SIAM J. IMAGING SCIENCES
Vol. 14, No. 4, pp. 1472–1503

Robust CUR Decomposition: Theory and Imaging Applications*

HanQin Cai†, Keaton Hamm‡, Longxiu Huang†, and Deanna Needell†

Abstract. This paper considers the use of robust principal component analysis (RPCA) in a CUR decomposition framework and applications thereof. Our main algorithms produce a robust version of column-row factorizations of matrices $D = L + S$, where L is low-rank and S contains sparse outliers. These methods yield interpretable factorizations at low computational cost and provide new CUR decompositions that are robust to sparse outliers, in contrast to previous methods. We consider two key imaging applications of RPCA: video foreground-background separation and face modeling. This paper examines the qualitative behavior of our robust CUR decompositions on the benchmark videos and face datasets and finds that our method works as well as standard RPCA while being significantly faster. Additionally, we consider hybrid randomized and deterministic sampling methods which produce a compact CUR decomposition of a given matrix and apply this to video sequences to produce canonical frames thereof.

Key words. CUR decomposition, RPCA, robust CUR, low-rank matrix approximation, interpolative decompositions, robust algorithms

AMS subject classifications. 15A23, 65F30, 68P20, 68W20, 68W25, 68Q25

DOI. 10.1137/20M1388322

1. Introduction.

The scale of data being collected is increasingly massive, which requires better and more computationally efficient tools to handle associated data matrices. One method of handling large data matrices is to extract a meaningful column submatrix and to either represent the full data matrix via a factorization with respect to the columns or to carry out any desired analysis on the column submatrix and then reconstruct the full data matrix from the columns. This technique is motivated by some foundational results that tell one when and how well column submatrices can represent a data matrix. In particular, for a low-rank matrix $\boldsymbol{L} \in \mathbb{R}^{m\times n}$ with $\operatorname{rank}(\boldsymbol{L}) = r$, one can write $\boldsymbol{L} = \boldsymbol{C}\boldsymbol{X}$, where $\boldsymbol{C}$ is a (much smaller) column submatrix of $\boldsymbol{L}$ as long as $\operatorname{rank}(\boldsymbol{C}) = \operatorname{rank}(\boldsymbol{L})$. Hence, one obtains a representation of $\boldsymbol{L}$ in terms of a few representative data vectors.

*Received by the editors December 28, 2020; accepted for publication (in revised form) July 20, 2021; published electronically October 18, 2021.

https://doi.org/10.1137/20M1388322

Funding: The work of the second author was partially supported by the Army Research Office under grant number W911NF-20-1-0076. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Office or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein. The work of the third and fourth authors was supported by NSF BIGDATA DMS-1740325 and NSF DMS-2011140.

†Department of Mathematics, University of California, Los Angeles, Los Angeles, CA 90095 USA (hqcai@math.ucla.edu, huangl3@math.ucla.edu, deanna@math.ucla.edu).

‡Department of Mathematics, University of Texas at Arlington, Arlington, TX 76019 USA (keaton.hamm@uta.edu).

In general, such factorizations are called *interpolative decompositions* [50], while for particular choices of $\boldsymbol{X}$ including $\boldsymbol{X} = \boldsymbol{C}^\dagger \boldsymbol{L}$ or $\boldsymbol{X} = \boldsymbol{U}^\dagger \boldsymbol{R}$, where $\boldsymbol{R}$ is a row submatrix of $\boldsymbol{L}$ and $\boldsymbol{U}$ is the overlap of the column and row indices of $\boldsymbol{C}$ and $\boldsymbol{R}$, they are termed *CUR decompositions* [7, 20, 22, 33, 37, 44]. By some, the latter examples are called *cross-approximations* [40] or *(pseudo-)skeleton approximations* [16, 26, 27, 28]. The problem of finding the best set of k columns with which to approximate $\boldsymbol{L}$ is the column subset selection problem [6].

Since their advent, column factorizations such as these have been shown to have great flexibility and approximation power as a general low-rank approximation tool. Specifically, they provide advantages over traditional factorization methods, For example, (1) column factorizations can yield compact representations of large data matrices; (2) for incoherent matrices, the complexity of generating the uniform sampling pattern is $\mathcal{O}(1)$[1] while yielding a factorization which well approximates the SVD (which requires $\mathcal{O}(mnr)$ time for a rank r matrix); and (3) the representation of a matrix in a basis consisting of actual data vectors provides a more interpretable representation of the data than abstract factorizations such as QR or SVD.

In many image and video processing applications, the observed data can be interpreted as follows:

1. Low-rank information of interest with some sparse outliers. For example, in the task of face modeling [51], the face data of the same person in different photos form a low-rank matrix that we want to recover, while the facial occlusions (accessories, shadows, and facial expressions) are considered sparse outliers.
2. A combination of two pieces of information of interest that have low rank and sparse properties, respectively. For example, in the task of video background subtraction [34], the static background has the low-rank property, and the foreground (i.e., the moving objects) is sparse.

Robust principal component analysis (RPCA) is an apt tool for solving the aforementioned problems and various other tasks in this arena [9, 14, 52].

In the RPCA framework, we assume we observe $\boldsymbol{D} = \boldsymbol{L} + \boldsymbol{S} \in \mathbb{R}^{m \times n}$, where $\boldsymbol{L}$ is a low-rank matrix and $\boldsymbol{S}$ is a sparse matrix, and the goal is to recover $\boldsymbol{L}$. For large-scale data problems, $\boldsymbol{D}$ may not be able to be stored in memory. The purpose of this paper is to elucidate how one may use column factorizations to assist in the RPCA task of finding the low-rank piece. We elaborate on our main contributions in section 1.2 below, but first we present the main ideas utilized throughout the paper.

1.1. Main ideas. There are several points to consider when trying to marry the techniques of CUR decompositions and robust low-rank matrix recovery from sparse outliers. The main goal of this paper is to do so in such a way as to guarantee speed, interpretability, and robustness. The first two goals are commonly achieved by CUR decompositions (as opposed to SVD based methods), while the third is achieved by RPCA but not typically by CUR. We elaborate more on how the method proposed here achieves these aims in comparison with existing techniques in the following subsection.

Taking these factors into consideration, we propose the following broadly applicable method: subsample columns and rows of $\boldsymbol{D}$, apply an existing RPCA algorithm to these

[1] Here and throughout, we take $\mathcal{O}(1)$ to mean asymptotically with respect to m and n.

matrices, and use the denoised column and row matrices to form a CUR decomposition of the underlying low-rank part, $\boldsymbol{L}$. This procedure has several advantages: (1) it is faster than standard RPCA as long as few columns and rows may be selected, which is guaranteed by some theoretical results on submatrices of incoherent matrices; (2) it can work on matrices that do not fit in memory because it first subsamples the data matrix; (3) it returns a column-row factorization that is known to be more interpretable than the SVD as it corresponds to representing data via other actual data points; (4) there are good theoretical guarantees for its performance; and (5) it is a flexible framework that allows the user to specify the RPCA algorithm utilized. The last point allows one to determine the tradeoff between robustness and computational cost existing in the various RPCA methods in use.

1.2. Main contributions. We consider two variations of algorithms for producing robust CUR decompositions. In particular, given $\boldsymbol{D} = \boldsymbol{L} + \boldsymbol{S}$, our first algorithm (Algorithm 3.1) uniformly randomly selects a column submatrix $\boldsymbol{C} = \boldsymbol{D}(:, J)$ and row submatrix $\boldsymbol{R} = \boldsymbol{D}(I, :)$ of $\boldsymbol{D}$, then performs RPCA on both of these matrices to produce $\widehat{\boldsymbol{C}}$ and $\widehat{\boldsymbol{R}}$ which approximate $\boldsymbol{L}(:, J)$ and $\boldsymbol{L}(I, :)$, respectively. Then we obtain an approximate CUR decomposition of the underlying low-rank matrix $\boldsymbol{L}$; namely, $\boldsymbol{L} \approx \widehat{\boldsymbol{C}}(\widehat{\boldsymbol{C}}(I, :))^{\dagger}\widehat{\boldsymbol{R}} \approx \boldsymbol{L}(:, J)(\boldsymbol{L}(I, J))^{\dagger}\boldsymbol{L}(I, :)$. We call such decompositions robust CUR (RCUR) decompositions.

The theoretical contributions of this procedure are as follows:

1. We prove bounds for all of the relevant properties of RPCA algorithms (incoherence, sparsity, and condition number) of randomly selected column and row submatrices of $\boldsymbol{L}$ in terms of the corresponding properties of $\boldsymbol{L}$.
2. Our algorithms provide a novel robust version of CUR decompositions.
3. Our error analysis shows that RCUR decompositions are capable of well approximating low-rank matrices with sparse, but possibly large magnitude noise, a task for which standard CUR decompositions are not designed.
4. Our RCUR decomposition yields relative error spectral norm guarantees, which have not been shown previously for standard CUR decompositions.
5. The column/row subsampling procedure allows our algorithm to work on large data matrices which cannot be stored in memory and typically has less computational cost than RPCA on the whole data matrix.
6. Our algorithm is flexible, as it allows the user to specify the RPCA algorithm used and thus to specify the tradeoff between robustness and computational cost.

For independent interest, we also consider how properties such as incoherence pass to submatrices under the greedy column selection procedure of [3]. Under greedy column selection, one may have adversarial deterministic sparse noise which prevents the column submatrix from being sparse.

Our second main algorithm is a hybrid random + deterministic method for choosing exactly $\operatorname{rank}(\boldsymbol{L})$ columns and rows of $\boldsymbol{D}$ to produce the most compact CUR decomposition of $\boldsymbol{L}$. We show that with high probability, this algorithm produces a good CUR approximation of $\boldsymbol{L}$ in the spectral norm.

1.3. Prior art. Both CUR decompositions and RPCA algorithms have been well developed in the literature. However, our work is one of the first to combine them. Randomized sampling algorithms for CUR decompositions and the column subset selection problem have

been studied in [16, 20, 22, 32, 37, 46, 47], for example, while deterministic sampling methods have been explored in [3, 5, 35], with hybrid methods explored in [6, 7]. For a general overview, of CUR decompositions and approximations, see [33]. In contrast to the current paper, CUR decompositions typically exhibit error guarantees in the Frobenius norm, while ours hold for the spectral norm. Additionally, typical upper bounds for CUR decompositions of matrices hold in the worst case when the matrix is full rank, and the CUR decomposition approximates the truncated SVD. Thus CUR decompositions are primarily a low-rank approximation tool. This is in contrast to RPCA algorithms, which are primarily concerned with estimating a low-rank matrix which is corrupted by sparse outliers, i.e., RPCA is a low-rank recovery tool.

The algorithms developed herein apply RPCA in a CUR decomposition framework [12, 14, 39]. Our Algorithm 3.1 is faster than merely applying a RPCA algorithm directly to $\boldsymbol{D}$ and is usable for matrices which are too large to store in memory; however, this comes at a small cost in robustness to extreme outliers. We note that the authors in [13] used CUR decompositions as an approximation to the SVD within an iterative RPCA framework; however, while that algorithm achieved empirical success, there is currently no theoretical result for the algorithm convergence or outlier toleration.

2. Preliminaries and layout.

We will always assume that $\boldsymbol{L} = \boldsymbol{W_L}\boldsymbol{\Sigma_L}\boldsymbol{V_L}^T$ is the compact SVD of $\boldsymbol{L} \in \mathbb{R}^{m\times n}$. The symbol $[n]$ denotes the set of integers $\{1, \dots, n\}$ for any $n \in \mathbb{N}$. We use $\kappa(\boldsymbol{L}) := \|\boldsymbol{L}\|_2\|\boldsymbol{L}^\dagger\|_2 = \frac{\sigma_{\max}(\boldsymbol{L})}{\sigma_{\min}(\boldsymbol{L})}$ to denote the spectral condition number of $\boldsymbol{L}$, where $\sigma_{\max}(\boldsymbol{L})$ and $\sigma_{\min}(\boldsymbol{L})$ are the maximum and minimum nonzero singular values of $\boldsymbol{L}$, respectively.

Below are two fundamental assumptions which are typically made in the RPCA framework, which we also utilize.

Definition 2.1. *Let $\boldsymbol{L} \in \mathbb{R}^{m\times n}$ with $\operatorname{rank}(\boldsymbol{L}) = r$, and let $\boldsymbol{L} = \boldsymbol{W_L}\boldsymbol{\Sigma_L}\boldsymbol{V_L}^T$ be its compact SVD. Then $\boldsymbol{L}$ has $\{\mu_1(\boldsymbol{L}), \mu_2(\boldsymbol{L})\}$-incoherence (i.e., $\mu_1(\boldsymbol{L})$-column incoherence and $\mu_2(\boldsymbol{L})$-row incoherence) for some constants $1 \le \mu_1(\boldsymbol{L}) \le \frac{m}{r}$, $1 \le \mu_2(\boldsymbol{L}) \le \frac{n}{r}$ provided*

$$\max_i \left\|\boldsymbol{W_L}^T\boldsymbol{e}_i\right\|_2 \le \sqrt{\frac{\mu_1(\boldsymbol{L})r}{m}} \quad \text{and} \quad \max_i \left\|\boldsymbol{V_L}^T\boldsymbol{e}_i\right\|_2 \le \sqrt{\frac{\mu_2(\boldsymbol{L})r}{n}}. \tag{A1}$$

Definition 2.2. *Let $\boldsymbol{S} \in \mathbb{R}^{m\times n}$ be an α-sparse matrix, and let $\boldsymbol{e}_i$ be the ith canonical unit basis vector of the appropriate dimension. Then $\boldsymbol{S}$ has at most αn nonzero entries in each row and at most αm nonzero entries in each column. In the other words,*

$$\max_i \left\|\boldsymbol{S}^T\boldsymbol{e}_i\right\|_0 \le \alpha n \quad \text{and} \quad \max_j \|\boldsymbol{S}\boldsymbol{e}_j\|_0 \le \alpha m, \tag{A2}$$

where $\|\cdot\|_0$ denotes the ℓ_0-norm.

Remark 2.3. For a successful reconstruction, we expect $\mu_i(\boldsymbol{L}) = \mathcal{O}(1)$, $i = 1, 2$, which is generally true in real-world RPCA applications [9, 11, 12]. For the purpose of theoretical analysis, if $\mu_i(\boldsymbol{L})$ is too large, then the RPCA problem may become too difficult to solve since the toleration of α usually depends on $\operatorname{poly}(\frac{1}{\mu_i(\boldsymbol{L})})$ [12, 14, 39, 53]. Additionally, the rank-sparsity uncertainty principle [15] states that a matrix cannot be simultaneously incoherent and sparse; that is, small $\mu_i(\boldsymbol{L})$ ensures $\boldsymbol{L}$ is not sparse.

2.1. Organization. The rest of the paper is laid out as follows: section 3 contains statements of our main results as well as our main algorithm; section 4 contains the proof of our results concerning how incoherence, sparsity, and condition number change via column selection (Theorem 3.1). Section 5 illustrates our methods via experiments on video background-foreground separation tasks. Finally, the technical details of proofs of the main results are contained in the appendices; specifically, the proof of Theorem 3.5 is contained in Appendix A, the combined error analysis of CUR decompositions and RPCA leading to the proof of Theorem 3.10 is in Appendix B, the proofs for the combined random and deterministic sampling result of Theorem 3.11 are in Appendix C, and the purely deterministic sampling results are contained in Appendices D and E.

3. Main results.

To develop a framework of combining RPCA and CUR decompositions, one must first understand how properties of incoherence and sparsity are inherited by submatrices. This, in turn, requires estimating pseudoinverses of submatrices of singular vectors of $\boldsymbol{L}$, which is generally a difficult task.

Recall the first step in the RCUR decomposition described above: we select column and row submatrices of the observed data $\boldsymbol{D} = \boldsymbol{L} + \boldsymbol{S}$, $\widetilde{\boldsymbol{C}} = \boldsymbol{D}(:, J)$, and $\widetilde{\boldsymbol{R}} = \boldsymbol{D}(I, :)$. We then apply RPCA algorithms to $\widetilde{\boldsymbol{C}}$, and $\widetilde{\boldsymbol{R}}$ to find "good" approximations for the submatrices $\boldsymbol{C} = \boldsymbol{L}(:, J)$ and $\boldsymbol{R} = \boldsymbol{L}(I, :)$ of the underlying low-rank matrix $\boldsymbol{L}$. To apply the RPCA algorithms on $\widetilde{\boldsymbol{C}}$ and $\widetilde{\boldsymbol{R}}$, we must ensure that $\boldsymbol{C}$ and $\boldsymbol{R}$ are incoherent and that $\boldsymbol{S}(I, :)$, $\boldsymbol{S}(: .J)$ are sparse. In the next subsection, we provide quite general bounds for the incoherence of submatrices for various column selection methods.

Second, to carry out an error analysis for RCUR decompositions, we must consider a concrete sampling method for which the submatrices $\boldsymbol{C}$ and $\boldsymbol{R}$ of $\boldsymbol{L}$ have low incoherence and the column and row submatrices of $\boldsymbol{S}$ are sparse, and we must consider the error of the CUR decomposition $\boldsymbol{L} - \widehat{\boldsymbol{C}}(\widehat{\boldsymbol{C}}(I, :))^{\dagger}\widehat{\boldsymbol{R}}$ where $\widehat{\boldsymbol{C}}$ is the output of a given RPCA algorithm whose input is $\widetilde{\boldsymbol{C}}$. The main results pertaining to this step are in section 3.2.

3.1. How incoherence passes to submatrices. We begin with our first main result that gives generic bounds on incoherence and condition numbers of column submatrices of low-rank matrices. Since this matter is of general interest, in this subsection, we will suppose that we have access to a low-rank matrix $\boldsymbol{L}$ itself, which is not true in the RPCA problem. We stress that this assumption is not at all necessary, but we will show that one can obtain concrete bounds for incoherence of submatrices compared to incoherence of the full matrix for a variety of column sampling schemes.

