorrupted entries in $\mathbf{m}$. This is true for every $\mathbf{m}$. Since $\mathbf{M}$ has n columns, it follows that on expectation, R2PCA will require at most $n2^{(r+1)^{2-\alpha}}$ iterations to recover $\mathbf{L}$. Once $\mathbf{L}$ is known, $\mathbf{S}$ can be trivially recovered as $\mathbf{S} = \mathbf{M} - \mathbf{L}$. This shows that on expectation, R2PCA will require at most $(d+n-r)2^{(r+1)^{2-\alpha}}$ iterations to recover U, $\mathbf{L}$ and $\mathbf{S}$ from $\mathbf{M}$. $\square$

B Noisy Variant

In Section 5 we described a noisy variant of R2PCA. This variant iteratively selects matrices $\mathbf{M}'_{\boldsymbol{\kappa}} \in \mathbb{R}^{k\times k}$ formed with k rows of k columns of $\mathbf{M}$, and verifies the $(r+1)^{\text{th}}$ singular value of $\mathbf{M}'_{\boldsymbol{\kappa}}$. If this singular value is within the noise level, Algorithm 1 will consider $\mathbf{M}'_{\boldsymbol{\kappa}}$ uncorrupted, and use it to estimate projections of U. Otherwise Algorithm 1 will discard $\mathbf{M}'_{\boldsymbol{\kappa}}$ and keep looking. This process is repeated until there are enough projections to recover U. Once U is estimated, Algorithm 1 proceeds to estimate the coefficients of $\mathbf{L}$ using k entries per column of $\mathbf{M}$. If these entries agree with the estimated subspace U, they will be considered uncorrupted, and used to estimate the coefficient of the corresponding column of $\mathbf{L}$. Otherwise, Algorithm 1 will discard these entries, and select an other k. This process is repeated until we recover all the coefficients of $\mathbf{L}$. This noisy variant of R2PCA is summarized in Algorithm 1.

Algorithm 1: Random Robust PCA (R2PCA, noisy variant)

Input: Data $\mathbf{M} \in \mathbb{R}^{d \times n}$, rank r,
  matrix $\mathbf{\Omega} \in \{0,1\}^{d \times (d-r)}$ satisfying (i),
  parameter $k \in \mathbb{N}$.
PART 1: Estimate $\mathbf{U}$
  for $i = 1, 2, \dots, d-r$ **do**
    $\boldsymbol{\omega}_i$ = indices of the $r+1$ nonzero rows of the i^{th} column in $\mathbf{\Omega}$.
    $\boldsymbol{\kappa}_i$ = subset of $\{1, \dots, d\}$ containing $\boldsymbol{\omega}_i$ and $k-r+1$ other rows selected randomly.
    repeat
      $\mathbf{M}'_{\boldsymbol{\kappa}_i} \in \mathbb{R}^{k \times k}$ = k columns of $\mathbf{M}_{\boldsymbol{\kappa}_i}$, selected randomly.
    until $(r+1)^{\text{th}}$ singular value of $\mathbf{M}'_{\boldsymbol{\kappa}_i}$ is within the noise level.
    $\mathbf{V}_{\boldsymbol{\kappa}_i} \in \mathbb{R}^{k \times r}$ = r leading singular vectors of $\mathbf{M}'_{\boldsymbol{\kappa}_i}$.
    $\boldsymbol{\upsilon}_i$ = subset of $\boldsymbol{\kappa}_i$ with exactly r elements, selected randomly.
    for each $j \in \boldsymbol{\kappa}_i \backslash \boldsymbol{\upsilon}_i$ **do**
      $\boldsymbol{\omega}_{ij} := \boldsymbol{\upsilon}_i \cup j$.
      $\mathbf{a}_{\boldsymbol{\omega}_{ij}} \in \mathbb{R}^{r+1}$ = nonzero vector in $\ker \mathbf{V}^{\mathsf{T}}_{\boldsymbol{\omega}_{ij}}$.
      $\mathbf{a}_{ij} \in \mathbb{R}^d$ = vector with $\mathbf{a}_{\boldsymbol{\omega}_{ij}}$ in the locations of $\boldsymbol{\omega}_{ij}$, and zeros elsewhere.
      Insert $\mathbf{a}_{ij}$ into $\mathbf{A}$.
  $\hat{\mathbf{U}} \in \mathbb{R}^{d \times r}$ = basis of $\ker \mathbf{A}^{\mathsf{T}}$.
PART 2: Estimate $\mathbf{\Theta}$
  for each column $\mathbf{m}$ in $\mathbf{M}$ **do**
    repeat
      $\boldsymbol{\kappa}$ = subset of $\{1, \dots, d\}$ with k elements, selected randomly.
    until $\mathbf{m}_{\boldsymbol{\kappa}}$ is close to span$\{\hat{\mathbf{U}}_{\boldsymbol{\kappa}}\}$ (within the noise level).
    $\hat{\boldsymbol{\theta}} = (\hat{\mathbf{U}}^{\mathsf{T}}_{\boldsymbol{\kappa}} \hat{\mathbf{U}}_{\boldsymbol{\kappa}})^{-1} \hat{\mathbf{U}}^{\mathsf{T}}_{\boldsymbol{\kappa}} \mathbf{m}_{\boldsymbol{\kappa}}$.
    Insert $\hat{\boldsymbol{\theta}}$ into $\hat{\mathbf{\Theta}}$.
Output: $\hat{\mathbf{U}}$, $\hat{\mathbf{L}} = \hat{\mathbf{U}}\hat{\mathbf{\Theta}}$, $\hat{\mathbf{S}} = \mathbf{M} - \hat{\mathbf{L}}$.

