Compressive PCA for Low-Rank Matrices on Graphs

Nauman Shahid, Nathanael Perraudin, Gilles Puy, Pierre Vandergheynst

Compressive PCA for Low-Rank Matrices on Graphs

Nauman Shahid*, Nathanael Perraudin, Gilles Puy†, Pierre Vandergheynst
Email: {nauman.shahid, nathanael.perraudin, pierre.vandergheynst}@epfl.ch, † gilles.puy@gmail.com

***Abstract*— We introduce a novel framework for an approximate recovery of data matrices which are low-rank on graphs, from sampled measurements. The rows and columns of such matrices belong to the span of the first few eigenvectors of the graphs constructed between their rows and columns. We leverage this property to recover the non-linear low-rank structures efficiently from sampled data measurements, with a low cost (linear in n). First, a Resrtricted Isometry Property (RIP) condition is introduced for efficient uniform sampling of the rows and columns of such matrices based on the cumulative coherence of graph eigenvectors. Secondly, a state-of-the-art fast low-rank recovery method is suggested for the sampled data. Finally, several efficient, parallel and parameter-free decoders are presented along with their theoretical analysis for decoding the low-rank and cluster indicators for the full data matrix. Thus, we overcome the computational limitations of the standard *linear* low-rank recovery methods for big datasets. Our method can also be seen as a major step towards efficient recovery of non-linear low-rank structures. For a matrix of size $n \times p$, on a single core machine, our method gains a speed up of p^2/k over Robust Principal Component Analysis (RPCA), where $k \ll p$ is the subspace dimension. Numerically, we can recover a low-rank matrix of size 10304×1000, 100 times faster than Robust PCA.**

***Index Terms*—Robust PCA, graph Laplacian, spectral graph theory, compressive sampling**

I. Introduction

In many applications in signal processing, computer vision and machine learning, the data has an intrinsic low-rank structure. One desires to extract this structure efficiently from the noisy observations. Robust Principal Component Analysis (RPCA) [7], a linear dimensionality reduction algorithm can be used to exactly describe a dataset lying on a single linear low-dimensional subspace. Low-rank Representation (LRR) [19], on the other hand can be used for data drawn from multiple linear subspaces. However, these methods suffer from two prominent problems:

1) They do not recover non-linear low-rank structures.
2) They do not scale for big datasets $Y \in \Re^{p \times n}$ (large p and large n, where p is the number of features).

Many high dimensional datasets lie intrinsically on a smooth and very low-dimensional manifold that can be characterized by a graph $\mathcal{G}$ between the data samples [4]. For a matrix $Y \in \Re^{p \times n}$, a $\mathcal{K}$-nearest neighbor undirected graph between the rows or columns of Y is denoted as $G = (\mathcal{V}, \mathcal{E})$, where $\mathcal{E}$ is the set of edges and $\mathcal{V}$ is the set of vertices. The first step in the construction of G consists of connecting each y_i to its $\mathcal{K}$ nearest neighbors y_j (using Euclidean distance), resulting in $|\mathcal{E}|$ connections. The y_i correspond to rows of Y if the graph G is the row graph or to the columns if G is a column graph. The $\mathcal{K}$-nearest neighbors are non-symmetric but a symmetric weighted adjacency matrix W is computed via a Gaussian kernel as $W_{ij} = \exp(-\|(y_i - y_j)\|_2^2/\sigma^2)$ if y_j is connected to y_i or vice versa and 0 otherwise. Let D be the diagonal degree matrix of G which is given as: $D_{ii} = \sum_j W_{ij}$. Then, the combinatorial Laplacian that characterizes the graph G is defined as $\mathcal{L} = D - W$ and its normalized form as $\mathcal{L}_n = D^{-1/2}(D - W)D^{-1/2}$ [38].