Theorem 3.1. *Suppose $\boldsymbol{L} \in \mathbb{R}^{m\times n}$ has rank r and satisfies (A1). Let $J \subseteq [n]$ such that $\boldsymbol{C} = \boldsymbol{L}(:, J)$ has rank r, and denote $\beta := \sqrt{\frac{|J|}{n}} \left\|\boldsymbol{V}_{\boldsymbol{L}}(J, :)^{\dagger}\right\|_2$. Then*

1. $\max_i \|\boldsymbol{W}_{\boldsymbol{C}}^T \boldsymbol{e}_i\|_2 \leq \sqrt{\frac{\mu_1(\boldsymbol{L})r}{m}}$,
2. $\max_i \|\boldsymbol{V}_{\boldsymbol{C}}^T \boldsymbol{e}_i\|_2 \leq \beta\kappa(\boldsymbol{L})\sqrt{\frac{\mu_2(\boldsymbol{L})r}{|J|}}$,
3. $\kappa(\boldsymbol{C}) \leq \beta\sqrt{\mu_2(\boldsymbol{L})r}\kappa(\boldsymbol{L})$.

In particular,

$$\mu_1(\boldsymbol{C}) \leq \mu_1(\boldsymbol{L}) \quad \text{and} \quad \mu_2(\boldsymbol{C}) \leq \beta^2\kappa(\boldsymbol{L})^2\mu_2(\boldsymbol{L}).$$

Remark 3.2. We assume the condition number $\kappa(\boldsymbol{L}) = \mathcal{O}(1)$, which is commonly assumed in many RPCA papers [11, 12, 53]. According to Theorem 3.1, a well-conditioned $\boldsymbol{L}$ with good incoherence ensures the corresponding properties of its submatrices, and this is a key to the success of RCUR.

Naturally, the value of β will depend heavily on the sampling method utilized to select the column indices J. Thus, we desire algorithms for selecting column submatrices of $\boldsymbol{L}$ for which $\beta \ll \mathcal{O}(n)$. However, it is typically the case that $|J|$ and β are inversely proportional and also that judicious sampling of smaller column indices can be significantly more costly than, say, uniform random sampling. Consequently, we will seek a trade-off for which both $|J|$ and β are small compared to n, while simultaneously maintaining low complexity in the sampling scheme that produces the column indices J.

Let us now better illustrate the character of the bounds given in Theorem 3.1 by some specific examples. We first consider deterministic sampling methods and begin with an existential bound based on maximum volume sampling. The *volume* of a matrix $M \in \mathbb{R}^{s\times t}$ is $\prod_{i=1}^{\min\{s,t\}} \sigma_i(M)$.

Proposition 3.3 ([42, Lemma 1]). *Invoke the notations and assumptions of Theorem 3.1, and assume that $|J| = \ell$. If J is chosen so that $\boldsymbol{V}_{\boldsymbol{L}}(J,:)$ is the maximal volume submatrix of $\boldsymbol{V}_{\boldsymbol{L}}$ of size $\ell \times r$, then*

$$\beta \le \sqrt{\frac{\ell}{n} + \frac{r\ell(n-\ell)}{n(\ell - r + 1)}}.$$

Note that if $\ell = r$, then $\beta = \mathcal{O}(r)$, whereas if $\ell = n$, then $\beta = 1$. Moreover, the function on the right-hand side of the estimate in Proposition 3.3 is decreasing as ℓ increases for fixed r and n. Thus maximal volume sampling achieves a good bound for β. However, these bounds come at substantial cost: finding maximal volume submatrices is NP-hard [4]. There are approximation algorithms available [25, 38, 41], but analysis of how close their output is to the optimal solution is unknown.

Regarding deterministic column sampling methods, Avron and Boutsidis prove upper bounds on β for a greedy column selection algorithm [3]. We will discuss this algorithm in more detail later, but for now, we mention the following.

Theorem 3.4 ([3, Theorem 3.1]). *Let $\boldsymbol{L} \in \mathbb{R}^{m\times n}$ with $\operatorname{rank}(\boldsymbol{L}) = r$. Suppose $\boldsymbol{L}$ satisfies (A1). Then Algorithm C.1 ([3, Algorithm 1]) applied to $\boldsymbol{V}_{\boldsymbol{L}}$ yields column indices $J \subseteq [n]$ with $|J| = r$ for which $\boldsymbol{C} = \boldsymbol{L}(:, J)$ satisfies $\operatorname{rank}(\boldsymbol{C}) = \operatorname{rank}(\boldsymbol{L})$, and $\beta \le r$.*

The greedy algorithm of [3] applied to $\boldsymbol{V}_{\boldsymbol{L}}$ requires $\mathcal{O}(nr^2 + nr(n-r))$ operations to select exactly r columns and yield the bound $\beta \le r$. Thus, this method is more feasible than maximal volume sampling. Both of the above procedures deterministically select exactly r columns of $\boldsymbol{L}$ and yield a good bound for β, but at nontrivial computational cost. We now turn our gaze to random sampling methods, which we will see can yield good bounds on β with high probability at trivial computational cost.

By far the easiest and cheapest sampling method is to choose J from $[n]$ via uniform sampling. While uniform sampling may not be successful for arbitrary matrices, it is known to be typically successful for incoherent matrices. The following theorem, derived from estimates

of pseudoinverses of orthogonal matrices due to Tropp [48], provides a bound for β under this paradigm.

Theorem 3.5. *Suppose that $\boldsymbol{L} \in \mathbb{R}^{m\times n}$ has rank r and satisfies condition (A1), and $\boldsymbol{V_L} \in \mathbb{R}^{n\times r}$ are its first r right singular vectors. Suppose that $J \subseteq [n]$ is chosen by sampling uniformly without replacement to yield $\boldsymbol{C} = \boldsymbol{L}(:, J)$ and that $|J| \geq \gamma\mu_2(\boldsymbol{L})r$ for some $\gamma > 0$. Then the quantity $\beta = \sqrt{|J|/n}\|(\boldsymbol{V_L}(J,:))^\dagger\|_2$ satisfies*

1. $\beta \leq \frac{1}{\sqrt{1-\delta}}$,
2. $\max_i \|\boldsymbol{V_C}^T\boldsymbol{e}_i\|_2 \leq \beta\kappa(\boldsymbol{L})\sqrt{\frac{\mu_2(\boldsymbol{L})r}{|J|}}$, *and*
3. $\kappa(\boldsymbol{C}) \leq \beta\sqrt{\mu_2(\boldsymbol{L})r}\kappa(\boldsymbol{L})$

with probability of at least $1 - \frac{r}{e^{\gamma(\delta+(1-\delta)\log(1-\delta))}}$. In particular, $\mu_1(\boldsymbol{C}) \leq \mu_1(\boldsymbol{L})$ and $\mu_2(\boldsymbol{C}) \leq \beta^2\kappa(\boldsymbol{L})^2\mu_2(\boldsymbol{L})$ with the given success probability.

Let us provide a concrete choice of parameters for illustration, though we stress that the estimate in Theorem 3.5 is quite flexible.

Corollary 3.6. *With the notations and assumptions of Theorem 3.5, put $\delta = 0.99$. Then sampling $|J| \geq 1.06\mu_2(L)r\log(rn)$ columns from $[n]$ uniformly without replacement yields*

1. $\beta \leq 10$,
2. $\max_i \|V_C^T e_i\|_2 \leq 10\kappa(L)\sqrt{\frac{\mu_2(L)r}{|J|}} \leq 10\kappa(\boldsymbol{L})(\log(rn))^{-\frac{1}{2}}$, *and*
3. $\kappa(C) \leq 10\sqrt{\mu_2(L)r}\kappa(L)$

with probability at least $1 - \frac{1}{n}$. In particular, $\mu_1(C) \leq \mu_1(L)$ and $\mu_2(C) \leq 100\kappa(L)^2\mu_2(L)$ with the given success probability.

A brief remark here is needed on sampling order in Corollary 3.6—the $r\log(rn)$ term is mild but necessary to achieve success probability which decays with the problem size, n; one can derive a similar result in which the success probability decays polynomially with the rank, r, while requiring only $|J| \gtrsim r\log(r)$; such a bound could be useful in problems in which the rank is not minuscule. Later, this sampling order may be compared with those of CUR approximations and will be seen to be favorable.

For ease of exposition above, we have focused solely on the case of column selection; however, we note that analogous results are easily obtained for row selection by taking transpose in the formulae above. For clarity, since we will subsequently be interested in selecting both column and row submatrices to form CUR decompositions, we state the combined bound on incoherence here.

Corollary 3.7. *With the notations and assumptions of Corollary 3.6, suppose also that $I \subseteq [m]$ with $|I| \geq 1.06\mu_1(\boldsymbol{L})r\log(rm)$ is sampled uniformly without replacement, and that $\boldsymbol{R} = \boldsymbol{L}(I,:)$. If $\beta' := \sqrt{|I|/m}\|(\boldsymbol{W_L}(I,:))^\dagger\|_2$, then with probability at least $(1-\frac{1}{n})(1-\frac{1}{m})$, the conclusion of Corollary 3.6 holds, and so do the following:*

1. $\beta' \leq 10$,
2. $\max_i \|\boldsymbol{V_R}^T\boldsymbol{e}_i\|_2 \leq \sqrt{\frac{\mu_2(\boldsymbol{L})r}{n}}$,
3. $\max_i \|\boldsymbol{W_R}^T\boldsymbol{e}_i\|_2 \leq 10\kappa(\boldsymbol{L})\sqrt{\frac{\mu_1(\boldsymbol{L})r}{|I|}}$,
4. $\kappa(\boldsymbol{R}) \leq 10\sqrt{\mu_1(\boldsymbol{L})r}\kappa(\boldsymbol{L})$.

In particular, $\mu_1(\boldsymbol{R}) \leq 100\kappa(\boldsymbol{L})^2\mu_1(\boldsymbol{L})$, and $\mu_2(\boldsymbol{R}) \leq \mu_2(\boldsymbol{L})$.

Remark 3.8. Note that if one used more sophisticated random sampling techniques, such as leverage score sampling [37] or volume sampling [18], one could potentially obtain good bounds on β with larger success probability than in the results stated here. After all, these distributions take into account more information about the structure of the column and row space of the matrix. However, using these comes at a nontrivial computational cost for computation of the probability distributions ($\mathcal{O}(mnk)$ for leverage scores and approximately $\mathcal{O}(\max\{m^2, n^2\})$ for volume sampling). Given that a constant bound for β can be obtained with relatively large probability via uniform sampling, which requires no computation, this method should be preferred for incoherent matrices.

There are faster methods than those mentioned above that approximate leverage score distributions, e.g., [21], but to be effective in the RPCA model, one needs to approximate leverage scores of $\boldsymbol{L}$ from observations of $\boldsymbol{L} + \boldsymbol{S}$. Leverage scores are unstable under addition of sparse, arbitrary magnitude outliers, so without imposing additional restrictions on $\boldsymbol{S}$, such approximations are likely unsuitable for the RPCA problem.

3.2. RCUR decomposition. Now we are ready to state our main algorithm. The idea of Algorithm 3.1 is simple yet effective due to the computational ease of uniform sampling and the known success of CUR decompositions. The algorithm may be considered a proto-type in that it has an RPCA algorithm as a subroutine in lines 5 and 6. We emphasize that any RPCA algorithm can be used in these lines, and we will quantify the accuracy and computational complexity of our algorithm based on those of the RPCA subroutine.

Remark 3.9. We note that Algorithm 3.1 and all of the theoretical results related to it here hold for both sampling column and row indices uniformly *with* or *without* replacement. For simplicity, we state all results here for sampling *with* replacement. The only minor change to any results here for sampling *without* replacement is that some of the constants in the big-$\mathcal{O}$ notations and success probabilities will change, but typically the change is small.

Now let us state a sample result illustrating the type of guarantee Algorithm 3.1 gives. To do so, we assume that the AltProj RPCA algorithm of [39] is used in lines 5 and 6.

Theorem 3.10. *Let* $\boldsymbol{L} \in \mathbb{R}^{m\times n}$ *and* $\boldsymbol{S} \in \mathbb{R}^{m\times n}$ *satisfy assumptions* (A1) *and* (A2)*, respectively, with* $\operatorname{rank}(\boldsymbol{L}) = r$*. Let* $\varepsilon, \delta \in (0,1)$ *and* $\alpha = \mathcal{O}(\frac{1-\delta}{\kappa(\boldsymbol{L})^2(\mu_1(\boldsymbol{L})\vee\mu_2(\boldsymbol{L}))})$*. Suppose that*

$$|I| \geq c_1 \max\{\mu_1(\boldsymbol{L}) r \log(m), \log(m)/\alpha\} \quad \text{and} \quad |J| \geq c_2 \max\{\mu_2(\boldsymbol{L}) r \log(n), \log(n)/\alpha\}$$

Algorithm 3.1. Uniform sampling RCUR

1: **Input:** $\boldsymbol{D} = \boldsymbol{L} + \boldsymbol{S} \in \mathbb{R}^{m\times n}$ with $\operatorname{rank}(\boldsymbol{L}) = r$; RPCA: the chosen RPCA solver.
2: **Output:** $\widehat{\boldsymbol{L}}$: approximation of $\boldsymbol{L}$.
3: **Initialization:** Draw sampling indices $I \subseteq [m]$, $J \subseteq [n]$ uniformly.
4: $\widetilde{\boldsymbol{C}} = \boldsymbol{D}(:, J)$, $\widetilde{\boldsymbol{R}} = \boldsymbol{D}(I, :)$
5: $[\widehat{\boldsymbol{C}}, \widehat{\boldsymbol{S}}_{\boldsymbol{C}}] = \text{RPCA}(\widetilde{\boldsymbol{C}}, r)$
6: $[\widehat{\boldsymbol{R}}, \widehat{\boldsymbol{S}}_{\boldsymbol{R}}] = \text{RPCA}(\widetilde{\boldsymbol{R}}, r)$
7: $\widehat{\boldsymbol{L}} = \widehat{\boldsymbol{C}}(\widehat{\boldsymbol{C}}(I, :))^{\dagger}\widehat{\boldsymbol{R}}$

rows and columns, respectively, are chosen uniformly without replacement and that AltProj [39] *is used as the* RPCA *subroutine in lines* 5 *and* 6 *of Algorithm* 3.1*. Then by running* $\mathcal{O}(r\log(\sqrt{\frac{mn}{|I||J|}}\frac{\kappa(\boldsymbol{L})}{(1-\delta)\varepsilon}))$ *iterations of* AltProj*, the output of Algorithm* 3.1 *satisfies*

$$\frac{\left\|\boldsymbol{L}-\widehat{\boldsymbol{L}}\right\|_2}{\|\boldsymbol{L}\|_2} \le \varepsilon\kappa(\boldsymbol{L})^{-1}$$

with probability at least $1-\frac{r}{n^{c_2(\delta+(1-\delta)\log(1-\delta))}}-\frac{r}{m^{c_1(\delta-(1-\delta)\log(1-\delta))}}-\frac{1}{n}-\frac{1}{m}$.

Moreover, the total complexity of Algorithm 3.1 *with* AltProj *as a subroutine is*

$$\mathcal{O}\left(\left(mr^3\log(n)+nr^3\log(m)\right)\log\left(\sqrt{\frac{mn}{|I||J|}}\frac{\kappa(\boldsymbol{L})}{(1-\delta)\varepsilon}\right)\right).$$

3.3. Hybrid algorithms for RCUR approximation. The output of Algorithm 3.1 is an approximation of the form $\boldsymbol{L}\approx\widehat{\boldsymbol{C}}\widehat{\boldsymbol{U}}^{\dagger}\widehat{\boldsymbol{R}}$, where $\widehat{\boldsymbol{C}}$ and $\widehat{\boldsymbol{R}}$ are cleaned versions of columns and rows, respectively, of $\boldsymbol{D}$. Put a different way, $\widehat{\boldsymbol{C}}$ approximates the column space of $\boldsymbol{L}$, and $\widehat{\boldsymbol{R}}$ approximates the row space of $\boldsymbol{L}$. However, these consist of $\mathcal{O}(r\log(rn))$ and $\mathcal{O}(r\log(rm))$ vectors, respectively, while the dimensions of the column and row space of $\boldsymbol{L}$ are both r; that is, we have redundant approximate bases for these spaces. We are interested in determining from these the most relevant vectors that represent the largest subspaces of both the column and row space of $\boldsymbol{L}$. In particular, from the randomly sampled indices I, J, we would prefer to choose *exactly* r elements of each. This problem in general is called the *column subset selection problem*: given $\widehat{\boldsymbol{C}}\in\mathbb{R}^{m\times\ell}$, choose $I_1\subseteq[\ell]$ (and set $\widehat{\boldsymbol{C}}_1$) with $|I_1|=r$ which minimizes $\|\widehat{\boldsymbol{C}}-\widehat{\boldsymbol{C}}_1\widehat{\boldsymbol{C}}_1^{\dagger}\widehat{\boldsymbol{C}}\|_2$ over all possible column submatrices of $\widehat{\boldsymbol{C}}_1$ consisting of exactly r columns. This problem is NP-complete [43] (see also [17] for other complexity results), but there are both randomized and deterministic approximation algorithms; a small subsampling of prior art on this problem is [3, 5, 6, 19, 47].

We consider now what happens when we apply a deterministic column subset selection algorithm to the matrices $\widehat{\boldsymbol{C}}$ and $\widehat{\boldsymbol{R}}$ given by the output of lines 5 and 6 of Algorithm 3.1 to choose exactly r columns and rows thereof. We state this algorithm generally but specialize our analysis to the greedy algorithm of [3] (described fully here in the appendix as Algorithm C.1) to state a concrete theorem. For clarity, we state the general algorithm here as Algorithm 3.2. Again, RPCA is any such algorithm the user specifies, and DeterministicCS$(\boldsymbol{A},r)$ is any algorithm that deterministically selects exactly r columns of $\boldsymbol{A}$ (I_1 and J_1 in Algorithm 3.2 are the indices chosen by the deterministic column selection algorithm).

The following theorem describes the accuracy of the output of Algorithm 3.2.