C Additional Results

Microscopy Segmentation In Section 6 we gave three examples of the background segmentation that we obtained for three microscopy videos from the Internet. Figure 1 shows more results.

Wallflower and I2R Datasets. To complement the real data experiments in Section 6, we also ran R2PCA and RPCA-ALM on the Wallflower [23] and the I2R [24] datasets. The results are summarized in Figure 2. We point out that many cases of the Wallflower and the I2R datasets have low coherence. In these cases, the performance of R2PCA and RPCA-ALM is very similar. Consistent with our theory, the advantage of R2PCA becomes more evident in highly coherent cases, like our microscopy and astronomy experiments.

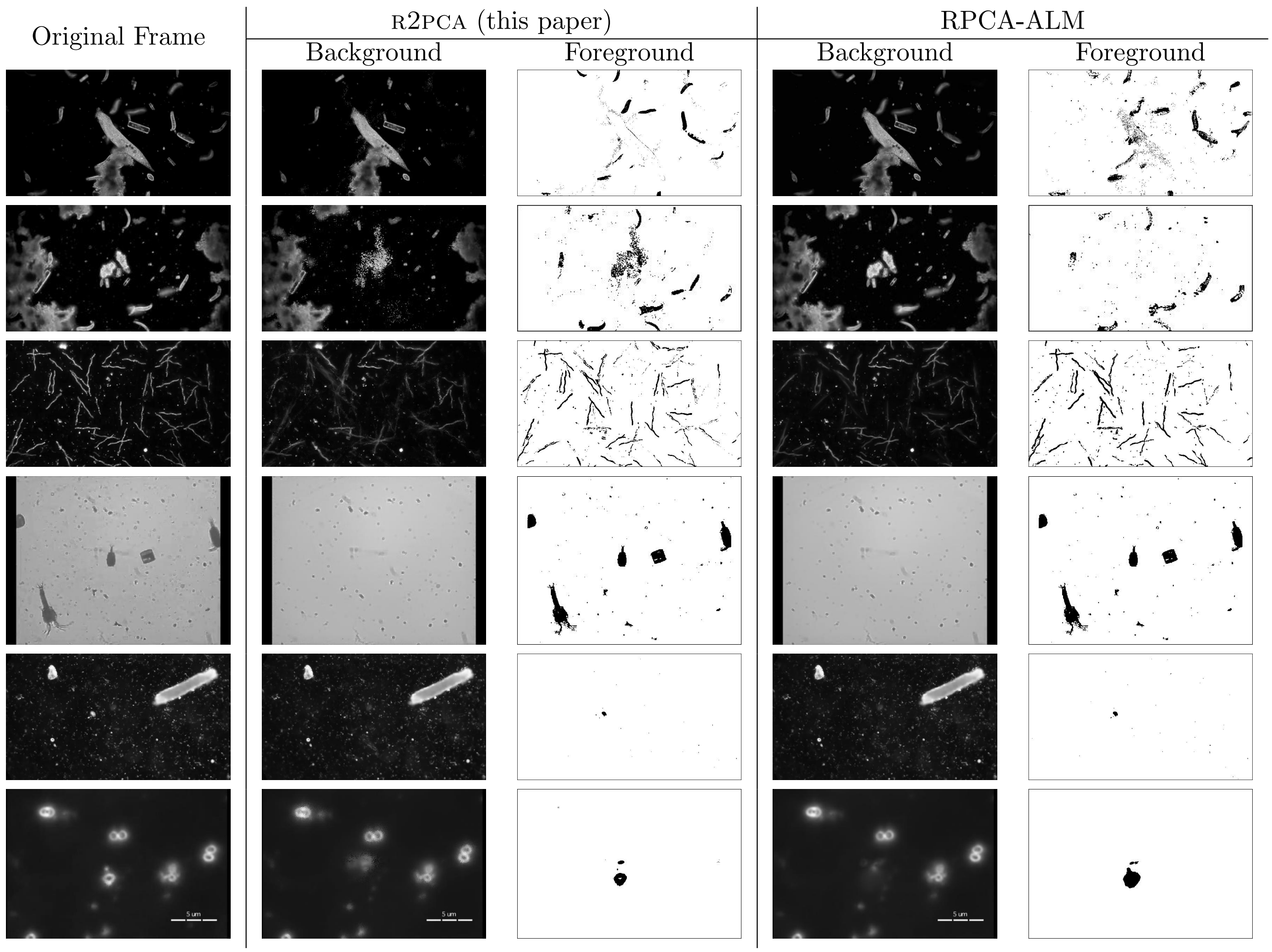