It is imperative to represent such datasets as a function of the smooth variations of the non-linear manifold, rather than a linear subspace. We refer to such a representation as a *non-linear low-rank structure*. In this context, the graph eigenvectors serve as a legitimate tool to characterize the smooth variations of the manifold. Consider the example of a 2D circle embedded in a 3D space as shown in the left most plot of Fig. 1. The noisy version of this circle qualifies as an example of a non-linear low-rank (2D) manifold embedded in a high dimensional (3D) space. Ideally one would like to recover the 2D circle as shown in the rightmost plot of Fig. 1, however, RPCA just reduces the manifold to a point in the space. Extensions of RPCA and LRR such as Robust PCA on Graphs (RPCAG) [35] and Graph Regularized LRR (GLRR) [20] propose to incorporate graph regularization as a method to recover non-linear low-rank structures. These methods still suffer from the scalability problem for big datasets.

Randomized techniques come into play to deal with the scalability problem associated with very high dimensional data (the case of large p) [43], [5], [44], [18], [13], [23], [29], [30], [12] using the tools of compression [8]. These works improve upon the computational complexity by reducing only the *data dimension* p but still scale in the same manner w.r.t n. The case of large n can be tackled by using the sampling schemes accompanied with Nystrom method [40]. However, this method works efficiently only for low-rank kernel matrices and does not recover the low-rank data matrix itself. Scalable extensions of LRR such as [45] exist but they focus only on the subspace clustering application. Recently, Aravkin et. al [1] proposed to speed-up RPCA and ease the parameter selection problem, however, the variational approach does not qualify

Affiliation: Signal Processing Laboratory 2 (LTS2), EPFL STI IEL, Lausanne, CH-1015, Switzerland. Phone: +41 21 69 34754. † G. Puy contributed to this work while he was at INRIA (Rennes - Bretagne Atlantique, Campus de Beaulieu, FR-35042 Rennes Cedex, France). N.Shahid and N.Perraudin are supported by the SNF grant no. 200021_154350/1 for the project "Towards signal processing on graphs".G.Puy was funded by the European Research Council, PLEASE project (ERC-StG-2011-277906).

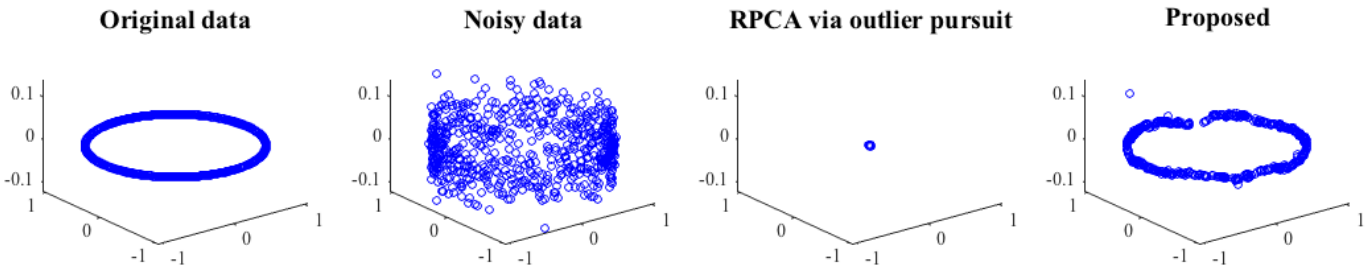

Figure 1: A 2D circle and its noisy version embedded in a 3D space, which qualify as a non-linear low-rank structure. The goal is to recover the circle from noise as shown in the rightmost plot, however, the state-of-the-art RPCA reduces the manifold to a point. Thus RPCA is not suitable to recover the non-linear low-rank structures.

to represent the non-linear low-rank structures. How to tackle the case of big n and non-linearity simultaneously then?

For many machine learning applications involving big data, such as clustering, an approximate low-rank representation might suffice. The recently introduced Fast Robust PCA on Graphs (FRPCAG) [36] approximates a recovery method for non-linear low-rank datasets, which are called *Low-rank matrices on graphs*. Inspired by the underlying stationarity assumption [27], the authors introduce a joint notion of low-rankness for the features and samples (rows and columns) of a data matrix. More specifically, a low-rank matrix on graphs is defined as a matrix whose rows and columns belong to the span of the first few eigenvectors of the graphs constructed between its rows and columns.