Theorem 3.11. *Take the notations and assumptions of Theorem* 3.10*. Let* $\widehat{\boldsymbol{C}}$ *and* $\widehat{\boldsymbol{R}}$ *be the intermediate outputs of Algorithm* 3.2 *in lines* 5 *and* 6 *with* AltProj *being used as the RPCA subroutine, where the latter runs* $\mathcal{O}(r\log(\frac{229\kappa(\boldsymbol{L})^5 r\sqrt{mn\mu_1(\boldsymbol{L})\mu_2(\boldsymbol{L})}}{\varepsilon(1-\delta)^2\sigma_{\max}(\boldsymbol{L})}))$ *iterations. Let* $\widehat{\boldsymbol{C}}_1=\widehat{\boldsymbol{C}}(:,J_1)$*,* $\widehat{\boldsymbol{U}}_1=\widehat{\boldsymbol{C}}(I_1,:)$*, and* $\widehat{\boldsymbol{R}}_1=\widehat{\boldsymbol{R}}(I_1,:)$*, where* I_1 *and* J_1 *are obtained by applying Algorithm* C.1 *on* $\widehat{\boldsymbol{R}}$ *and* $\widehat{\boldsymbol{C}}$ *with* $|I_1|=|J_1|=r$ *in lines* 8 *and* 9*. Then with probability at least* $1-\frac{r}{n^{c_2(\delta+(1-\delta)\log(1-\delta))}}-\frac{r}{m^{c_1(\delta-(1-\delta)\log(1-\delta))}}-\frac{1}{n}-\frac{1}{m}$,

$$\left\|\boldsymbol{L}-\widehat{\boldsymbol{C}}_1\widehat{\boldsymbol{U}}_1^{\dagger}\widehat{\boldsymbol{R}}_1\right\|_2\le\frac{1}{3}\varepsilon\sigma_{\min}(\boldsymbol{L}).$$

Algorithm 3.2. Uniform + deterministic RCUR

Input: $\boldsymbol{D} = \boldsymbol{L} + \boldsymbol{S} \in \mathbb{R}^{m\times n}$ with $\operatorname{rank}(\boldsymbol{L}) = r$; RPCA: the chosen RPCA solver; DeterministicCS: the chosen deterministic row/column selection method.
Output: $\widehat{\boldsymbol{L}}$: approximation of $\boldsymbol{L}$.
Initialization: Draw sampling indices $I \subseteq [m]$, $J \subseteq [n]$ uniformly.
$\widetilde{\boldsymbol{C}} = \boldsymbol{D}(:, J)$, $\widetilde{\boldsymbol{R}} = \boldsymbol{D}(I, :)$
$[\widehat{\boldsymbol{C}}, \widehat{\boldsymbol{S}}_{\boldsymbol{C}}] = \text{RPCA}(\widetilde{\boldsymbol{C}}, r)$
$[\widehat{\boldsymbol{R}}, \widehat{\boldsymbol{S}}_{\boldsymbol{R}}] = \text{RPCA}(\widetilde{\boldsymbol{R}}, r)$
$[\widehat{\boldsymbol{C}}_1, \widehat{\boldsymbol{S}}_{\boldsymbol{C}_1}, J_1] = \text{DeterministicCS}(\widehat{\boldsymbol{C}}, r)$
$[\widehat{\boldsymbol{R}}_1, \widehat{\boldsymbol{S}}_{\boldsymbol{R}_1}, I_1] = \text{DeterministicCS}(\widehat{\boldsymbol{R}}^T, r)$
$\widehat{\boldsymbol{L}} = \widehat{\boldsymbol{C}}_1(\widehat{\boldsymbol{C}}_1(I_1, :))^\dagger \widehat{\boldsymbol{R}}_1$

Remark 3.12. We note for the reader two things here. First, the proofs of the approximation theorems here are valid for the Frobenius norm as well as the smaller spectral norm; for presentation purposes we choose the latter. Second, the bounds in Theorems 3.10 and 3.11 are the same by dividing by $\|\boldsymbol{L}\|_2$; e.g., the bound in Theorem 3.11 is the same as the relative error bound

$$\frac{\|\boldsymbol{L} - \widehat{\boldsymbol{C}}_1 \widehat{\boldsymbol{U}}_1^\dagger \widehat{\boldsymbol{R}}\|_2}{\|\boldsymbol{L}\|_2} \leq \frac{1}{3}\varepsilon\kappa(\boldsymbol{L})^{-1}.$$

3.4. Deterministic column selection. In Appendices D and E, we study deterministic column sampling methods, in particular via the greedy algorithm of Avron and Boutsidis [3]. We are able to prove that, with proper preconditioning, sampling columns and rows via the greedy algorithm on the singular vectors of $\boldsymbol{D}$ produce submatrices of $\boldsymbol{L}$ with similar incoherence (see Remark D.2). In particular, we find that the method described in Appendix D produces deterministically a submatrix $\boldsymbol{C}$ for which $\beta \leq 4\kappa(\boldsymbol{L})\sqrt{r}$. However, this procedure does not guarantee $\mathcal{O}(\alpha)$-sparsity of the corresponding submatrices of $\boldsymbol{S}$. This we leave as an open problem for future work.

4. Passing properties to submatrices.

In this section, we study how three properties of a matrix pass to column submatrices thereof: incoherence, condition number, and sparsity. These results are of independent interest but are also necessary to study the use of RPCA algorithms on submatrices.

4.1. Incoherence.

In this subsection, we study the relation of the incoherence of the matrix $\boldsymbol{L} \in \mathbb{R}^{m\times n}$ to that of a column submatrix of it, say $\boldsymbol{C} = \boldsymbol{L}(:, J) \in \mathbb{R}^{m\times \ell}$. We find that column incoherence is unchanged, while row incoherence inflates by no more than a factor of $\beta\kappa(\boldsymbol{L})$.

Let us begin by the observation that if $\boldsymbol{L}$ has column incoherence as in the first inequality of (A1), then any column submatrix $\boldsymbol{C}$ has the same left incoherence.

Lemma 4.1. *Suppose* $\boldsymbol{L} \in \mathbb{R}^{m\times n}$ *satisfies* (A1). *For any* $J \subseteq [n]$ *such that* $\operatorname{rank}(\boldsymbol{C}) = \operatorname{rank}(\boldsymbol{L})$ *with* $\boldsymbol{C} = \boldsymbol{L}(:, J)$, *then* $\boldsymbol{C}$ *satisfies*

$$\max_i \left\|\boldsymbol{W}_{\boldsymbol{C}}^T \boldsymbol{e}_i\right\|_2 \leq \sqrt{\frac{\mu_1(\boldsymbol{L})r}{m}}.$$

Proof. Notice that $\boldsymbol{C} = \boldsymbol{W_L}\boldsymbol{\Sigma_L}(\boldsymbol{V_L}(J,:))^T = \boldsymbol{W_L}\widetilde{\boldsymbol{W}}_{\boldsymbol{C}}\boldsymbol{\Sigma_C}\boldsymbol{V_C}^T$, where $\boldsymbol{\Sigma_L}(\boldsymbol{V_L}(J,:))^T$ has the compact SVD decomposition $\widetilde{\boldsymbol{W}}_{\boldsymbol{C}}\boldsymbol{\Sigma_C}\boldsymbol{V_C}^T$ with $\widetilde{\boldsymbol{W}}_{\boldsymbol{C}} \in \mathbb{R}^{r\times r}$ being an orthonormal matrix and $\boldsymbol{V_C}^T \in \mathbb{R}^{r\times|J|}$. Therefore, $\boldsymbol{W_C} = \boldsymbol{W_L}\widetilde{\boldsymbol{W}}_{\boldsymbol{C}}$. Thus, we have

$$\max_i \left\|\boldsymbol{W_C}^T\boldsymbol{e}_i\right\|_2 = \max_i \left\|\widetilde{\boldsymbol{W}}_{\boldsymbol{C}}^T\boldsymbol{W_L}^T\boldsymbol{e}_i\right\|_2 = \max_i \left\|\boldsymbol{W_L}^T\boldsymbol{e}_i\right\|_2 \le \sqrt{\frac{\mu_1(\boldsymbol{L})r}{m}}.$$

Lemma 4.1 implies that we need not be concerned with the column incoherence of any column submatrix $\boldsymbol{C}$ of $\boldsymbol{L}$ given that it always matches that of the matrix it is obtained from. Now we need to understand the incoherence for $\boldsymbol{V_C}$, which requires some more work. First, let us make two preliminary observations.

Lemma 4.2. *Let $\boldsymbol{L} \in \mathbb{R}^{m\times n}$ with rank r satisfy (A1). Then for any $J \subseteq [n]$, $\boldsymbol{C} = \boldsymbol{L}(:,J)$ satisfies*

$$\max_i \left\|\boldsymbol{V_C}^T\boldsymbol{e}_i\right\|_2 \le \kappa(\boldsymbol{L})\frac{\left\|\boldsymbol{C}^\dagger\right\|_2}{\left\|\boldsymbol{L}^\dagger\right\|_2}\sqrt{\frac{\mu_2(\boldsymbol{L})r}{n}}.$$

Proof. Notice that $\boldsymbol{V_C}^T = \boldsymbol{\Sigma_C}^{-1}\boldsymbol{W_C}^T\boldsymbol{C}$. Thus for any i,

$$\begin{aligned}\left\|\boldsymbol{V_C}^T\boldsymbol{e}_i\right\|_2 &= \left\|\boldsymbol{\Sigma_C}^{-1}\boldsymbol{W_C}^T\boldsymbol{C}\boldsymbol{e}_i\right\|_2\\ &\le \left\|\boldsymbol{\Sigma_C}^{-1}\right\|_2\left\|\boldsymbol{W_C}^T\right\|_2\left\|\boldsymbol{C}\boldsymbol{e}_i\right\|_2\\ &\le \frac{1}{\sigma_{\min}(\boldsymbol{C})}\left\|\boldsymbol{C}\boldsymbol{e}_i\right\|_2.\end{aligned}$$

Since $\boldsymbol{C} = \boldsymbol{L}(:,J) = \boldsymbol{W_L}\boldsymbol{\Sigma_L}(\boldsymbol{V_L}(J,:))^T$,

$$\begin{aligned}\frac{1}{\sigma_{\min}(\boldsymbol{C})}\left\|\boldsymbol{C}e_i\right\|_2 &= \frac{1}{\sigma_{\min}(\boldsymbol{C})}\left\|\boldsymbol{W_L}\boldsymbol{\Sigma_L}(\boldsymbol{V}(J,:))^T\boldsymbol{e}_i\right\|_2\\ &\le \frac{\sigma_1(\boldsymbol{L})}{\sigma_{\min}(\boldsymbol{C})}\left\|(\boldsymbol{V_L}(J,:))^T\boldsymbol{e}_i\right\|_2\\ &\le \frac{\sigma_1(\boldsymbol{L})}{\sigma_{\min}(\boldsymbol{C})}\left\|\boldsymbol{V_L}^T\boldsymbol{e}_i\right\|_2\\ &\le \kappa(\boldsymbol{L})\frac{\sigma_{\min}(\boldsymbol{L})}{\sigma_{\min}(\boldsymbol{C})}\sqrt{\frac{\mu_2(\boldsymbol{L})r}{n}}\\ &= \kappa(\boldsymbol{L})\frac{\left\|\boldsymbol{C}^\dagger\right\|_2}{\left\|\boldsymbol{L}^\dagger\right\|_2}\sqrt{\frac{\mu_2(\boldsymbol{L})r}{n}}.\end{aligned}$$

Estimation of the quantity $\frac{\|\boldsymbol{C}^\dagger\|_2}{\|\boldsymbol{L}^\dagger\|_2}$ will generally depend heavily on how the column submatrix $\boldsymbol{C}$ is selected. One way to estimate this is to consider the norm of the pseudoinverse of the corresponding column submatrix of $\boldsymbol{V_L}$ as follows.

Lemma 4.3. *Let $\boldsymbol{L} \in \mathbb{R}^{m\times n}$ and $\boldsymbol{C} = \boldsymbol{L}(:,J)$ for some $J \subseteq [n]$. Then*

$$\frac{\left\|\boldsymbol{C}^\dagger\right\|_2}{\left\|\boldsymbol{L}^\dagger\right\|_2} \le \left\|(\boldsymbol{V_L}(J,:))^\dagger\right\|_2.$$

Proof. Note that $\boldsymbol{C} = \boldsymbol{W_L}\boldsymbol{\Sigma_L}\left(\boldsymbol{V_L}(J,:)\right)^T$. Then one has

$$\boldsymbol{C}^\dagger = \left(\boldsymbol{\Sigma_L}\boldsymbol{V_L}(J,:)^T\right)^\dagger \boldsymbol{W_L}^T = \left(\boldsymbol{V_L}(J,:)^T\right)^\dagger \boldsymbol{\Sigma_L}^{-1}\boldsymbol{W_L}^T,$$

where the first equality comes from the fact that $\boldsymbol{W_L}$ has orthonormal columns and $\boldsymbol{W_L}^\dagger = \boldsymbol{W_L}^T$, while the second equality follows from the fact that $\boldsymbol{\Sigma_L}$ has full column rank and $\boldsymbol{V_L}(J,:)^T$ has full row rank. With this in hand, we have

$$\left\|\boldsymbol{C}^\dagger\right\|_2 = \left\|\left(\left(\boldsymbol{V_L}(J,:)\right)^T\right)^\dagger \boldsymbol{\Sigma_L}^{-1}\boldsymbol{W_L}^T\right\|_2 \le \left\|\left(\boldsymbol{V_L}(J,:)^\dagger\right)^T\right\|_2 \left\|\boldsymbol{\Sigma_L}^{-1}\right\|_2 = \left\|\boldsymbol{L}^\dagger\right\|_2 \left\|\left(\boldsymbol{V_L}(J,:)\right)^\dagger\right\|_2,$$

whence the result. ∎

Combining the conclusions of Lemmata 4.1–4.3, we see that

$$\max_i \left\|\boldsymbol{V_C}^T\boldsymbol{e}_i\right\|_2 \le \kappa(\boldsymbol{L})\sqrt{\frac{|J|}{n}}\sqrt{\frac{\mu_2(\boldsymbol{L})r}{|J|}}\left\|\left(\boldsymbol{V_L}(J,:)\right)^\dagger\right\|_2 = \beta\kappa(\boldsymbol{L})\sqrt{\frac{\mu_2(\boldsymbol{L})r}{|J|}}. \tag{4.1}$$

Our primary concern now is to attempt to bound the quantity $\beta = \sqrt{\frac{|J|}{n}}\|(\boldsymbol{V_L}(J,:))^\dagger\|_2$. Unfortunately, there are relatively few general results for estimating pseudoinverses of submatrices of orthogonal matrices except in special instances. We make this into the parameter β of Theorem 3.1.

4.2. Condition number. Many RPCA algorithms require an estimate on the condition number of the matrix that is input into the algorithm to provide good theoretical guarantees. Here, we estimate $\kappa(\boldsymbol{C})$ in terms of $\kappa(\boldsymbol{L})$ in the case that $\boldsymbol{L}$ has a certain incoherence level.

Lemma 4.4. *Let $\boldsymbol{L} \in \mathbb{R}^{m\times n}$ have rank r and satisfy assumption (A1), and suppose $J \subseteq [n]$. Then*

1. $\|\boldsymbol{C}\|_2 \le \sqrt{\frac{\mu_2(\boldsymbol{L})r|J|}{n}}\|\boldsymbol{L}\|_2$;
2. $\kappa(\boldsymbol{C}) \le \sqrt{\mu_2(\boldsymbol{L})r}\kappa(\boldsymbol{L})\sqrt{\frac{|J|}{n}}\|\boldsymbol{V_L}(J,:)^\dagger\|_2$.

Proof. Since $\boldsymbol{L} = \boldsymbol{W_L}\boldsymbol{\Sigma_L}\boldsymbol{V_L}^T$ and $\boldsymbol{C} = \boldsymbol{L}(:,J)$, we have $\boldsymbol{C} = \boldsymbol{W_L}\boldsymbol{\Sigma_L}(\boldsymbol{V_L}(J,:))^T$. For all $\boldsymbol{x} \in \mathbb{R}^{|J|}$ with $\|\boldsymbol{x}\|_2 = 1$,

$$\begin{aligned}
\|\boldsymbol{C}\boldsymbol{x}\|_2 &= \left\|\boldsymbol{W_L}\boldsymbol{\Sigma_L}(\boldsymbol{V_L}(J,:))^T\boldsymbol{x}\right\|_2 \\
&\le \|\boldsymbol{\Sigma_L}\|_2\left\|(\boldsymbol{V_L}(J,:))^T\boldsymbol{x}\right\|_2 \\
&= \|\boldsymbol{L}\|_2\left\|(\boldsymbol{V_L}(J,:))^T\sum_{i=1}^{|J|}\boldsymbol{x}_i\boldsymbol{e}_i\right\|_2 \\
&\le \|\boldsymbol{L}\|_2\sum_{i\in J}|\boldsymbol{x}_i|\left\|(\boldsymbol{V_L}(J,:))^T\boldsymbol{e}_i\right\|_2 \\
&\le \|\boldsymbol{L}\|_2\sum_{i\in J}|\boldsymbol{x}_i|\sqrt{\frac{\mu_2(\boldsymbol{L})r}{n}} \\
&\le \|\boldsymbol{L}\|_2\sqrt{|J|}\|\boldsymbol{x}\|_2\sqrt{\frac{\mu_2(\boldsymbol{L})r}{n}} \\
&= \sqrt{\frac{\mu_2(\boldsymbol{L})r|J|}{n}}\|\boldsymbol{L}\|_2.
\end{aligned}$$

To prove the second inequality, note that

$$\kappa(\boldsymbol{C}) = \|\boldsymbol{C}\|_2 \left\|\boldsymbol{C}^\dagger\right\|_2 \leq \sqrt{\frac{\mu_2(\boldsymbol{L})r|J|}{n}} \|\boldsymbol{L}\|_2 \left\|\boldsymbol{V}_{\boldsymbol{L}}(J,:)^\dagger\right\|_2 \left\|\boldsymbol{L}^\dagger\right\|_2,$$

which gives the required result upon rearranging terms. ∎

The results above combine to prove the main theorem (Theorem 3.1) regarding the quantity β.

Proof of Theorem 3.1. The proof follows by combining Lemmata 4.1–4.4 and (4.1). ∎

4.3. Sparsity. Our ability to apply a RPCA algorithm to a column submatrix of $\boldsymbol{D}$ requires that the corresponding submatrix of $\boldsymbol{S}$ remains a sparse matrix. The content of the following proposition is that uniform sampling of sufficiently many ($\mathcal{O}(r\log n)$) columns yields a 2α-sparse submatrix if $\boldsymbol{S}$ was α-sparse.