Figure 1: Sparse (foreground) plus low-rank (background) decomposition of some video frames from several microscopy videos from the Internet [25] using R2PCA and RPCA-ALM [18,19]. Notice that the background obtained by RPCA-ALM contains foreground objects, while the background obtained by R2PCA is much cleaner. This is because it these videos the background is mostly dark with a few bright regions (which implies a highly coherent subspace) and the location of the errors is highly correlated (the location of an object in consecutive frames is very similar). In contrast to other optimization methods [5-12,18,19], we make no assumptions about coherence or the distribution of the sparse errors, and so this does not affect our results.

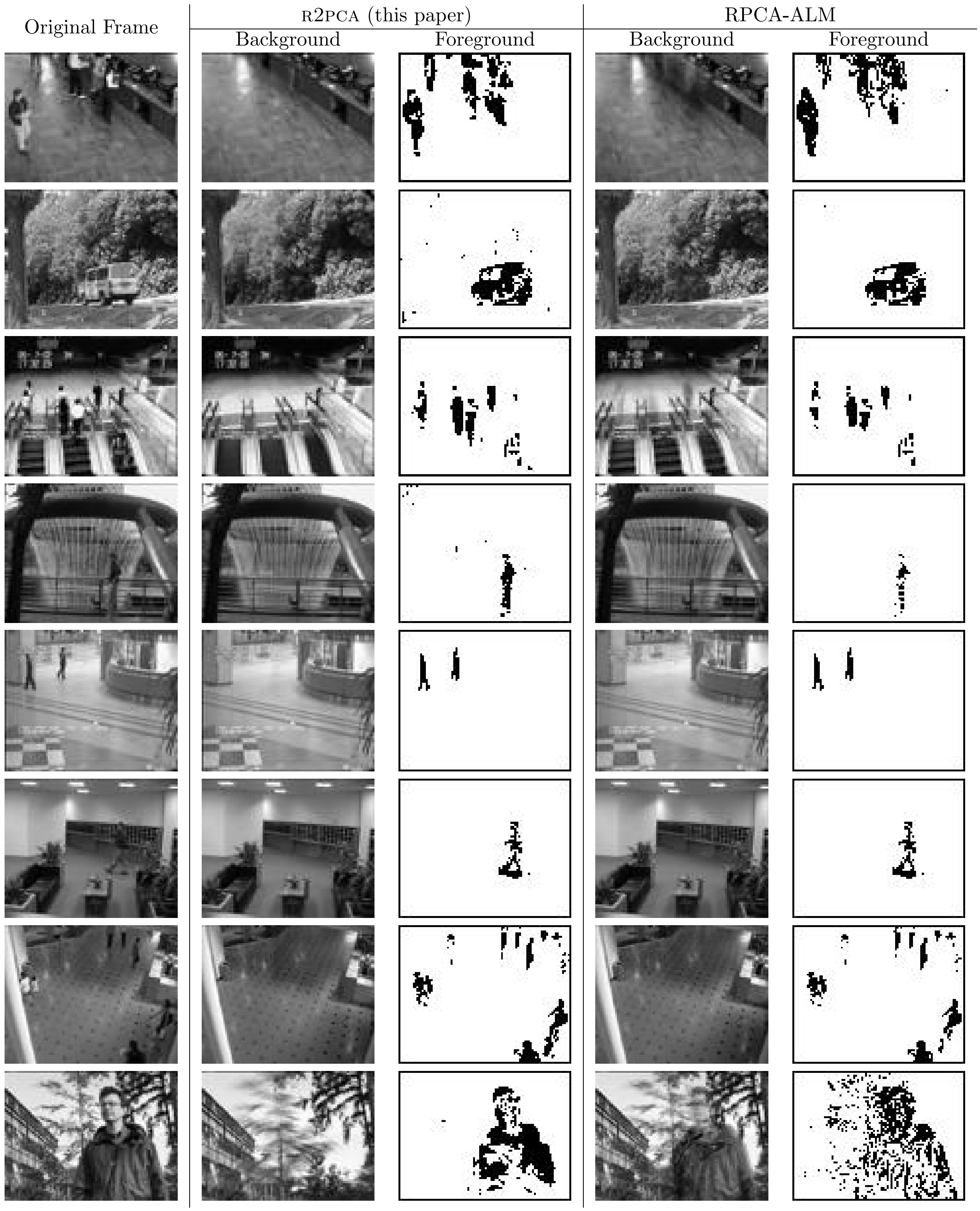

Figure 2: Sparse (foreground) plus low-rank (background) decomposition of some video frames from the Wallflower [23] and I2R [24] datasets using r2pca and RPCA-ALM [18,19].