FRPCAG does not require an SVD and scales linearly with n. It relies on fast dual graph filtering operations which involve matrix vector multiplications and can be parallelized on a GPU in every iteration. However, the size of the problem is still an issue for big datasets because the problem cannot be broken down into small sub-problems and the solution merged at the end. Thus, for the non-GPU implementation, it still suffers from 1) memory requirements 2) cost of k-means for clustering 3) the cost of parameter tuning for large p and large n and 4) scalability for very big datasets. This said, sometimes one might not even have access to the full dataset Y. This is typical, for instance for the biomedical applications, such as MRI and tomography. In such applications the number of observations are limited by the data acquisition protocols. In MRI, the number of observations is proportional to the time and dose required for the procedure. In tomography one might have access to the projections only. Thus, FRPCAG is not be usable if 1) the dataset is large and 2) only a subset of the dataset or measurements are available. Despite the above limitations of the data acquisition, one might have access to some additional information about the unobserved samples. In MRI for instance, sparsity of the samples in the Fourier domain serves as a good prior.

A. *The Problem Statement*

In this work we answer the following questions: *1) What would be an efficient and highly scalable recovery framework, involving compression, for datasets which are jointly low-rank on two manifolds? 2) Alternatively, given a few randomly sampled observations and features from a data matrix $Y \in \Re^{p \times n}$, is it possible to efficiently recover the complete non-linear low-rank representation?* We mostly limit ourselves to the case 1 above, where a graphical prior is available or can be conveniently constructed for the complete set of observations for the application under consideration. A brief initial treatment of the 2nd case constitutes Section VII.C of this work.

B. *Contributions*

PCA has been widely used for two different types of applications: 1) Low-rank recovery and 2) clustering in the low-dimensional space. It is crucial to point out here that the clustering is not a standard application of PCA, because PCA is just a feature extraction method. However, the clustering experiments had been widely adopted as a standard procedure to demonstrate the quality of the feature extraction methods [11], [41], [46], [15], [6], [37], [14]. Thus, to be consistent with the state-of-the-art, our contributions focus on both of the above applications. Below we describe our contributions in detail.

1. Sampling & RIP for low-rank matrices on graphs: To solve the scalability problem of FRPCAG we propose to perform a dual uniform sampling of the data matrices, along rows and columns. We present a restricted isometry property (RIP) for low-rank matrices on graphs and relate it to the cumulative coherence of the graph eigenvectors. FRPCAG is then used to recover the low-rank representation for the sampled data.

2. Decoders for low-rank recovery: We present two (ideal and alternate) convex and efficient decoders for recovering the full low-rank matrix from the corresponding low-rank matrix of the sampled data. However, our main contribution comprises the set of 3 additional parallel, low-cost and parameter-free *approximate* decoders, which significantly boost the speed of our framework by introducing a few approximations. Our rigorous theoretical analysis also proves that the recovery error of the above decoders depends on the spectral gaps of the row and column graph Laplacians.

3. Low-Rank Clustering: For the clustering application of PCA, we propose a low-cost and parallel scheme based on CPCA. The key idea is to decode the labels of the complete

dataset from the labels of a sampled low-rank dataset, without computing the complete low-rank matrix.

4. Extensive Experimentation: Low-rank recovery experiments on 3 real video datasets and clustering experiments on 5 benchmark datasets reveal that the performance of our model is comparable to 10 different state-of-the-art PCA and non-PCA based methods. We also study some cases where CPCA fails to perform as well as the state-of-the-art.

Our proposed framework is inspired by the recently introduced sampling of band-limited signals on graphs [28]. While we borrow several concepts from here, our framework is significantly different from [28] in many contexts. We target the *low-rank* recovery of matrices, whereas [28] targets the recovery of band-limited signals / vectors. For our framework it is important for the data matrix to be low-rank jointly on the row and column graphs. Thus, our sampling scheme and RIP are generalized for two graphs. The design of a sampling scheme is the major focus of [28], while we just focus on the case of *uniform sampling* and instead focus on *how much to sample jointly* given the two graphs. Of course, our method can be extended directly for the other sampling schemes in [28]. A major difference lies in the application domain and hence the experiments. Unlike [28], we target two applications related to PCA: 1) low-rank recovery and 2) clustering. Thus, contrary to [28] our proposed decoders are designed for these applications. A major contribution of our work in contrast to [28] is the design of approximate decoders for low-rank recovery and clustering which significantly boost the speed of our framework for big datasets without compromising on the performance.