Proposition 4.5. *Let $\boldsymbol{S}$ be an α-sparse matrix. We consider two cases of uniform sampling:*

1. *(With replacement) Consider the submatrix of $\boldsymbol{S} \in \mathbb{R}^{m\times n}$ formed by $|I|$ rows that were uniformly sampled with replacement, namely $\boldsymbol{R}$. Assume $|I| = cr\log(n)$ with $c \geq \frac{16}{3\alpha r}$; then $\boldsymbol{R} \in \mathbb{R}^{|I|\times n}$ is 2α-sparse with probability at least $1 - n^{-1}$. Similarly, $\boldsymbol{C} \in \mathbb{R}^{m\times|J|}$, formed by $|J| = cr\log(m)$ columns of $\boldsymbol{S}$ that were uniformly sampled with replacement, is also 2α-sparse with probability at least $1 - m^{-1}$.*
2. *(Without replacement) Consider the submatrix of $\boldsymbol{S} \in \mathbb{R}^{m\times n}$ formed by $|I|$ rows that were uniformly sampled without replacement, namely $\boldsymbol{R}$. Assume $|I| = cr\log(n)$ with $c \geq \frac{8}{\alpha r}$; then $\boldsymbol{R} \in \mathbb{R}^{|I|\times n}$ is 2α-sparse with probability at least $1 - 2n^{-1}$. Similarly, $\boldsymbol{C} \in \mathbb{R}^{m\times|J|}$, formed by $|J| = cr\log(m)$ columns of $\boldsymbol{S}$ that were uniformly sampled without replacement, is also 2α-sparse with probability at least $1 - 2m^{-1}$.*

Proof. Part 1. We first prove the version of uniform sampling *with* replacement. By the definition of α-sparsity, each row of $\boldsymbol{R}$ has no more than αn nonzero entries. Moreover, assume the jth column of $\boldsymbol{S}$ has exactly αn nonzero entries. Let X_i indicate whether $\boldsymbol{R}_{i,j}$ is nonzero. One can see that $X_i \sim \mathrm{Ber}(\alpha)$. Consequently, by Bernstein's inequality, we have

$$\begin{aligned}
\mathbb{P}\left(\sum_{i\in I} X_i > 2\alpha|I|\right) &= \mathbb{P}\left(\sum_{i\in I} X_i - \mathbb{E}(X_i) > \alpha|I|\right) \\
&\leq \exp\left(-\frac{\alpha^2|I|^2}{2|I|\alpha(1-\alpha) + \frac{2}{3}\alpha|I|}\right) \\
&\leq \exp\left(-\frac{\alpha|I|}{2(1-\alpha) + \frac{2}{3}}\right) \\
&\leq \exp\left(-2\log(n)\right) \\
&= n^{-2}
\end{aligned}$$

for the jth column of $\boldsymbol{R}$, where the last inequality uses the assumption $\alpha|I| \geq \frac{16}{3}\log(n)$. Note that the probability inequality still holds if $\boldsymbol{S}$ has less than αn nonzero entries. Furthermore, to have all n columns being 2α-sparse, we get

$$\mathbb{P}(\boldsymbol{R} \text{ is } 2\alpha\text{-sparse}) \geq (1 - n^{-2})^n \geq 1 - n^{-1}.$$

Hence, $\boldsymbol{R}$ is 2α-sparse with high probability; the proof that $\boldsymbol{C}$ is 2α-sparse with high probability is the same and so is omitted.

Part 2. Now, we prove the version of uniform sampling *without* replacement. By [29, Theorem 1], the Bernstein inequality for uniform sampling *without* replacement is

$$\mathbb{P}\left(\sum_{i=0}^{m} Y_i > t\right) \leq 2\exp\left(-\frac{t^2}{4m\sigma^2}\right),$$

where Y_i is zero-mean random variables with uniform probability and σ is the variance. Taking the same argument as in Part 1 except sampling uniformly *without* replacement, we have

$$\begin{aligned}
\mathbb{P}\left(\sum_{i\in I} X_i > 2\alpha|I|\right) &= \mathbb{P}\left(\sum_{i\in I} X_i - \mathbb{E}(X_i) > \alpha|I|\right) \\
&\leq 2\exp\left(-\frac{\alpha^2|I|^2}{4|I|\alpha(1-\alpha)}\right) \\
&\leq 2\exp\left(-\frac{\alpha|I|}{4(1-\alpha)}\right) \\
&\leq 2\exp\left(-2\log(n)\right) \\
&= 2n^{-2}
\end{aligned}$$

for the jth column of $\boldsymbol{R}$, where the last inequality uses the assumption $\alpha|I| \geq 8\log(n)$. Therefore,

$$\mathbb{P}(\boldsymbol{R} \text{ is } 2\alpha\text{-sparse}) \geq (1 - 2n^{-2})^n \geq 1 - 2n^{-1}$$

for $\boldsymbol{R}$ generated by uniform sampling *without* replacement. A similar result holds for $\boldsymbol{C}$. ■

5. Simulations. In this section, the effectiveness and computational efficiency of RCUR (i.e., Algorithm 3.1) is illustrated on two key imaging applications, video background-foreground separation and face modeling, and can be summarized in the following three ways:

1. We show that running time of RCUR on a video background subtraction task is typically an order of magnitude faster than the state-of-the-art RPCA algorithm. Qualitatively, we also show that the output produced by RCUR is similar to that of RPCA benchmarks.
2. We illustrate the interpretable nature of the dictionary vectors produced by RCUR.
3. We show that the running time of RCUR on face modeling is typically faster than that of RPCA and the output produced by RCUR is comparable to that of RPCA.

Moreover, for fair comparison, we use AccAltProj[2] as the RPCA algorithm in all the numerical experiments, including the RPCA subroutine that is used in lines 5–6 of Algorithm 3.1 and 3.2.

[2]AccAltProj [12] is a fast state-of-the-art RPCA algorithm that accelerates the popular AltProj [39].

5.1. Application to video background-foreground separation. Background-foreground separation is an important tool to extract moving objects from a static background in a video [8]. In this section, we compare the performance of the RCUR decomposition with that of RPCA on three public benchmark videos: *Shoppingmall*, *Restaurant*, and *OSU* datasets. The size of each frame of *Shoppingmall* is 256×320, that of *Restaurant* is 120×160, and that of *OSU* is 240×320. The total number of frames are 1000 for *Shoppingmall*, 3055 for *Restaurant*, and 1506 for *OSU*. Each video can be represented as a matrix $\boldsymbol{D} \in \mathbb{R}^{m \times n}$ by vectorizing and stacking the frames as columns of the matrix; thus the *Shoppingmall* video produces a matrix in $\mathbb{R}^{81,920 \times 1,000}$ while *Restaurant*'s is in $\mathbb{R}^{19,200 \times 3,055}$ and *OSU*'s is in $\mathbb{R}^{76,800 \times 1506}$. Then $\boldsymbol{D} = \boldsymbol{L} + \boldsymbol{S}$, where $\boldsymbol{L}$ is the background, and $\boldsymbol{S}$ contains the foreground elements treated as sparse outliers.

5.1.1. Runtime and quality. We apply Algorithm 3.1 to $\boldsymbol{D}$, the data matrix from *Shoppingmall*, *Restaurant*, and *OSU*. We take $r = 2$ to be the underlying rank of the background for each video sequence. Then according to Theorem 3.10, we randomly choose column $\widetilde{\boldsymbol{C}} = \boldsymbol{D}(:, J)$ and row $\widetilde{\boldsymbol{R}} = \boldsymbol{D}(I, :)$ submatrices of size $m \times 15r \log(n)$ and $25r \log(m) \times n$, respectively. Note that $\mu_1(\boldsymbol{L}) \neq \mu_2(\boldsymbol{L})$ in many real-world applications, and $\mu_1(\boldsymbol{L}) > \mu_2(\boldsymbol{L})$ under this problem setup. Then we apply the selected RPCA algorithm (i.e., AccAltProj [12]) on $\widetilde{\boldsymbol{C}}$ and $\widetilde{\boldsymbol{R}}$ (lines 5 and 6) and denote the corresponding output low-rank matrices by $\widehat{\boldsymbol{C}}$ and $\widehat{\boldsymbol{R}}$. We then approximate the static background of the video, which is a low-rank matrix $\boldsymbol{L}$, as $\boldsymbol{L} \approx \widehat{\boldsymbol{L}} = \widehat{\boldsymbol{C}}(\widehat{\boldsymbol{C}}(I, :))^{\dagger}\widehat{\boldsymbol{R}}$. The foreground $\boldsymbol{S}$, which represents the moving objects, is then approximated by $\boldsymbol{D} - \widehat{\boldsymbol{L}}$.

Table 1 shows the runtime for Algorithm 3.1 as well as the runtime for applying RPCA directly on $\boldsymbol{D}$. In all cases, Algorithm 3.1 runs substantially faster; in particular, on most problems, the runtime is a full order of magnitude faster than that of RPCA.

Running time is not the whole story; as the quality of the separation of foreground and background is of prime importance as well. Note that in the background-foreground separation tasks within the RPCA framework, there is no ground truth, and thus comparisons must be done qualitatively. Figures 1, 2, and 3 illustrate the outputs of RCUR and RPCA on *Restaurant*, *Shoppingmall*, and *OSU* sequences, respectively. Each figure presents three randomly selected frames from the full video sequence as well as the background and foreground images from each algorithm. In both cases, we note that the reconstruction algorithms yield comparable results; thus, the significant speed advantage afforded by RCUR is desirable in this particular RPCA task.

Table 1
Video size and runtime.

	Frame size	Frame number	Runtime (sec)	
			RCUR	RPCA
Shoppingmall	256×320	1000	7.69	44.30
Restaurant	120×160	3055	3.48	31.63
OSU	240×320	1506	10.39	68.62

Figure 1. *Restaurant: The first column contains three randomly selected frames from the original video. The middle two columns are the separated background and foreground outputs of RCUR, respectively. The right two columns are the separated background and foreground outputs of RPCA, respectively.*

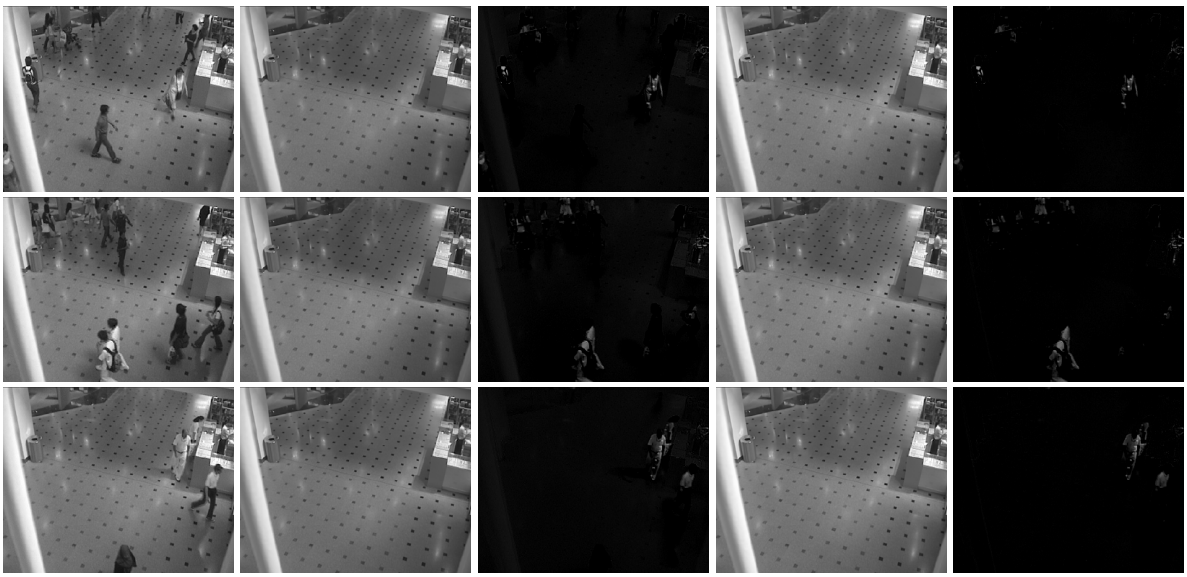

Figure 2. *Shoppingmall: The first column contains three randomly selected frames from the original video. The middle two columns are the separated background and foreground outputs of RCUR, respectively. The right two columns are the separated background and foreground outputs of RPCA, respectively.*

5.1.2. Interpretability: Extracting canonical frames. CUR decompositions have long been touted as providing better interpretability of data representations than abstract bases such as those obtained by QR or SVD (see, e.g., [33, 37, 44]), the idea being that a subset of the data itself is a good dictionary for the whole. This is also known as the self-expressivity

Figure 3. *OSU: The first column contains three randomly selected frames from the original video. The middle two columns are the separated background and foreground outputs of RCUR, respectively. The right two columns are the separated background and foreground outputs of RPCA, respectively.*

property of data and has been used to good effect in solving the subspace clustering problem [1, 2, 23, 31, 36, 49]. Additionally, as pointed out in [44], column selection can produce a dictionary which explains multiple variances which are not orthogonal to each other, in contrast to standard principal component analysis.

In the context of background-foreground separation, we explore this aspect of RCUR decompositions. To do so, we use Algorithm 3.2 to first uniformly sample columns (i.e., frames) of the video sequences according to the sampling requirement of Theorem 3.10; then following background-foreground separation via RPCA, we apply the deterministic sampling method of Algorithm C.1 to select exactly $r = 2$ frames from within those chosen in the first stage. Resulting from this procedure, we obtain two frames of the video background which are highly representative of the rest of the frames; in particular, they approximately minimize $\left\|\boldsymbol{L} - \boldsymbol{C}\boldsymbol{C}^{\dagger}\boldsymbol{L}\right\|_F$, the error when projecting the rest of the video frames onto the chosen two. Figure 4 shows the canonical background frames selected from this procedure compared with the first two basis vectors obtained by taking the SVD of the video matrix $\boldsymbol{D}$.

Note that in the first row of Figure 4, the left two panes are the two background frames obtained by Algorithm 3.2 and represent the two canonical states of the background: one with light shining through the window onto the floor and one with the light occluded. In contrast, the second SVD basis vector captures only the light portion of the background. In the other video sequences, the background is relatively static, and so our RCUR method produces two very similar canonical background frames. In contrast to this, the second SVD basis vectors are uninformative for these sequences as one might expect. Consequently, the CUR basis vectors capture the essential features of the different states of the background and are easily interpreted by the user. In contrast to this, the SVD basis vectors may appear meaningless in terms of the actual background. This difference stems from the fact that the

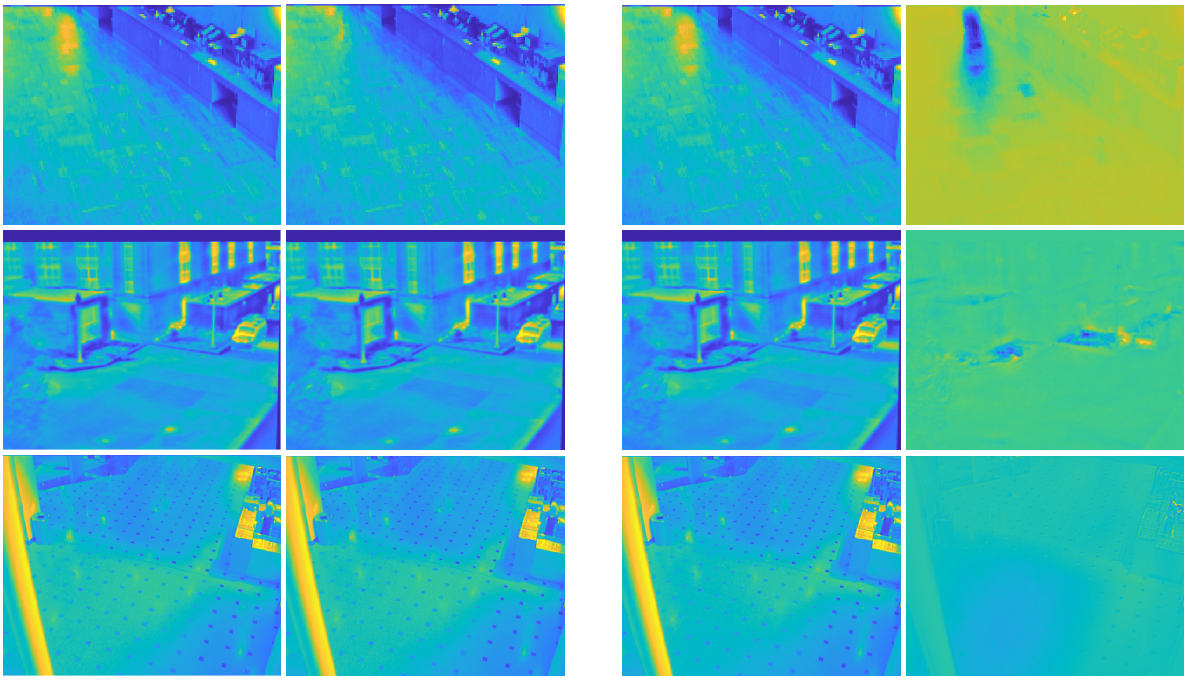

Figure 4. *Basis vectors for background of video sequences (Row 1: Restaurant, Row 2: OSU, Row 3: Shoppingmall). In each row, the first two columns are the selected frames by using the deterministic CUR algorithm described above, and the last two columns are the first two orthogonal basis vectors of the SVD of the recovered low-rank matrix, respectively.*

column vectors are approximating actual frames, whereas the SVD is capturing orthogonal images corresponding to decreasing variance.

5.2. Application to face modeling. In this section, we consider the problem of robust face modeling. We use the Extended Yale Face Database B (abbreviated *ExtYaleB*) [24] for a benchmark, which includes the face data of 28 human objects under 64 illumination conditions; each face image has size of 168×192. We vectorize the face images and stack the faces of the person together to form a data matrix, so we have 28 data matrices in $\mathbb{R}^{32,256\times 64}$. Then in terms of $\boldsymbol{D} = \boldsymbol{L} + \boldsymbol{S}$, the underlying face models form the low-rank $\boldsymbol{L}$, and the facial occlusions are sparse outlier $\boldsymbol{S}$. We apply Algorithm 3.1 to each of the face data matrix $\boldsymbol{D}$ where $r = 1$ is used to extract the face model. Again, since $\mu_2(\boldsymbol{L}) > \mu_1(\boldsymbol{L})$ in this problem, we randomly choose column $\widetilde{\boldsymbol{C}} = \boldsymbol{D}(:, J)$ and row $\widetilde{\boldsymbol{R}} = \boldsymbol{D}(I, :)$ submatrices of size $m \times 10r\log(n)$ and $25r\log(m) \times n$, respectively. The rest of the experimental setup is same as section 5.1.

In Table 2, we report the total runtime for Algorithm 3.1 and RPCA on the face modeling task. Clearly, Algorithm 3.1 runs faster than RPCA, but the advantage is not as large as in the video background-foreground separation task. The reason is that the face data matrix is too rectangular (i.e., $n = 64$ is too small), which gives less room for acceleration. Moreover, we present the visual face modeling results of a selected human object in Figure 5, wherein we find both algorithms achieve the desired modeling quality.

Table 2
Face data size and runtime.

	Image size	Image number per person	Person number	Total runtime (sec)	
				RCUR	RPCA
ExtYaleB	168×192	64	28	20.16	31.38

Figure 5. *Face modeling on ExtYaleB: Visual comparison of the outputs by RCUR and RPCA for face modeling task. The first row contains the original face images. The second and third rows are the face models and the facial occlusions outputted by RCUR, respectively. The last two rows are the face models and the facial occlusions outputted by RPCA, respectively.*

Appendix A. Uniform sampling and the proof of Theorem 3.5.

Here, we use a special case of the analysis of Tropp [48] to estimate the quantity β in Theorem 3.1.

Theorem A.1 ([48, Lemma 3.4]). *Suppose that $\boldsymbol{L} \in \mathbb{R}^{m\times n}$ has rank r and satisfies condition (A1) and that $\boldsymbol{V_L} \in \mathbb{R}^{n\times r}$ are its first r right singular vectors. Suppose that $J \subseteq [n]$ is chosen by sampling uniformly without replacement and that $|J| \geq \gamma\mu_2(\boldsymbol{L})r$ for some $\gamma > 0$. Then the quantity $\beta = \sqrt{|J|/n}\,\left\|(\boldsymbol{V_L}(J,:))^\dagger\right\|_2$ satisfies*

$$\beta \leq \frac{1}{\sqrt{1-\delta}} \quad \text{with probability at least} \quad 1 - r\left(\frac{e^{-\delta}}{(1-\delta)^{1-\delta}}\right)^{\gamma} \quad \text{for all} \quad \delta \in [0,1).$$

Proof of Theorem 3.5. Combine Theorems A.1 and 3.1. ∎

Appendix B. Error analysis and proof of Theorem 3.10. In this section we provide a combined error analysis of RPCA (in particular, AltProj [39]) and CUR approximations to obtain the main result concerning Algorithm 3.1, namely Theorem 3.10.

B.1. RPCA. Convergence has been widely studied for many RPCA algorithms [39]. Here, we list one for AltProj [39], which is one of the classic nonconvex RPCA algorithms. We emphasize that our results will stand with other RPCA convergence theories with slight variation.

Theorem B.1 ([39, Theorem 1]). *Let $\boldsymbol{L}$ and $\boldsymbol{S}$ satisfy assumptions* (A1) *and* (A2) *with $\alpha \le \mathcal{O}(\frac{1}{(\mu_1(\boldsymbol{L})\vee\mu_2(\boldsymbol{L}))r})$, respectively. With properly chosen parameters, the output of* AltProj*, $\boldsymbol{L}_k$, satisfies*

$$\|\boldsymbol{L}-\boldsymbol{L}_k\|_2 \le \varepsilon, \quad \|\boldsymbol{S}-\boldsymbol{S}_k\|_\infty \le \frac{\varepsilon}{\sqrt{mn}} \quad \text{and} \quad \operatorname{supp}(\boldsymbol{S}_k) \subseteq \operatorname{supp}(\boldsymbol{S})$$

in $k = \mathcal{O}(r\log(\frac{\|\boldsymbol{L}\|_2}{\varepsilon}))$ iterations.

Note the statement of Theorem B.1 in the original paper gives a Frobenius norm bound for $\boldsymbol{L}-\boldsymbol{L}_k$, but the version stated here follows easily from the fact that $\|\cdot\|_2 \le \|\cdot\|_F$.

B.2. CUR. Error analysis for CUR decompositions has been considered in various places, typically under assumptions on the column selection scheme. Below is a sample perturbation bound which is independent of the column selection procedure.

Theorem B.2 ([32, Remark 3.14]). *Let $\boldsymbol{L}$ have rank r and compact SVD $\boldsymbol{L} = \boldsymbol{W}\boldsymbol{\Sigma}\boldsymbol{V}^T$; let $\boldsymbol{C} = \boldsymbol{L}(:,J)$, $\boldsymbol{R} = \boldsymbol{L}(I,:)$, and $\boldsymbol{U} = \boldsymbol{L}(I,J)$ for some $I \subseteq [m]$, $J \subseteq [n]$. Let $\widehat{\boldsymbol{C}} \in \mathbb{R}^{m\times|J|}$ and $\widehat{\boldsymbol{R}} \in \mathbb{R}^{|I|\times n}$ be arbitrary with $\widehat{\boldsymbol{U}} = \widehat{\boldsymbol{C}}(I,:)$. For any Schatten p-norm, if $\sigma_r(\boldsymbol{U}) = \sigma_{\min}(\boldsymbol{U}) \ge 12\max\{\|\widehat{\boldsymbol{R}}-\boldsymbol{R}\|, \|\widehat{\boldsymbol{C}}-\boldsymbol{C}\|\}$, then the following holds:*

$$\begin{aligned}\text{(B.1)}\quad \left\|\boldsymbol{L}-\widehat{\boldsymbol{C}}\widehat{\boldsymbol{U}}_r^\dagger\widehat{\boldsymbol{R}}\right\| \le \Bigg(&\frac{7}{6}\left(\left\|\boldsymbol{W}_{\boldsymbol{L}}(I,:)^\dagger\right\| + \left\|\boldsymbol{V}_{\boldsymbol{L}}(J,:)^\dagger\right\|\right)\\ &+\frac{25}{6}\left\|\boldsymbol{W}_{\boldsymbol{L}}(I,:)^\dagger\right\|\left\|\boldsymbol{V}_{\boldsymbol{L}}(J,:)^\dagger\right\| + \frac{1}{6}\Bigg)\max\left\{\left\|\widehat{\boldsymbol{R}}-\boldsymbol{R}\right\|, \left\|\widehat{\boldsymbol{C}}-\boldsymbol{C}\right\|\right\}.\end{aligned}$$

Lemma B.3. *Suppose that $\boldsymbol{L}$ satisfies* (A1)*. Let $J \subseteq [n]$ with $|J| \ge \gamma_1\mu_2(\boldsymbol{L})r$ and $I \subseteq [m]$ with $|I| \ge \gamma_2\mu_1(\boldsymbol{L})r$ for some $\gamma_1, \gamma_2 > 0$ be chosen by sampling $[n]$ and $[m]$ uniformly without replacement respectively. Let $\boldsymbol{U} = \boldsymbol{L}(I,J)$. Then with probability at least $1 - r(\frac{e^{-\delta}}{(1-\delta)^{1-\delta}})^{\gamma_1} - r(\frac{e^{-\delta}}{(1-\delta)^{1-\delta}})^{\gamma_2}$,*

$$\text{(B.2)}\qquad \left\|\boldsymbol{U}^\dagger\right\|_2 \le \frac{1}{(1-\delta)\sigma_{\min}(\boldsymbol{L})}\sqrt{\frac{mn}{|I||J|}}.$$

Proof. The proof follows by similar argument to the proof of [32, Lemma 3.12]. ∎

Theorem B.4. *Given the notations and assumptions of Theorem 3.10, suppose that $\boldsymbol{C}$ and $\boldsymbol{R}$ satisfy conditions* (A1) *and* (A2) *with sparsity levels $\alpha \le \mathcal{O}(\frac{1}{(\mu_1(\boldsymbol{C})\vee\mu_2(\boldsymbol{C}))r})$ and $\mathcal{O}(\frac{1}{(\mu_1(\boldsymbol{R})\vee\mu_2(\boldsymbol{R})r})$, respectively. Suppose that* AltProj *is run for sufficiently many iterations*

in lines 5 and 6 of Algorithm 3.1 such that $\max\{\|\boldsymbol{C}-\widehat{\boldsymbol{C}}\|_2, \|\boldsymbol{R}-\widehat{\boldsymbol{R}}\|_2\} \le \varepsilon\frac{(1-\delta)\sigma_{\min}(\boldsymbol{L})}{12}\sqrt{\frac{|I||J|}{mn}}$. *Set* $\widehat{\boldsymbol{U}} = \widehat{\boldsymbol{C}}(I,:)$. *Then*

$$\left\|\boldsymbol{L} - \widehat{\boldsymbol{C}}\widehat{\boldsymbol{U}}^{\dagger}\widehat{\boldsymbol{R}}\right\|_2 \le \frac{7}{8}\varepsilon\sigma_{\min}(\boldsymbol{L}) \tag{B.3}$$

with probability at least $1 - \frac{r}{n^{c_2(\delta+(1-\delta)\log(1-\delta))}} - \frac{r}{m^{c_1(\delta-(1-\delta)\log(1-\delta))}}$.

Proof. Since I and J are chosen uniformly, by Lemma B.3 we have that with probability at least $1 - \frac{r}{n^{c_2(\delta+(1-\delta)\log(1-\delta))}} - \frac{r}{m^{c_1(\delta-(1-\delta)\log(1-\delta))}}$,

$$\left\|\boldsymbol{U}^{\dagger}\right\|_2 \le \frac{1}{(1-\delta)\sigma_{\min}(\boldsymbol{L})}\sqrt{\frac{mn}{|I||J|}}.$$

In addition, $\max\{\|\boldsymbol{C}-\widehat{\boldsymbol{C}}\|_2, \|\boldsymbol{R}-\widehat{\boldsymbol{R}}\|_2\} \le \varepsilon\frac{\sigma_{\min}(\boldsymbol{L})(1-\delta)}{12}\sqrt{\frac{|I||J|}{mn}}$ and $\widehat{\boldsymbol{U}} = \widehat{\boldsymbol{C}}(I,:)$. Combining these estimates and noting that $\|\boldsymbol{U}^{\dagger}\|_2 = \frac{1}{\sigma_{\min}(\boldsymbol{U})}$, we have

$$\sigma_r(\boldsymbol{U}) \ge (1-\delta)\sigma_{\min}(\boldsymbol{L})\sqrt{\frac{|I||J|}{mn}} > 12\varepsilon\frac{(1-\delta)\sigma_{\min}(\boldsymbol{L})}{12}\sqrt{\frac{|I||J|}{mn}} \ge 12\max\left\{\left\|\widehat{\boldsymbol{C}}-\boldsymbol{C}\right\|_2, \left\|\widehat{\boldsymbol{R}}-\boldsymbol{R}\right\|_2\right\}$$

with probability at least $1 - \frac{r}{n^{c_2(\delta+(1-\delta)\log(1-\delta))}} - \frac{r}{m^{c_1(\delta-(1-\delta)\log(1-\delta))}}$. The conclusion of the theorem follows from applying Theorem B.2 together with estimates of $\left\|\boldsymbol{W}_{\boldsymbol{L}}(I,:)^{\dagger}\right\|_2$ and $\left\|\boldsymbol{V}_{\boldsymbol{L}}(J,:)^{\dagger}\right\|_2$ by Theorem A.1, which implies that with the given success probability,

$$\left\|(\boldsymbol{V}_{\boldsymbol{L}}(J,:))^{\dagger}\right\|_2 \le \sqrt{\frac{n}{(1-\delta)|J|}},$$

and similarly for $\boldsymbol{W}_{\boldsymbol{L}}(I,:)$ by replacing n and $|J|$ by m and $|I|$, respectively. ∎

Proof of Theorem 3.10. Note that the first terms in the lower bound for $|I|$ and $|J|$ make the conclusions of Theorem 3.5 and Lemma B.3 valid, while the second terms ensure 2α-sparsity of $\boldsymbol{S}(I,:)$ and $\boldsymbol{S}(:,J)$ (Proposition 4.5). Consequently, $\boldsymbol{C} = \boldsymbol{L}(:,J)$ and $\boldsymbol{R} = \boldsymbol{L}(I,:)$ satisfy condition (A1) with the incoherence parameters in Theorem 3.1. Additionally, to apply Theorem B.1 to $\widetilde{\boldsymbol{C}} = \boldsymbol{D}(:,J)$ and $\widetilde{\boldsymbol{R}} = \boldsymbol{D}(I,:)$, we require that $2\alpha \le \mathcal{O}(\frac{1}{(\mu_1(\boldsymbol{C})\vee\mu_2(\boldsymbol{C}))r})$ and $2\alpha \le \mathcal{O}(\frac{1}{(\mu_1(\boldsymbol{R})\vee\mu_2(\boldsymbol{R}))r})$. Using the bounds of Theorem 3.5, which hold with high probability given how I and J were selected, we see that the assumption that $\alpha = \mathcal{O}(\frac{1-\delta}{\kappa(\boldsymbol{L})^2(\mu_1(\boldsymbol{L})\vee\mu_2(\boldsymbol{L}))})$ guarantees that α is of the required order and that, in turn, Theorem B.1 holds with the given success probability.

Finally, Theorem B.4 holds on account of the above observations and the fact that the iteration count being $\mathcal{O}(r\log(\sqrt{\frac{mn}{|I||J|}}\frac{\kappa(\boldsymbol{L})}{(1-\delta)\varepsilon}))$ implies the bounds on $\|\boldsymbol{C}-\widehat{\boldsymbol{C}}\|_2$ and $\|\boldsymbol{R}-\widehat{\boldsymbol{R}}\|_2$. Hence, we conclude that the outputs $\widehat{\boldsymbol{C}}, \widehat{\boldsymbol{R}}$ of Algorithm 3.1 satisfy

$$\left\|\boldsymbol{L}-\widehat{\boldsymbol{L}}\right\|_2 = \left\|\boldsymbol{L}-\widehat{\boldsymbol{C}}(\widehat{\boldsymbol{C}}(I,:))^{\dagger}\widehat{\boldsymbol{R}}\right\|_F \le \frac{7}{8}\varepsilon\sigma_{\min}(\boldsymbol{L})$$

with the given success probability, which completes the proof. ∎

Appendix C. Uniform + deterministic: Proof of Theorem 3.11. The deterministic greedy column selection algorithm of [3] is reproduced as Algorithm C.1. Here, we provide the proof of Theorem 3.11, beginning with a bound on the pseudoinverse of $\boldsymbol{V_L}(:, J)$, assuming that the column submatrix of $\boldsymbol{L}$ and $\boldsymbol{D}$ are not too far apart. We will find use for this result later as well when we analyze the purely deterministic column sampling method.

Recall that Theorem 3.11 analyzes the error of approximation for the hybrid procedure of uniformly sampling column and row submatrices of $\boldsymbol{D} = \boldsymbol{L} + \boldsymbol{S}$, running RPCA on these to output $\widehat{\boldsymbol{C}}$ and $\widehat{\boldsymbol{R}}$, and finally using Algorithm C.1 to select exactly $r = \operatorname{rank}(\boldsymbol{L})$ columns and rows of $\widehat{\boldsymbol{C}}$ and $\widehat{\boldsymbol{R}}$, respectively, to give a compact approximation of $\boldsymbol{L}$.

Theorem C.1. *Let $\boldsymbol{L} \in \mathbb{R}^{m\times n}$ with $\operatorname{rank}(\boldsymbol{L}) = r$. Let $\widetilde{\boldsymbol{L}} = \boldsymbol{L} + \boldsymbol{E}$ with compact SVD $\widetilde{\boldsymbol{L}} = \widetilde{\boldsymbol{W}}\widetilde{\boldsymbol{\Sigma}}\widetilde{\boldsymbol{V}}^T$, and suppose $\|\boldsymbol{E}\|_2 \le \frac{1}{2}\sigma_{\min}(\boldsymbol{L})$. Suppose that $I \subseteq [m]$ and $J \subseteq [n]$ with $|I| = |J| = r$ are obtained by applying Algorithm C.1 on $\widetilde{\boldsymbol{W}}$ and $\widetilde{\boldsymbol{V}}$. Suppose that*

$$\left\|\boldsymbol{L}(:, J) - \widetilde{\boldsymbol{L}}(:, J)\right\|_2 = \left\|\boldsymbol{W_L}\boldsymbol{\Sigma_L}(\boldsymbol{V_L}(J, :))^T - \widetilde{\boldsymbol{W}}\widetilde{\boldsymbol{\Sigma}}(\widetilde{\boldsymbol{V}}(J, :))^T\right\|_2 \le \frac{1}{8}\sqrt{\frac{|J|}{rn}}\sigma_{\min}(\boldsymbol{L}).$$

Then $\|(\boldsymbol{V_L}(J, :))^\dagger\|_2 \le 4\kappa(\boldsymbol{L})\sqrt{\frac{rn}{|J|}}$.