II. A Glimpse of Compressive PCA (CPCA)

Let $\mathcal{L}_c \in \mathbb{R}^{n\times n}$ be the Laplacian of the graph G_c connecting the different columns of Y and $\mathcal{L}_r \in \mathbb{R}^{p\times p}$ the Laplacian of the graph G_r that connects the rows of Y. Furthermore, let $\mathcal{L}_c = Q\Lambda_c Q^\top = Q_{k_c}\Lambda_{ck_c}Q_{k_c}^\top + \bar{Q}_{k_c}\bar{\Lambda}_{ck_c}\bar{Q}_{k_c}^\top$, where $\Lambda_{ck_c} \in \Re^{k_c\times k_c}$ is a diagonal matrix of lower eigenvalues and $\bar{\Lambda}_{ck_c} \in \Re^{(n-k_c)\times(n-k_c)}$ is a diagonal matrix of higher graph eigenvalues. Similarly, let $\mathcal{L}_r = P\Lambda_r P^\top = P_{k_r}\Lambda_{rk_r}P_{k_r}^\top + \bar{P}_{k_r}\bar{\Lambda}_{rk_r}\bar{P}_{k_r}^\top$. All the values in Λ_r and Λ_c are sorted in increasing order. For a $\mathcal{K}$-nearest neighbors graph constructed from k_c-clusterable data (along columns) one can expect $\lambda_{k_c}/\lambda_{k_c+1} \approx 0$ as $\lambda_{k_c} \approx 0$ and $\lambda_{k_c} \ll \lambda_{k_c+1}$. We refer to the ratio $\lambda_{k_c}/\lambda_{k_c+1}$ as the spectral gap of $\mathcal{L}_c$. The same holds for the Laplacian $\mathcal{L}_r$. Then, low-rank matrices on graphs can be defined as following and recovered by solving FRPCAG [36].

Definition 1. *A matrix $Y^* \in \mathbb{R}^{p\times n}$ is (k_r, k_c)-low-rank on the graphs $\mathcal{L}_r$ and $\mathcal{L}_c$ if its columns $y_j \in \text{span}(P_{k_r})$ for all $j = 1,\dots,n$ and its rows $y_i \in \text{span}(Q_{k_c})$ for all $i = 1,\dots,p$. The set of (k_r, k_c)-low-rank matrices on the graphs $\mathcal{L}_r$ and $\mathcal{L}_c$ is denoted by $\mathcal{LR}(P_{k_r}, Q_{k_c})$.*

Given a data matrix $Y \in \Re^{p\times n} = \bar{X} + \bar{E}$, where $\bar{X} \in \mathcal{LR}(P_{k_r}, Q_{k_c})$ and $\bar{E}$ models the errors, the goal is to develop a method to efficiently recover $\bar{X}$. We propose to 1) Construct Laplacians $\mathcal{L}_r$ and $\mathcal{L}_c$ between the rows and columns of Y using the scheme of Section II. 2) Sample the rows and columns of Y to get a subsampled matrix $\tilde{Y} = \tilde{Y}^* + E$ using the sampling scheme of Section III. 3) Construct the compressed Laplacians $\tilde{\mathcal{L}}_r, \tilde{\mathcal{L}}_c$ from $\mathcal{L}_r, \mathcal{L}_c$ (Section IV-A). 4) Determine a low-rank matrix $\tilde{X}$ for $\tilde{Y}$ with $\tilde{\mathcal{L}}_r, \tilde{\mathcal{L}}_c$ in algorithm 1 of FRPCAG:

$$\min_{\tilde{X}} \phi(\tilde{Y} - \tilde{X}) + \gamma_c \,\text{tr}(\tilde{X}\tilde{\mathcal{L}}_c\tilde{X}^\top) + \gamma_r \,\text{tr}(\tilde{X}^\top\tilde{\mathcal{L}}_r\tilde{X}),$$

where ϕ is a loss function (possibly l_p norm), $\tilde{X} = \tilde{X}^* + \tilde{E} = M_r\bar{X}M_c + \tilde{E}$, $\tilde{E}$ models the errors in the recovery of the subsampled low-rank matrix $\tilde{X}$ and M_r, M_c are the row and column sampling matrices whose design is discussed in Section III. 5) Use the decoders presented in Section V to decode the low-rank matrix $\bar{X} = \bar{X}^* + E^*$ (where E^* denotes the error on the recovery of optimal $\bar{X}^*$) on graphs $\mathcal{L}_r, \mathcal{L}_c$ if the task is low-rank recovery, or perform k-means on $\tilde{X}$ to get cluster labels $\tilde{C}$ and use the clustering algorithm (presented in Section VI) to get the cluster labels C for the full matrix X.

Throughout this work we use the approximate nearest neighbor algorithm (FLANN [21]) for graph construction whose complexity is $\mathcal{O}(np\log(n))$ for $p \ll n$ [33] (and it can be performed in parallel).

III. RIP for low-rank matrices on graphs

Let $M_r \in \Re^{\rho_r\times p}$ be the subsampling matrix for sampling the rows and $M_c \in \Re^{n\times\rho_c}$ for sampling the columns of Y. M_c and M_r are constructed by drawing ρ_c and ρ_r indices $\Omega_c = \{\omega_1 \cdots \omega_{\rho_c}\}$ and $\Omega_r = \{\omega_1 \cdots \omega_{\rho_r}\}$ **uniformly without replacement** from the sets $\{1, 2, \cdots, n\}$ and $\{1, 2, \cdots, p\}$ and satisfy:

$$M_c^{ij} = \begin{cases} 1 & \text{if } i = \omega_j \\ 0 & \text{otherwise} \end{cases} \quad M_r^{ij} = \begin{cases} 1 & \text{if } j = \omega_i \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

Now, the subsampled data matrix $\widetilde{Y} \in \Re^{\rho_c\times\rho_r}$ can be written as $\widetilde{Y} = M_r Y M_c$. CPCA requires M_r and M_c to be constructed such that the "low-rankness" property of the data Y is preserved under sampling. Before discussing this, we introduce a few basic definitions in the context of graphs G_c and G_r.

Definition 2. ***(Graph cumulative coherence).*** *The cumulative coherence of order k_c, k_r of G_c and G_r is:*

$$\nu_{k_c} = \max_{1\le i\le n} \sqrt{n}\|Q_{k_c}^\top\Delta_i^c\|_2 \quad \& \quad \nu_{k_r} = \max_{1\le j\le p} \sqrt{p}\|P_{k_r}^\top\Delta_j^r\|_2,$$

where $\Delta^c \in \{0,1\}^n, \Delta^r \in \{0,1\}^p$ are binary vectors and $\Delta_i^c = 1$ if the i^{th} entry of Δ^c is 1 and 0 otherwise. Thus, Δ_i^c corresponds to a specific node of the graph.

In the above equations $Q_{k_c}^\top\Delta_i^c$ and $P_{k_r}^\top\Delta_j^r$ characterize the first k_c and k_r fourier modes [38] of the nodes i and j on the graphs G_c and G_r respectively. Thus, the cumulative coherence is a measure of how well the energy of the (k_r, k_c) low-rank matrices spreads over the nodes of the graphs. These quantities exactly control the number of vertices ρ_c and ρ_r that need to be sampled from the graphs G_r and G_c such that the properties of the graphs are preserved [28].

Consider the example where a particular node i has a high coherence. Then, it implies that their exist some low-rank signals whose energy is highly concentrated on the node i. Removing this node would result in a loss of information in the data. If the coherence of this node is low then removing it in the sampling process would result in no loss of information. We already mentioned that we are interested in the case of uniform sampling. Therefore in order to be able to sample a small number of nodes uniformly from the graphs, the cumulative coherence should be as low as possible.

We remind that for our application we desire to sample the data matrix Y such that its low-rank structure is preserved under this sampling. How can we ensure this via the graph cumulative coherence? This follows directly from the fact that we are concerned about the data matrices which are also low-rank with respect to the two graphs under consideration $Y \in \mathcal{LR}(P_{k_r}, Q_{k_c})$. In simple words, the columns of the data matrix Y belong to the span of the eigenvectors P_{k_r} and the rows to the span of Q_{k_c}. Thus, the coherence conditions for the graph directly imply the coherence condition on the data matrix Y itself. Therefore, using these quantities to sample the data matrix Y will ensure the preservation of two properties under sampling: 1) the structure of the corresponding graphs and 2) the low-rankness of the data matrix Y. Given the above definitions, we are now ready to present the restricted-isometry theorem for the low-rank matrices on the graphs.