Proof. Set $\widetilde{\boldsymbol{C}} = \widetilde{\boldsymbol{W}}\widetilde{\boldsymbol{\Sigma}}(\widetilde{\boldsymbol{V}}(J, :))^T$ and $\boldsymbol{C} = \boldsymbol{L}(:, J) = \boldsymbol{W_L}\boldsymbol{\Sigma_L}(\boldsymbol{V_L}(J, :))^T$. Then we have

$$\left\|(\boldsymbol{V_L}(J, :))^\dagger\right\|_2 = \left\|\boldsymbol{C}^\dagger\boldsymbol{W_L}\boldsymbol{\Sigma_L}\right\|_2 \le \sigma_{\max}(\boldsymbol{L})\left\|\boldsymbol{C}^\dagger\right\|_2. \tag{C.1}$$

Notice that $\operatorname{rank}(\boldsymbol{C}) = \operatorname{rank}(\widetilde{\boldsymbol{C}}) = r$. By [45, Theorems 3.1–3.4], we have

$$\left\|\boldsymbol{C}^\dagger - \widetilde{\boldsymbol{C}}^\dagger\right\|_2 \le 2\left\|\boldsymbol{C}^\dagger\right\|_2\left\|\widetilde{\boldsymbol{C}}^\dagger\right\|_2\left\|\boldsymbol{C} - \widetilde{\boldsymbol{C}}\right\|_2.$$

Algorithm C.1. A deterministic greedy removal algorithm for subset selection [3]

Input: $\boldsymbol{X} \in \mathbb{R}^{r\times m}$ with $\operatorname{rank}(\boldsymbol{X}) = r$; k: sampling size.
Output: $\mathcal{S} \subseteq [m]$: set of cardinality k.
Initialization: $\mathcal{S}_0 := [m]$
Compute the SVD of $\boldsymbol{X}_{\mathcal{S}_0}$: $\boldsymbol{X}_{\mathcal{S}_0} = \boldsymbol{U}^{(0)}\boldsymbol{\Sigma}^{(0)}\boldsymbol{Y}^{(0)}$
for $i = 1, 2, \ldots, m - k$ **do**
  Let the singular values of $\boldsymbol{X}_{\mathcal{S}_{i-1}}$ be $\sigma_1^{(i-1)}, \ldots, \sigma_r^{(i-1)}$.
  Let the columns of $\boldsymbol{Y}^{(i-1)}$ be $\{y_k^{(i-1)}\}_{k\in\mathcal{S}_{i-1}}$. Denote by y_{kj} the jth element of $y_k^{(i-1)}$.
  $k_i := \underset{k\in\mathcal{S}_{i-1},\left\|y_k^{(i-1)}\right\|_2<1}{\operatorname{argmin}} \left(\frac{\sum_{j=1}^r (y_{kj}^{(i-1)}/\sigma_k^{(i-1)})^2}{1-\left\|y_k^{(i-1)}\right\|_2^2}\right)$.
  Set $\mathcal{S}_i := \mathcal{S}_{i-1} \setminus \{k_i\}$
  Downdate the SVD of $\boldsymbol{X}_{\mathcal{S}_{i-1}}$ to obtain an SVD of $\boldsymbol{X}_{\mathcal{S}_i} = \boldsymbol{U}^{(i)}\boldsymbol{\Sigma}^{(i)}\boldsymbol{Y}^{(i)}$ (see [30])
end for

In addition, by Proposition 3.3,

$$\begin{aligned}\left\|\widetilde{\boldsymbol{C}}^{\dagger}\right\|_2 &= \left\|\left((\widetilde{\boldsymbol{V}}(J,:))^T\right)^{\dagger}\widetilde{\boldsymbol{\Sigma}}^{\dagger}\widetilde{\boldsymbol{W}}^T\right\|_2 \\ &\le \left\|\left(\widetilde{\boldsymbol{V}}(J,:)\right)^{\dagger}\right\|_2\left\|\widetilde{\boldsymbol{\Sigma}}^{\dagger}\right\|_2 \\ &\le \sqrt{\frac{rn}{|J|}}\frac{1}{\sigma_{\min}(\widetilde{\boldsymbol{L}})} \\ &\le \sqrt{\frac{rn}{|J|}}\frac{1}{\sigma_{\min}(\boldsymbol{L})-\|\boldsymbol{E}\|_2} \\ &\le 2\sqrt{\frac{rn}{|J|}}\frac{1}{\sigma_{\min}(\boldsymbol{L})}. \end{aligned} \tag{C.2}$$

Since $\|\boldsymbol{W_L}\boldsymbol{\Sigma_L}(\boldsymbol{V_L}(J,:))^T-\widetilde{\boldsymbol{W}}\widetilde{\boldsymbol{\Sigma}}(\widetilde{\boldsymbol{V}}(J,:))^T\|_2 \le \frac{1}{8}\sqrt{\frac{|J|}{rn}}\sigma_{\min}(\boldsymbol{L})$, $\|\boldsymbol{C}^{\dagger}\|_2 \le \frac{\|\widetilde{\boldsymbol{C}}^{\dagger}\|_2}{1-2\|\widetilde{\boldsymbol{C}}^{\dagger}\|_2\|\boldsymbol{C}-\widetilde{\boldsymbol{C}}\|_2} \le 2\|\widetilde{\boldsymbol{C}}^{\dagger}\|_2$. Combining, (C.1) and (C.2), we have

$$\left\|\boldsymbol{V_L}(J,:)^{\dagger}\right\| \le \sqrt{\frac{rn}{|J|}}\frac{4\sigma_{\max}(\boldsymbol{L})}{\sigma_{\min}(\boldsymbol{L})} = 4\kappa(\boldsymbol{L})\sqrt{\frac{rn}{|J|}}. \tag{C.3}$$

■

Proof of Theorem 3.11. The assumptions of Theorem 3.10 imply that $\widetilde{\boldsymbol{C}} = \boldsymbol{D}(:,J) = \boldsymbol{L}(:,J)+\boldsymbol{S}(:,J)$ and $\widetilde{\boldsymbol{R}} = \boldsymbol{D}(I,:) = \boldsymbol{L}(I,:)+\boldsymbol{S}(I,:)$ satisfy the necessary incoherence and sparsity conditions for RPCA to be successful. Thus we pick up most of the way through the proof of Theorem 3.10 above and note that the new assumption on the iteration count for AltProj in the statement of this theorem implies by Theorem B.1 that we have

$$\max\left\{\left\|\boldsymbol{C}-\widehat{\boldsymbol{C}}\right\|_2, \left\|\boldsymbol{R}-\widehat{\boldsymbol{R}}\right\|_2\right\} \le \frac{\varepsilon(1-\delta)^2\sigma_{\min}(\boldsymbol{L})}{229\kappa(\boldsymbol{L})^4 r\sqrt{mn\mu_1(\boldsymbol{L})\mu_2(\boldsymbol{L})}}. \tag{C.4}$$

According to Theorem A.1, the following statements hold:

$$\begin{aligned}\left\|\boldsymbol{W_L}(I,:)^{\dagger}\right\|_2 &\le \sqrt{\frac{m}{(1-\delta)|I|}}, \\ \left\|\boldsymbol{V_L}(J,:)^{\dagger}\right\|_2 &\le \sqrt{\frac{n}{(1-\delta)|J|}}\end{aligned} \tag{C.5}$$

with probability at least $1-\frac{r}{n^{c_2(\delta+(1-\delta)\log(1-\delta))}}-\frac{r}{m^{c_1(\delta-(1-\delta)\log(1-\delta))}}$. From (C.4), we have

$$\begin{aligned}\left\|\widehat{\boldsymbol{C}}_1-\boldsymbol{C}(:,J_1)\right\|_2 &\le \frac{\varepsilon(1-\delta)^2\sigma_{\min}(\boldsymbol{L})}{229\kappa(\boldsymbol{L})^4 r\sqrt{mn\mu_1(\boldsymbol{L})\mu_2(\boldsymbol{L})}}, \\ \left\|\widehat{\boldsymbol{R}}_1-\boldsymbol{R}(I_1,:)\right\|_2 &\le \frac{\varepsilon(1-\delta)^2\sigma_{\min}(\boldsymbol{L})}{229\kappa(\boldsymbol{L})^4 r\sqrt{mn\mu_1(\boldsymbol{L})\mu_2(\boldsymbol{L})}}.\end{aligned}$$

In addition,

$$\frac{1}{8}\sqrt{\frac{|J_1|}{r|J|}}\sigma_{\min}(\boldsymbol{C}) = \frac{1}{8\sqrt{|J|}}\frac{1}{\|\boldsymbol{C}^\dagger\|_2} \geq \frac{1}{8\sqrt{|J|}}\frac{1}{\|\boldsymbol{L}^\dagger\|_2\left\|(\boldsymbol{V}_{\boldsymbol{L}}(J,:))^\dagger\right\|_2} \geq \frac{\sqrt{1-\delta}\sigma_{\min}(\boldsymbol{L})}{8\sqrt{n}}.$$

Thus, $\|\widehat{\boldsymbol{C}}_1 - \boldsymbol{C}(:,J_1)\|_2 \leq \frac{1}{8}\sqrt{\frac{|J_1|}{r|J|}}\sigma_{\min}(\boldsymbol{C})$. Similarly, we have $\|\widehat{\boldsymbol{R}}_1 - \boldsymbol{R}(I_1,:)\|_2 \leq \frac{1}{8}\sqrt{\frac{|I_1|}{r|I|}}\sigma_{\min}(\boldsymbol{R})$. By Theorem C.1,

$$\left\|\boldsymbol{V}_{\boldsymbol{C}}(J_1,:)^\dagger\right\| \leq 4\kappa(\boldsymbol{C})\sqrt{|J|},$$
$$\left\|\boldsymbol{W}_{\boldsymbol{R}}(I_1,:)^\dagger\right\| \leq 4\kappa(\boldsymbol{R})\sqrt{|I|}.$$

Since $\boldsymbol{C}(:,J_1) = \boldsymbol{W}_{\boldsymbol{C}}\boldsymbol{\Sigma}_{\boldsymbol{C}}(\boldsymbol{V}_{\boldsymbol{C}}(J_1,:))^T = \boldsymbol{W}_{\boldsymbol{L}}\boldsymbol{\Sigma}_{\boldsymbol{L}}(\boldsymbol{V}_{\boldsymbol{L}}(\tilde{J}_1,:))^T$, we have

$$\begin{aligned}
\left\|\boldsymbol{V}_{\boldsymbol{L}}(\tilde{J}_1,:)^\dagger\right\|_2 &\leq \left\|\boldsymbol{V}_{\boldsymbol{C}}(J_1,:)^\dagger\right\|_2 \|\boldsymbol{\Sigma}_{\boldsymbol{L}}\|_2 \left\|\boldsymbol{\Sigma}_{\boldsymbol{C}}^{-1}\right\|_2 \\
&\overset{\text{Lemma 4.3}}{=} 4\kappa(\boldsymbol{C})\sqrt{|J|}\,\|\boldsymbol{L}\|_2\left\|\boldsymbol{L}^\dagger\right\|_2\left\|\boldsymbol{V}_{\boldsymbol{L}}(J,:)^\dagger\right\|_2 \\
&\overset{\text{Lemma 4.4}}{\leq} 4\kappa(\boldsymbol{L})^2\frac{|J|\sqrt{\mu_2 r}}{\sqrt{n}}\left\|\boldsymbol{V}_{\boldsymbol{L}}(J,:)^\dagger\right\|_2^2 \\
&\overset{\text{(C.5)}}{\leq} 4\kappa(\boldsymbol{L})^2\frac{\sqrt{\mu_2(\boldsymbol{L})rn}}{1-\delta}.
\end{aligned}$$

Similarly, we have

$$\left\|\boldsymbol{W}_{\boldsymbol{L}}(I_1,:)^\dagger\right\| \leq 4\kappa(\boldsymbol{L})^2\frac{\sqrt{\mu_1(\boldsymbol{L})rm}}{1-\delta}.$$

Thus,

$$\left\|\boldsymbol{U}^\dagger\right\|_2 \leq \frac{16\kappa^4(\boldsymbol{L})r\sqrt{\mu_1(\boldsymbol{L})\mu_2(\boldsymbol{L})mn}}{(1-\delta)^2\sigma_{\min}(\boldsymbol{L})},$$

i.e., $\sigma_r(\boldsymbol{U}) \geq \frac{(1-\delta)^2\sigma_{\min}(\boldsymbol{L})}{16\kappa^4(\boldsymbol{L})r\sqrt{\mu_1(\boldsymbol{L})\mu_2(\boldsymbol{L})mn}} \geq 12\max\{\|\widehat{\boldsymbol{C}}_1 - \boldsymbol{L}(:,\tilde{J}_1)\|_2, \|\widehat{\boldsymbol{R}}_1 - \boldsymbol{L}(\tilde{I}_1,:)\|_2\}$. Using the bounds of $\|(\boldsymbol{W}(I,:))^\dagger\|_2$, $\|(\boldsymbol{V}(J,:))^\dagger\|_2$ and Theorem B.2, we obtain

$$\left\|\boldsymbol{L} - \widehat{\boldsymbol{C}}_1\widehat{\boldsymbol{U}}_1^\dagger\widehat{\boldsymbol{R}}_1\right\| \leq \frac{1}{3}\varepsilon\sigma_r(\boldsymbol{L})$$

after some simple calculations. ∎

Appendix D. Deterministic column and row selection. In this section, we study what happens when we apply the greedy algorithm (Algorithm C.1) on the singular vectors of $\boldsymbol{D} = \boldsymbol{L}+\boldsymbol{S}$ to choose column and row submatrices. We primarily consider how the incoherence

Algorithm D.1. Initialization by one step of alternating projections

Input: $\boldsymbol{D} = \boldsymbol{L} + \boldsymbol{S}$: matrix to be split; r: rank of $\boldsymbol{L}$; η: thresholding parameter.
Output: $\boldsymbol{L}_0, \boldsymbol{S}_0$
$\zeta_0 = \eta \cdot \sigma_1(\boldsymbol{D})$
$\boldsymbol{S}_0 = \mathcal{T}_{\zeta_0}(\boldsymbol{D})$ ▷ $\mathcal{T}_\zeta$: hard thresholding operation with thresholding value ζ
$\boldsymbol{L}_0 = \mathcal{D}_r(\boldsymbol{D} - \boldsymbol{S}_0)$ ▷ $\mathcal{D}_r$: truncated rank r SVD operator

of $\boldsymbol{C} = \boldsymbol{L}(:, J)$ relates to that of $\boldsymbol{L}$. Our main result is that, if we first initialize $\boldsymbol{D}$ by one step of alternating projections [39] with a judiciously chosen hard thresholding parameter, then the resulting approximation $\boldsymbol{L}_0$ to $\boldsymbol{L}$ is good enough to ensure that the incoherence does not inflate significantly. To achieve this, we study conditions under which the hypothesis of Theorem B.4 holds, namely such that $\|\boldsymbol{L}(:, J) - \boldsymbol{L}_0(:, J)\|_2 \leq \frac{1}{8}\sqrt{\frac{|J|}{rn}}\sigma_{\min}(\boldsymbol{L})$, where $\boldsymbol{L}_0$ is the output of Algorithm D.1. To proceed, we first state the initialization required.

Here, $\mathcal{T}_{\zeta_0}(\boldsymbol{D})$ is the hard thresholding operator whose output is (entrywise) $\boldsymbol{D}_{ij}$ if $\boldsymbol{D}_{ij} \geq \zeta_0$ and 0 otherwise. Given a matrix $\boldsymbol{A}$, $\mathcal{D}_r(\boldsymbol{A})$ is its projection onto the set of rank r matrices (e.g., $\mathcal{D}_r(\boldsymbol{A}) = \boldsymbol{U}_{\boldsymbol{A},r}\boldsymbol{\Sigma}_{\boldsymbol{A},r}\boldsymbol{V}_{\boldsymbol{A},r}^T$).

Theorem D.1. *Let $\boldsymbol{L} \in \mathbb{R}^{m\times n}$ be a rank r $\{\mu_1(\boldsymbol{L}), \mu_2(\boldsymbol{L})\}$-incoherent matrix and $\boldsymbol{S} \in \mathbb{R}^{m\times n}$ be an α-sparse matrix. Let $\mu := \max\{\mu_1(\boldsymbol{L}), \mu_2(\boldsymbol{L})\}$. If the sparsity level satisfies $\alpha \leq \frac{1}{256\mu^{3/2}r^2\kappa(\boldsymbol{L})}$ and the thresholding parameter in Algorithm D.1 obeys $\frac{\mu r\sigma_{\max}(\boldsymbol{L})}{\sqrt{mn}\sigma_{\max}(\boldsymbol{D})} \leq \eta \leq \frac{3\mu r\sigma_{\max}(\boldsymbol{L})}{\sqrt{mn}\sigma_{\max}(\boldsymbol{D})}$, then the outputs of Algorithm D.1 satisfy*

$$\begin{aligned}
\|\boldsymbol{L} - \boldsymbol{L}_0\|_2 &\leq 8\alpha\mu r\sigma_{\max}(\boldsymbol{L}),\\
\max_i \left\|(\boldsymbol{L}_0 - \boldsymbol{L})^T\boldsymbol{e}_i\right\|_2 &\leq \frac{32\alpha\mu^{1.5}r^{1.5}}{\sqrt{m}}\sigma_{\max}(\boldsymbol{L}),\\
\max_j \|(\boldsymbol{L}_0 - \boldsymbol{L})\boldsymbol{e}_j\|_2 &\leq \frac{32\alpha\mu^{1.5}r^{1.5}}{\sqrt{n}}\sigma_{\max}(\boldsymbol{L}),\\
\|\boldsymbol{S} - \boldsymbol{S}_0\|_\infty &\leq \frac{\mu r}{n}\sigma_{\max}(\boldsymbol{L}), \quad \text{and} \quad \operatorname{supp}(\boldsymbol{S}_0) \subseteq \operatorname{supp}(\boldsymbol{S}).
\end{aligned}$$

Note that since $\|\boldsymbol{A}(I,:)\|_F^2 \leq \sum_{i\in I}\left\|\boldsymbol{A}^T\boldsymbol{e}_i\right\|_2^2$ and $\|\boldsymbol{A}(:,J)\|_F^2 \leq \sum_{j\in J}\|\boldsymbol{A}\boldsymbol{e}_j\|_2^2$, Theorem D.1 implies

$$\begin{aligned}
\|\boldsymbol{L}(I,:) - \boldsymbol{L}_0(I,:)\|_F &\leq \frac{32\alpha\mu^{1.5}r^{1.5}\sqrt{|I|}}{\sqrt{m}}\sigma_{\max}(\boldsymbol{L}) \leq \frac{1}{8}\sqrt{\frac{|I|}{rm}}\sigma_{\min}(\boldsymbol{L}),\\
\|\boldsymbol{L}(:,J) - \boldsymbol{L}_0(:,J)\|_F &\leq \frac{32\alpha\mu^{1.5}r^{1.5}\sqrt{|J|}}{\sqrt{n}}\sigma_{\max}(\boldsymbol{L}) \leq \frac{1}{8}\sqrt{\frac{|J|}{rn}}\sigma_{\min}(\boldsymbol{L}),\\
\|\boldsymbol{L} - \boldsymbol{L}_0\|_F &\leq \sqrt{2r}\,\|\boldsymbol{L} - \boldsymbol{L}_0\|_2 \leq \frac{1}{2}\sigma_{\min}(\boldsymbol{L}),
\end{aligned}$$

where the second parts of the inequalities follow from bound of α.

Remark D.2. Note that the conclusion of Theorem D.1 implies that $\boldsymbol{L}_0$ satisfies the necessary requirement for Theorem C.1. Consequently, we find that the output of Algorithm C.1 applied to $\boldsymbol{L}_0$ yields a column submatrix $\boldsymbol{C} = \boldsymbol{L}(:, J)$ with parameter $\beta \le 4\kappa(\boldsymbol{L})\sqrt{r}$, and thus incoherence $\mu_2(\boldsymbol{C}) \le 16\kappa(\boldsymbol{L})^2 r$ according to Theorem 3.1. This bound for β is a factor of $\sqrt{r}$ better than simply applying the greedy algorithm directly to $\boldsymbol{V}_{\boldsymbol{L}}$ due to good initialization to find $\boldsymbol{L}_0$. This method also has the benefit of being tractable as it does not require knowledge of $\boldsymbol{V}_{\boldsymbol{L}}$.

Appendix E. Proof of Theorem D.1.

As shown in prior art [10, 12, 11, 39], the convergence analysis of the alternating projections-based RPCA algorithms can be reduced to the case of symmetric matrices, since nonsymmetric matrix recovery problems can be cast as problems with respect to symmetric augmented matrices. For more details about how to reduce the general RPCA problems to symmetric cases, we refer the interested reader to [10, section 3.2]. Firstly, we shall present two technical lemmata in the symmetric setting.

Lemma E.1 ([39, Lemma 4]). *Let $\boldsymbol{S} \in \mathbb{R}^{n\times n}$ be an α-sparse symmetric matrix. Then, the following inequality holds:*

$$\|\boldsymbol{S}\|_2 \le \alpha n \|\boldsymbol{S}\|_\infty .$$

Lemma E.2 ([39, Lemma 5]). *Let $\boldsymbol{S} \in \mathbb{R}^{n\times n}$ be an α-sparse symmetric matrix. Let $\boldsymbol{W} \in \mathbb{R}^{n\times r}$ be an orthogonal matrix with μ-incoherence, i.e., $\|\boldsymbol{W}^T\boldsymbol{e}_i\|_2 \le \sqrt{\frac{\mu r}{n}}$ for all i. Then*

$$\|\boldsymbol{W}^T\boldsymbol{S}^a\boldsymbol{e}_i\|_2 \le \sqrt{\frac{\mu r}{n}}\,(\alpha n \|\boldsymbol{S}\|_\infty)^a$$

holds for all i and $a \ge 0$.

We are now ready to prove Theorem D.1.

Proof of Theorem D.1. As discussed above, we only need to prove the theorem under symmetric setting, and it can be generalized to nonsymmetric matrices. The proof consists of four parts. The first three parts have been shown in the proof of [12, Theorem 2], but we still include them here for the completeness.

Part 1. First, note that

$$\|\boldsymbol{L}\|_\infty = \max_{ij} |\boldsymbol{e}_i^T\boldsymbol{W}_{\boldsymbol{L}}\boldsymbol{\Sigma}_{\boldsymbol{L}}\boldsymbol{V}_{\boldsymbol{L}}^T\boldsymbol{e}_j| \le \max_{ij} \|\boldsymbol{W}_{\boldsymbol{L}}^T\boldsymbol{e}_i\|_2 \|\boldsymbol{\Sigma}_{\boldsymbol{L}}\|_2 \|\boldsymbol{V}_{\boldsymbol{L}}^T\boldsymbol{e}_j\|_2 \le \frac{\mu r}{n}\sigma_1(\boldsymbol{L}),$$

where the last inequality follows from μ-incoherence of $\boldsymbol{L}$. Since $\eta \ge \frac{\mu r \sigma_1(\boldsymbol{L})}{n\sigma_1(\boldsymbol{D})}$, we get

$$\|\boldsymbol{L}\|_\infty \le \eta\sigma_1(\boldsymbol{D}) =: \zeta_0. \tag{E.1}$$

Considering the entries of $\boldsymbol{S}_0$, we notice

$$[\boldsymbol{S}_0]_{ij} = [\mathcal{T}_{\zeta_0}(\boldsymbol{S}+\boldsymbol{L})]_{ij} = \begin{cases} \mathcal{T}_{\zeta_0}([\boldsymbol{S}+\boldsymbol{L}]_{ij}), & (i,j) \in \operatorname{supp}(\boldsymbol{S}), \\ \mathcal{T}_{\zeta_0}([\boldsymbol{L}]_{ij}), & (i,j) \notin \operatorname{supp}(\boldsymbol{S}), \end{cases}$$

since $[\boldsymbol{S}]_{ij} = 0$ outside of its support. Together with (E.1), we have $[\boldsymbol{S}_0]_{ij} = 0$ for all $(i,j) \notin \operatorname{supp}(\boldsymbol{S})$, which implies $\operatorname{supp}(\boldsymbol{S}_0) \subseteq \operatorname{supp}(\boldsymbol{S})$. Furthermore, we have

$$[\boldsymbol{S} - \boldsymbol{S}_0]_{ij} = \begin{cases} 0, \\ [\boldsymbol{L}]_{ij}, \\ [\boldsymbol{S}]_{ij} \end{cases} \leq \begin{cases} 0, \\ \|\boldsymbol{L}\|_\infty, \\ \|\boldsymbol{L}\|_\infty + \zeta_0 \end{cases} \leq \begin{cases} 0, & (i,j) \notin \operatorname{supp}(\boldsymbol{S}), \\ \frac{\mu r}{n}\sigma_1(\boldsymbol{L}), & (i,j) \in \operatorname{supp}(\boldsymbol{S}_0), \\ \frac{4\mu r}{n}\sigma_1(\boldsymbol{L}), & (i,j) \in \operatorname{supp}(\boldsymbol{S}) \setminus \operatorname{supp}(\boldsymbol{S}_0), \end{cases}$$

where the last inequality follows from $\eta \leq \frac{3\mu r \sigma_1(\boldsymbol{L})}{n\sigma_1(\boldsymbol{D})}$, which implies $\zeta_0 \leq \frac{3\mu r}{n}\sigma_1(\boldsymbol{L})$. Overall, we get

$$\operatorname{supp}(\boldsymbol{S}_0) \subseteq \operatorname{supp}(\boldsymbol{S}) \quad \text{and} \quad \|\boldsymbol{S} - \boldsymbol{S}_0\|_\infty \leq \frac{4\mu r}{n}\sigma_1(\boldsymbol{L}). \tag{E.2}$$

Moreover, this implies that $\boldsymbol{S} - \boldsymbol{S}_0$ is also α-sparse.

Part 2. Since $\boldsymbol{L}_0 = \mathcal{D}_r(\boldsymbol{S} - \boldsymbol{S}_0)$ is the best rank r approximation of $\boldsymbol{D} - \boldsymbol{S}_0$, we have

$$\begin{aligned} \|\boldsymbol{L} - \boldsymbol{L}_0\|_2 &\leq \|\boldsymbol{L} - (\boldsymbol{D} - \boldsymbol{S}_0)\|_2 + \|(\boldsymbol{D} - \boldsymbol{S}_0) - \boldsymbol{L}_0\|_2 \\ &\leq 2\|\boldsymbol{L} - (\boldsymbol{D} - \boldsymbol{S}_0)\|_2 \\ &= 2\|\boldsymbol{L} - (\boldsymbol{L} + \boldsymbol{S} - \boldsymbol{S}_0)\|_2 \\ &= 2\|\boldsymbol{S} - \boldsymbol{S}_0\|_2 \\ &\leq 2\alpha n \|\boldsymbol{S} - \boldsymbol{S}_0\|_\infty, \end{aligned}$$

where the last inequality uses Lemma E.1. By applying (E.2), we have

$$\|\boldsymbol{L} - \boldsymbol{L}_0\|_2 \leq 8\alpha\mu r \sigma_1(\boldsymbol{L}). \tag{E.3}$$

Part 3. Let λ_i denote the *ith* eigenvalue of $\boldsymbol{D} - \boldsymbol{S}_0$ ordered as $|\lambda_1| \geq |\lambda_2| \geq \cdots \geq |\lambda_n|$. Since $\boldsymbol{D} - \boldsymbol{S}_0 = \boldsymbol{L} + (\boldsymbol{S} - \boldsymbol{S}_0)$, by Weyl's inequality, we have

$$\big|\sigma_i(\boldsymbol{L}) - |\lambda_i|\big| \leq \|\boldsymbol{S} - \boldsymbol{S}_0\|_2 \leq \alpha n \|\boldsymbol{S} - \boldsymbol{S}_0\|_\infty \leq \frac{\sigma_r(\boldsymbol{L})}{8} \tag{E.4}$$

for all i, where the last inequality follows from the fact that $\alpha \leq \frac{1}{32\mu r \kappa(\boldsymbol{L})}$. Therefore, we have

$$\frac{7}{8}\sigma_i(\boldsymbol{L}) \leq |\lambda_i| \leq \frac{9}{8}\sigma_i(\boldsymbol{L}), \qquad 1 \leq i \leq r, \tag{E.5}$$

and

$$\frac{\|\boldsymbol{S} - \boldsymbol{S}_0\|_2}{|\lambda_r|} \leq \frac{\frac{\sigma_r(\boldsymbol{L})}{8}}{\frac{7\sigma_r(\boldsymbol{L})}{8}} = \frac{1}{7}. \tag{E.6}$$

Part 4. Let $\boldsymbol{D} - \boldsymbol{S}_0 = [\boldsymbol{W}_0, \ddot{\boldsymbol{W}}_0]\left[\begin{smallmatrix} \boldsymbol{\Lambda} & \boldsymbol{0} \\ \boldsymbol{0} & \ddot{\boldsymbol{\Lambda}} \end{smallmatrix}\right][\boldsymbol{W}_0, \ddot{\boldsymbol{W}}_0]^T = \boldsymbol{W}_0\boldsymbol{\Lambda}\boldsymbol{W}_0^T + \ddot{\boldsymbol{W}}_0\ddot{\boldsymbol{\Lambda}}\ddot{\boldsymbol{W}}_0^T$ be the eigenvalue decomposition of $\boldsymbol{D} - \boldsymbol{S}_0$, where the eigenvalues are sorted by magnitude and $\boldsymbol{\Lambda}$ consists of the r largest ones while $\ddot{\boldsymbol{\Lambda}}$ contains the rest. Correspondingly, $\boldsymbol{W}_0$ contains the first r eigenvectors,

and $\ddot{\boldsymbol{W}}_0$ has the rest. By the symmetric setting, we have $\boldsymbol{L}_0 = \mathcal{D}_r(\boldsymbol{D} - \boldsymbol{S}_0) = \boldsymbol{W}_0\boldsymbol{\Lambda}\boldsymbol{W}_0^T$. Denote $\boldsymbol{E} = \boldsymbol{S} - \boldsymbol{S}_0$ and $\boldsymbol{W}(:,i)$ to be the ith eigenvector of $\boldsymbol{D} - \boldsymbol{S}_0 = \boldsymbol{L} + \boldsymbol{E}$. So we can see

$$(\boldsymbol{L} + \boldsymbol{E})\boldsymbol{W}(:,i) = \lambda_i \boldsymbol{W}(:,i)$$

for $1 \le i \le r$. Hence,

$$\boldsymbol{W}(:,i) = \left(I - \frac{\boldsymbol{E}}{\lambda_i}\right)^{-1} \frac{\boldsymbol{L}}{\lambda_i}\boldsymbol{W}(:,i) = \sum_{j=0}^{\infty} \left(\frac{\boldsymbol{E}}{\lambda_i}\right)^j \frac{\boldsymbol{L}}{\lambda_i}\boldsymbol{W}(:,i) \tag{E.7}$$

for all $1 \le i \le r$. Note that the expansion in the last equality is valid since (E.6) implies $\frac{\|\boldsymbol{E}\|_2}{|\lambda_i|} < \frac{1}{7} \le 1$ for all $1 \le i \le r$.

We will first prove the *row version* of the inequality. By applying (E.7), we get

$$\begin{aligned}
&\max_i \left\|(\boldsymbol{L}_0 - \boldsymbol{L})^T \boldsymbol{e}_i\right\|_2 \\
&\quad = \max_i \left\|(\boldsymbol{W}_0\boldsymbol{\Lambda}\boldsymbol{W}_0^T - \boldsymbol{L})^T \boldsymbol{e}_i\right\|_2 \\
&\quad = \max_i \left\| \left(\boldsymbol{L}\boldsymbol{W}_0\boldsymbol{\Lambda}^{-1}\boldsymbol{W}_0^T\boldsymbol{L} - \boldsymbol{L} + \sum_{a+b>0} \boldsymbol{E}^a\boldsymbol{L}\boldsymbol{W}_0\boldsymbol{\Lambda}^{-(a+b+1)}\boldsymbol{W}_0^T\boldsymbol{L}\boldsymbol{E}^b\right)^T \boldsymbol{e}_i \right\|_2 \\
&\quad \le \max_i \left\|(\boldsymbol{L}\boldsymbol{W}_0\boldsymbol{\Lambda}^{-1}\boldsymbol{W}_0^T\boldsymbol{L} - \boldsymbol{L})^T \boldsymbol{e}_i\right\|_2 + \sum_{a+b>0} \max_i \left\| \left(\boldsymbol{E}^a\boldsymbol{L}\boldsymbol{W}_0\boldsymbol{\Lambda}^{-(a+b+1)}\boldsymbol{W}_0^T\boldsymbol{L}\boldsymbol{E}^b\right)^T \boldsymbol{e}_i \right\|_2 \\
&\quad := \boldsymbol{Y}_0 + \sum_{a+b>0} \boldsymbol{Y}_{ab}.
\end{aligned}$$

We will first bound $\boldsymbol{Y}_0$. By the incoherence property of $\boldsymbol{L}$, i.e., $\max_i \|\boldsymbol{W}\boldsymbol{W}^T\boldsymbol{e}_i\|_2 \le \sqrt{\frac{\mu r}{n}}$, we have

$$\begin{aligned}
\boldsymbol{Y}_0 &= \max_i \left\|(\boldsymbol{L}\boldsymbol{W}_0\boldsymbol{\Lambda}^{-1}\boldsymbol{W}_0^T\boldsymbol{L} - \boldsymbol{L})^T \boldsymbol{e}_i\right\|_2 \\
&= \max_i \left\|(\boldsymbol{L}\boldsymbol{W}_0\boldsymbol{\Lambda}^{-1}\boldsymbol{W}_0^T\boldsymbol{L} - \boldsymbol{L})^T \boldsymbol{W}\boldsymbol{W}^T\boldsymbol{e}_i\right\|_2 \\
&\le \max_i \left\|\boldsymbol{W}\boldsymbol{W}^T\boldsymbol{e}_i\right\|_2 \left\|\boldsymbol{L}\boldsymbol{W}_0\boldsymbol{\Lambda}^{-1}\boldsymbol{W}_0^T\boldsymbol{L} - \boldsymbol{L}\right\|_2 \\
&\le \sqrt{\frac{\mu r}{n}} \left\|\boldsymbol{L}\boldsymbol{W}_0\boldsymbol{\Lambda}^{-1}\boldsymbol{W}_0^T\boldsymbol{L} - \boldsymbol{L}\right\|_2,
\end{aligned}$$

where the second equation follows from the fact that $\boldsymbol{L} = \boldsymbol{W}\boldsymbol{W}^T\boldsymbol{L}$.

Note that $\boldsymbol{L} = \boldsymbol{W}_0\boldsymbol{\Lambda}\boldsymbol{W}_0^T + \ddot{\boldsymbol{W}}_0\ddot{\boldsymbol{\Lambda}}\ddot{\boldsymbol{W}}_0^T - \boldsymbol{E}$, so we get

$$\begin{aligned}
&\left\|\boldsymbol{L}\boldsymbol{W}_0\boldsymbol{\Lambda}^{-1}\boldsymbol{W}_0^T\boldsymbol{L} - \boldsymbol{L}\right\|_2 \\
&\quad = \left\|(\boldsymbol{W}_0\boldsymbol{\Lambda}\boldsymbol{W}_0^T + \ddot{\boldsymbol{W}}_0\ddot{\boldsymbol{\Lambda}}\ddot{\boldsymbol{W}}_0^T - \boldsymbol{E})\boldsymbol{W}_0\boldsymbol{\Lambda}^{-1}\boldsymbol{W}_0^T(\boldsymbol{W}_0\boldsymbol{\Lambda}\boldsymbol{W}_0^T + \ddot{\boldsymbol{W}}_0\ddot{\boldsymbol{\Lambda}}\ddot{\boldsymbol{W}}_0^T - \boldsymbol{E}) - \boldsymbol{L}\right\|_2 \\
&\quad = \left\|\boldsymbol{W}_0\boldsymbol{\Lambda}\boldsymbol{W}_0^T - \boldsymbol{L} - \boldsymbol{W}_0\boldsymbol{W}_0^T\boldsymbol{E} - \boldsymbol{E}\boldsymbol{W}_0\boldsymbol{W}_0^T - \boldsymbol{E}\boldsymbol{W}_0\boldsymbol{\Lambda}^{-1}\boldsymbol{W}_0^T\boldsymbol{E}\right\|_2 \\
&\quad \le \left\|\boldsymbol{E} - \ddot{\boldsymbol{W}}_0\ddot{\boldsymbol{\Lambda}}\ddot{\boldsymbol{W}}_0^T\right\|_2 + 2\|\boldsymbol{E}\|_2 + \frac{\|\boldsymbol{E}\|_2^2}{|\lambda_r|} \\
&\quad \le \left\|\ddot{\boldsymbol{W}}_0\ddot{\boldsymbol{\Lambda}}\ddot{\boldsymbol{W}}_0^T\right\|_2 + 4\|\boldsymbol{E}\|_2 \\
&\quad \le |\lambda_{r+1}| + 4\|\boldsymbol{E}\|_2 \\
&\quad \le 5\|\boldsymbol{E}\|_2,
\end{aligned}$$

where the first and fourth inequality follow from (E.4) and (E.6), and $|\lambda_{r+1}| \le \|\boldsymbol{E}\|_2$ since $\sigma_{r+1}(\boldsymbol{L}) = 0$. Together, we have

$$\boldsymbol{Y}_0 \le 5\sqrt{\frac{\mu r}{n}}\,\|\boldsymbol{E}\|_2 \le 5\alpha\sqrt{\mu r n}\,\|\boldsymbol{E}\|_\infty\,, \tag{E.8}$$

where the last inequality follows from Lemma E.1.