Theorem 1. ***(Restricted-isometry property (RIP) for low-rank matrices on graphs)*** *Let M_c and M_r be two random subsampling matrices as constructed in (1). For any $\delta, \epsilon \in (0,1)$, with probability at least $1-\epsilon$,*

$$(1-\delta)\|Y\|_F^2 \leq \frac{np}{\rho_r \rho_c}\|M_r Y M_c\|_F^2 \leq (1+\delta)\|Y\|_F^2 \qquad (2)$$

for all $Y \in \mathcal{LR}(P_{k_r}, Q_{k_c})$ provided that

$$\rho_c \geq \frac{27}{\delta^2}\nu_{k_c}^2 \log\left(\frac{4k_c}{\epsilon}\right) \;\; \& \;\; \rho_r \geq \frac{27}{\delta^2}\nu_{k_r}^2 \log\left(\frac{4k_r}{\epsilon}\right), \qquad (3)$$

where ν_{k_c}, ν_{k_r} characterize the graph cumulative coherence as in Definition 2 and $\frac{np}{\rho_c\rho_r}$ is just a normalization constant which quantifies the norm conservation in (2).

Proof. Please refer to Appendix A. □

Theorem 1 is a direct extension of the RIP for k-bandlimited signals on one graph [28]. It states that the information in $Y \in \mathcal{LR}(P_{k_r}, Q_{k_c})$ is preserved with overwhelming probability if the sampling matrices (1) are constructed with a uniform sampling strategy satisfying (3). Note that ρ_r and ρ_c depend on the cumulative coherence of the graph eigenvectors. The better spread the eigenvectors are, the smaller is the number of vertices that need to be sampled.

It is proved in [28] that $\nu_{k_c} \geq \sqrt{k_c}$ and $\nu_{k_r} \geq \sqrt{k_r}$. Hence, when the lower bounds are attained, one only needs to sample an order of $O(k_c \log(k_c))$ columns and $O(k_r \log(k_r))$ rows to ensure that the RIP (eq. (3)) holds. This is the ideal scenario. However, one can also have $\nu_{k_c} = \sqrt{n}$ or $\nu_{k_r} = \sqrt{p}$ in some situations. Let us give some examples.

The lower bound on ν_k is attained, e.g, when the graph is the regular lattice. In this case the graph Fourier transform is the "usual" Fourier transform and $\nu_k = \sqrt{k}$ for all k. Another example where the lower bound is attained is when the graph contains k disconnected components of identical size. In this case, one can prove that $\nu_k = \sqrt{k}$ [28]. Intuitively, the coherence remains close to this lower bound when these k components are weakly interconnected [28].

The upper bound on ν_k is attained when, for example, the graph has one of its nodes not connected to any other node. In this case, one must sample this node. Indeed, there is no way to guess the value of the signal on this node from any neighbour. As the sampling is random, one is sure to sample this node only when all the nodes are sampled. Uniform sampling is not the best strategy in this setting. Furthermore, note that such a case is only possible if the graph is noisy or the data has strong outliers. One should resort to a more distribution aware sampling in such a case as presented in [28].

We choose in this paper to present the results using a uniform distribution for simplicity. Note however that one can adapt the sampling distribution to the underlying structure of the graph to ensure optimal sampling results. A consequence of the result in [28] is that there always exist distributions that ensure that the RIP holds when sampling $O(k_r \log(k_r))$ rows and $O(k_c \log(k_c))$ columns only. The optimal sampling distribution for which this result holds is defined in [28] (see Section 2.2). Furthermore, a fast algorithm to compute this distribution also exists (Section 4 of [28]).

IV. Compressed Low-Rank Matrix

Once the compressed dataset $\tilde{Y} \in \Re^{\rho_r \times \rho_c}$ is obtained the low-rank representation has to be extracted which takes into account the graph structures. Thus we propose the following two step strategy:

1) Construct graphs for compressed data.
2) Run Fast Robust PCA on Graphs (FRPCAG) on the compressed data.