Next, we will bound $\boldsymbol{Y}_{ab}$. Note that

$$\begin{aligned}
\boldsymbol{Y}_{ab} &= \max_i \left\|(\boldsymbol{E}^a\boldsymbol{L}\boldsymbol{W}_0\boldsymbol{\Lambda}^{-(a+b+1)}\boldsymbol{W}_0^T\boldsymbol{L}\boldsymbol{E}^b)^T\boldsymbol{e}_i\right\|_2 \\
&= \max_i \left\|\boldsymbol{E}^b\boldsymbol{L}\boldsymbol{W}_0\boldsymbol{\Lambda}^{-(a+b+1)}\boldsymbol{W}_0^T\boldsymbol{L}(\boldsymbol{W}\boldsymbol{W}^T\boldsymbol{E}^a\boldsymbol{e}_i)\right\|_2 \\
&\le \max_i \left\|\boldsymbol{W}^T\boldsymbol{E}^a\boldsymbol{e}_i\right\|_2 \left\|\boldsymbol{L}\boldsymbol{W}_0\boldsymbol{\Lambda}^{-(a+b+1)}\boldsymbol{W}_0^T\boldsymbol{L}\right\|_2 \|\boldsymbol{E}\|_2^b \\
&\le \sqrt{\frac{\mu r}{n}}(\alpha n\,\|\boldsymbol{E}\|_\infty)^{a+b}\left\|\boldsymbol{L}\boldsymbol{W}_0\boldsymbol{\Lambda}^{-(a+b+1)}\boldsymbol{W}_0^T\boldsymbol{L}\right\|_2 \\
&\le \alpha\sqrt{\mu r n}\,\|\boldsymbol{E}\|_\infty\left(\frac{\sigma_r(\boldsymbol{L})}{8}\right)^{a+b-1}\left\|\boldsymbol{L}\boldsymbol{W}_0\boldsymbol{\Lambda}^{-(a+b+1)}\boldsymbol{W}_0^T\boldsymbol{L}\right\|_2,
\end{aligned}$$

where the second inequality uses Lemmata E.1 and E.2. Moreover, by applying $\boldsymbol{L} = \boldsymbol{W}_0\boldsymbol{\Lambda}\boldsymbol{W}_0^T + \ddot{\boldsymbol{W}}_0\ddot{\boldsymbol{\Lambda}}\ddot{\boldsymbol{W}}_0^T - \boldsymbol{E}$, we get

$$\begin{aligned}
&\left\|\boldsymbol{L}\boldsymbol{W}_0\boldsymbol{\Lambda}^{-(a+b+1)}\boldsymbol{W}_0^T\boldsymbol{L}\right\|_2 \\
&= \left\|(\boldsymbol{W}_0\boldsymbol{\Lambda}\boldsymbol{W}_0^T + \ddot{\boldsymbol{W}}_0\ddot{\boldsymbol{\Lambda}}\ddot{\boldsymbol{W}}_0^T - \boldsymbol{E})\boldsymbol{W}_0\boldsymbol{\Lambda}^{-(a+b+1)}\boldsymbol{W}_0^T(\boldsymbol{W}_0\boldsymbol{\Lambda}\boldsymbol{W}_0^T + \ddot{\boldsymbol{W}}_0\ddot{\boldsymbol{\Lambda}}\ddot{\boldsymbol{W}}_0^T - \boldsymbol{E})\right\|_2 \\
&= \Big\|\boldsymbol{W}_0\boldsymbol{\Lambda}^{-(a+b-1)}\boldsymbol{W}_0^T - \boldsymbol{E}\boldsymbol{L}\boldsymbol{W}_0\boldsymbol{\Lambda}^{-(a+b)}\boldsymbol{W}_0^T - \boldsymbol{L}\boldsymbol{W}_0\boldsymbol{\Lambda}^{-(a+b)}\boldsymbol{W}_0^T\boldsymbol{E} \\
&\quad + \boldsymbol{E}\boldsymbol{L}\boldsymbol{W}_0\boldsymbol{\Lambda}^{-(a+b+1)}\boldsymbol{W}_0^T\boldsymbol{E}\Big\|_2 \\
&\le |\lambda_r|^{-(a+b-1)} + |\lambda_r|^{-(a+b)}\,\|\boldsymbol{E}\|_2 + |\lambda_r|^{-(a+b)}\,\|\boldsymbol{E}\|_2 + |\lambda_r|^{-(a+b+1)}\,\|\boldsymbol{E}\|_2^2 \\
&= |\lambda_r|^{-(a+b-1)}\left(1 + \frac{2\|\boldsymbol{E}\|_2}{|\lambda_r|} + \left(\frac{\|\boldsymbol{E}\|_2}{|\lambda_r|}\right)^2\right) \\
&= |\lambda_r|^{-(a+b-1)}\left(1 + \frac{\|\boldsymbol{E}\|_2}{|\lambda_r|}\right)^2 \\
&\le 2|\lambda_r|^{-(a+b-1)} \\
&\le 2\left(\frac{7}{8}\sigma_r(\boldsymbol{L})\right)^{-(a+b-1)},
\end{aligned}$$

where the second inequality follows from (E.6) and the last inequality follows from (E.5). Hence,

$$\begin{aligned}\sum_{a+b>0} \boldsymbol{Y}_{ab} &\le \sum_{a+b>0} 2\alpha\sqrt{\mu r n}\,\|\boldsymbol{E}\|_\infty \left(\frac{\frac{1}{8}\sigma_r^L}{\frac{7}{8}\sigma_r^L}\right)^{a+b-1} \\ &\le 2\alpha\sqrt{\mu r n}\,\|\boldsymbol{E}\|_\infty \sum_{a+b>0}\left(\frac{1}{7}\right)^{a+b-1} \\ &\le 2\alpha\sqrt{\mu r n}\,\|\boldsymbol{E}\|_\infty \left(\frac{1}{1-\frac{1}{7}}\right)^2 \\ &\le 3\alpha\sqrt{\mu r n}\,\|\boldsymbol{E}\|_\infty . \qquad \text{(E.9)}\end{aligned}$$

Combining (E.8) and (E.9), we have the *row version* of the inequality:

$$\begin{aligned}\max_i \left\|(\boldsymbol{L}_0-\boldsymbol{L})^T\boldsymbol{e}_i\right\|_2 &\le \boldsymbol{Y}_0 + \sum_{a+b>0}\boldsymbol{Y}_{ab} \\ &\le 5\alpha\sqrt{\mu r n}\,\|\boldsymbol{E}\|_\infty + 3\alpha\sqrt{\mu r n}\,\|\boldsymbol{E}\|_\infty \\ &\le \frac{32\alpha\mu^{1.5}r^{1.5}}{\sqrt{n}}\sigma_{\max}(\boldsymbol{L}),\end{aligned}$$

where the last step uses (E.2).

The *column version* of the inequality, i.e.,

$$\max_j \|(\boldsymbol{L}_0-\boldsymbol{L})\boldsymbol{e}_j\|_2 \le \frac{32\alpha\mu^{1.5}r^{1.5}}{\sqrt{n}}\sigma_{\max}(\boldsymbol{L}),$$

can be proved similarly. This finishes the proof. ∎

REFERENCES

[1] A. Aldroubi, K. Hamm, A. B. Koku, and A. Sekmen, *CUR decompositions, similarity matrices, and subspace clustering*, Front. Appl. Math. Stat., 4 (2019), p. 65.

[2] A. Aldroubi, A. Sekmen, A. B. Koku, and A. F. Cakmak, *Similarity matrix framework for data from union of subspaces*, Appl. Comput. Harmon. Anal., 45 (2018), pp. 425–435.

[3] H. Avron and C. Boutsidis, *Faster subset selection for matrices and applications*, SIAM J. Matrix Anal. Appl., 34 (2013), pp. 1464–1499.

[4] J. J. Bartholdi, III, *A good submatrix is hard to find*, Oper. Res. Lett., 1 (1982), pp. 190–193.

[5] A. Bhaskara, A. Rostamizadeh, J. Altschuler, M. Zadimoghaddam, T. Fu, and V. Mirrokni, *Greedy column subset selection: New bounds and distributed algorithms*, in Proceedings of the 33rd International Conference on Machine Learning, 2016, pp. 2539–2548.

[6] C. Boutsidis, P. Drineas, and M. Magdon-Ismail, *Near-optimal column-based matrix reconstruction*, SIAM J. Comput., 43 (2014), pp. 687–717.

[7] C. Boutsidis and D. P. Woodruff, *Optimal CUR matrix decompositions*, SIAM J. Comput., 46 (2017), pp. 543–589.

[8] T. Bouwmans, *Subspace learning for background modeling: A survey*, Recent Patents Comput. Sci., 2 (2009), pp. 223–234.

[9] T. Bouwmans, S. Javed, H. Zhang, Z. Lin, and R. Otazo, *On the applications of robust PCA in image and video processing*, Proc. IEEE, 106 (2018), pp. 1427–1457.

[10] H. Cai, *Accelerating Truncated Singular-Value Decomposition: A Fast and Provable Method for Robust Principal Component Analysis*, PhD thesis, University of Iowa, 2018.

[11] H. Cai, J.-F. Cai, T. Wang, and G. Yin, *Accelerated structured alternating projections for robust spectrally sparse signal recovery*, IEEE Trans. Signal Process., 69 (2021), pp. 809–821.
[12] H. Cai, J.-F. Cai, and K. Wei, *Accelerated alternating projections for robust principal component analysis*, J. Mach. Learn. Res., 20 (2019), pp. 685–717.
[13] H. Cai, K. Hamm, L. Huang, J. Li, and T. Wang, *Rapid robust principal component analysis: CUR accelerated inexact low rank estimation*, IEEE Signal Process. Lett., 28 (2021), pp. 116–120.
[14] E. J. Candès, X. Li, Y. Ma, and J. Wright, *Robust principal component analysis?*, J. ACM, 58 (2011), pp. 1–37.
[15] V. Chandrasekaran, S. Sanghavi, P. A. Parrilo, and A. S. Willsky, *Rank-sparsity incoherence for matrix decomposition*, SIAM J. Optim., 21 (2011), pp. 572–596.
[16] J. Chiu and L. Demanet, *Sublinear randomized algorithms for skeleton decompositions*, SIAM J. Matrix Anal. Appl., 34 (2013), pp. 1361–1383.
[17] A. Çivril, *Column subset selection problem is UG-hard*, J. Comput. System Sci., 80 (2014), pp. 849–859.
[18] M. Dereziński and M. K. Warmuth, *Reverse iterative volume sampling for linear regression*, J. Mach. Learn. Res., 19 (2018), pp. 853–891.
[19] A. Deshpande and L. Rademacher, *Efficient volume sampling for row/column subset selection*, in Proceedings of the 51st Annual IEEE Symposium on Foundations of Computer Science, IEEE, 2010, pp. 329–338.
[20] P. Drineas, R. Kannan, and M. W. Mahoney, *Fast Monte Carlo algorithms for matrices* III*: Computing a compressed approximate matrix decomposition*, SIAM J. Comput., 36 (2006), pp. 184–206.
[21] P. Drineas, M. Magdon-Ismail, M. W. Mahoney, and D. P. Woodruff, *Fast approximation of matrix coherence and statistical leverage*, J. Mach. Learn. Res., 13 (2012), pp. 3475–3506.
[22] P. Drineas, M. W. Mahoney, and S. Muthukrishnan, *Relative-error CUR matrix decompositions*, SIAM J. Matrix Anal. Appl., 30 (2008), pp. 844–881.
[23] E. Elhamifar and R. Vidal, *Sparse subspace clustering: Algorithm, theory, and applications*, IEEE Trans. Pattern Anal. Mach. Intell., 35 (2013), pp. 2765–2781.
[24] A. Georghiades, P. Belhumeur, and D. Kriegman, *From few to many: Illumination cone models for face recognition under variable lighting and pose*, IEEE Trans. Pattern Anal. Mach. Intelligence, 23 (2001), pp. 643–660.
[25] S. A. Goreinov, I. V. Oseledets, D. V. Savostyanov, E. E. Tyrtyshnikov, and N. L. Zamarashkin, *How to find a good submatrix*, in Matrix Methods: Theory, Algorithms and Applications: Dedicated to the Memory of Gene Golub, World Scientific, River Edge, NJ, 2010, pp. 247–256.
[26] S. A. Goreĭnov, E. E. Tyrtyshnikov, and N. L. Zamarashkin, *A theory of pseudoskeleton approximations*, Linear Algebra Appl., 261 (1997), pp. 1–21.
[27] S. A. Goreĭnov, N. L. Zamarashkin, and E. E. Tyrtyshnikov, *Pseudo-skeleton approximations*, Dokl. Akad. Nauk, 343 (1995), pp. 151–152.
[28] S. A. Goreĭnov, N. L. Zamarashkin, and E. E. Tyrtyshnikov, *Pseudo-skeleton approximations by matrices of maximal volume*, Math. Notes, 62 (1997), pp. 515–519.
[29] D. Gross and V. Nesme, *Note on Sampling Without Replacing from a Finite Collection of Matrices*, arXiv preprint, arXiv:1001.2738 [CS.IT], 2010.
[30] M. Gu and S. C. Eisenstat, *Downdating the singular value decomposition*, SIAM J. Matrix Anal. Appl., 16 (1995), pp. 793–810.
[31] B. D. Haeffele, C. You, and R. Vidal, *A critique of self-expressive deep subspace clustering*, International Conference on Learning Representations, 2021, https://openreview.net/pdf?id=FOyuZ26emy.
[32] K. Hamm and L. Huang, *Perturbations of CUR decompositions*, SIAM J. Matrix Anal. Appl., 42 (2021), pp. 351–375.
[33] K. Hamm and L. Huang, *Perspectives on CUR decompositions*, Appl. Comput. Harmon. Anal., 48 (2020), pp. 1088–1099.
[34] W.-D. Jang, C. Lee, and C.-S. Kim, *Primary object segmentation in videos via alternate convex optimization of foreground and background distributions*, in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 696–704.
[35] X. Li and Y. Pang, *Deterministic column-based matrix decomposition*, IEEE Trans. Knowledge and Data Engrg., 22 (2010), pp. 145–149.

[36] G. Liu, Z. Lin, S. Yan, J. Sun, Y. Yu, and Y. Ma, *Robust recovery of subspace structures by low-rank representation*, IEEE Trans. Pattern Anal. Mach. Intell., 35 (2012), pp. 171–184.
[37] M. W. Mahoney and P. Drineas, *CUR matrix decompositions for improved data analysis*, Proc. Natl. Acad. Sci. USA, 106 (2009), pp. 697–702.
[38] A. Mikhalev and I. V. Oseledets, *Rectangular maximum-volume submatrices and their applications*, Linear Algebra Appl., 538 (2018), pp. 187–211.
[39] P. Netrapalli, U. Niranjan, S. Sanghavi, A. Anandkumar, and P. Jain, *Non-convex robust PCA*, Adv. Neural Inf. Process. Syst., 27 (2014), pp. 1107–1115.
[40] I. Oseledets and E. Tyrtyshnikov, *Tt-cross approximation for multidimensional arrays*, Linear Algebra Appl., 432 (2010), pp. 70–88.
[41] A. Osinsky, *Rectangular Maximum Volume and Projective Volume Search Algorithms*, arXiv preprint, arXiv:1809.02334 [math.NA], 2018.
[42] A. Osinsky and N. L. Zamarashkin, *Pseudo-skeleton approximations with better accuracy estimates*, Linear Algebra Appl., 537 (2018), pp. 221–249.
[43] Y. Shitov, *Column subset selection is NP-complete*, Linear Algebra Appl., 610 (2021), pp. 52–58.
[44] D. C. Sorensen and M. Embree, *A DEIM induced CUR factorization*, SIAM J. Sci. Comput., 38 (2016), pp. A1454–A1482.
[45] G. W. Stewart, *On the perturbation of pseudo-inverses, projections and linear least squares problems*, SIAM Rev., 19 (1977), pp. 634–662.
[46] S.Wang and Z.Zhang, *Improving CUR matrix decomposition and the Nyström approximation via adaptive sampling*, J. Mach. Learn. Res., 14 (2013), pp. 2729–2769.
[47] J. A. Tropp, *Column subset selection, matrix factorization, and eigenvalue optimization*, in Proceedings of the 20th Annual ACM-SIAM Symposium on Discrete Algorithms, SIAM, 2009, pp. 978–986.
[48] J. A. Tropp, *Improved analysis of the subsampled randomized Hadamard transform*, Adv. Adapt. Data Anal., 3 (2011), pp. 115–126.
[49] R. Vidal, *Subspace clustering*, IEEE Signal Process Mag., 28 (2011), pp. 52–68.
[50] S. Voronin and P.-G. Martinsson, *Efficient algorithms for CUR and interpolative matrix decompositions*, Adv. Comput. Math., 43 (2017), pp. 495–516.
[51] J. Wright, A. Y. Yang, A. Ganesh, S. S. Sastry, and Y. Ma, *Robust face recognition via sparse representation*, IEEE Trans. Pattern Anal. Mach. Intell. Trans. Inform., 31 (2008), pp. 210–227.
[52] H. Xu, C. Caramanis, and S. Sanghavi, *Robust PCA via outlier pursuit*, IEEE Trans. Inform. Theory, 58 (2012), pp. 3047–3064.
[53] X. Yi, D. Park, Y. Chen, and C. Caramanis, *Fast algorithms for robust PCA via gradient descent*, in Proceedings of the 30th International Conference on Neural Information Processing Systems, 2016, pp. 4152–4160.