These two steps are elaborated in the following subsections.

A. Graphs for Compressed data

To ensure the preservation of algebraic and spectral properties one can construct the compressed Laplacians $\tilde{\mathcal{L}}_r \in \Re^{\rho_r \times \rho_r}$ and $\tilde{\mathcal{L}}_c \in \Re^{\rho_c \times \rho_c}$ from the Kron reduction of $\mathcal{L}_r$ and $\mathcal{L}_c$ [9]. Let Ω be the set of sampled nodes and $\bar{\Omega}$ the complement set and let $\mathcal{L}(A_r, A_c)$ denote the (row, column) sampling of $\mathcal{L}$ w.r.t sets A_r, A_c then the Laplacian $\tilde{\mathcal{L}}_c$ for the columns of compressed matrix $\tilde{Y}$ is:

$$\tilde{\mathcal{L}}_c = \mathcal{L}_c(\Omega,\Omega) - \mathcal{L}_c(\Omega,\bar{\Omega})\mathcal{L}_c^{-1}(\bar{\Omega},\bar{\Omega})\mathcal{L}_c(\bar{\Omega},\Omega).$$

Let $\mathcal{L}_c$ has k_c connected components or $\lambda_{k_c}/\lambda_{k_c+1} \approx 0$. Then, as argued in theorem $III.4$ of [9] two nodes α, β are not connected in $\tilde{\mathcal{L}}_c$ if there is no path between them in $\mathcal{L}_c$ via $\bar{\Omega}$. Assume that each of the connected components has the same number of nodes. Then, if the sampling is done uniformly within each of the connected components according to the sampling bounds described in eq.(3), one can expect $\tilde{\mathcal{L}}_c$ to

have k_c connected components as well. This is an inherent property of the Kron reduction method. However, for the case of large variation of the number of nodes among the connected components one might want to resort to a more distribution aware sampling scheme. Such schemes have been discussed in [28] and have not been addressed in this work. Nevertheless, the Kron reduction strategy mentioned here is independent of the sampling strategy used. The same concepts holds for $\tilde{\mathcal{L}}_r$ as well.

The Kron reduction method involves the multiplication of 3 sparse matrices. The only expensive operation above is the inverse of $\mathcal{L}(\bar{\Omega}, \bar{\Omega})$ which can be performed with $\mathcal{O}(O_l \mathcal{K} n)$ cost using the Lancoz method [39], where O_l is the number of iterations for Lancoz approximation.

B. *FRPCAG on the Compressed Data*

Once the Laplacians $\tilde{\mathcal{L}}_r \in \Re^{\rho_r \times \rho_r}, \tilde{\mathcal{L}}_c \in \Re^{\rho_c \times \rho_c}$ are obtained, the next step is to recover the low-rank matrix $\tilde{X} \in \Re^{\rho_r \times \rho_c}$. Let $\tilde{\mathcal{L}}_c = \tilde{Q}\tilde{\Lambda}_c\tilde{Q}^\top = \tilde{Q}_{k_c}\tilde{\Lambda}_{ck_c}\tilde{Q}_{k_c}^\top + \bar{\tilde{Q}}_{k_c}\bar{\tilde{\Lambda}}_{ck_c}\bar{\tilde{Q}}_{k_c}^\top$, where $\tilde{\Lambda}_{k_c} \in \Re^{k_c \times k_c}$ is a diagonal matrix of lower eigenvalues and $\bar{\tilde{\Lambda}}_{k_c} \in \Re^{(\rho_c - k_c)\times(\rho_c - k_c)}$ is a diagonal matrix of higher graph eigenvalues. Similarly, let $\tilde{\mathcal{L}}_r = \tilde{P}\tilde{\Lambda}_r\tilde{P}^\top = \tilde{P}_{k_r}\tilde{\Lambda}_{rk_r}\tilde{P}_{k_r}^\top + \bar{\tilde{P}}_{k_r}\bar{\tilde{\Lambda}}_{rk_r}\bar{\tilde{P}}_{k_r}^\top$. Furthermore assume that all the values in $\tilde{\Lambda}_r$ and $\tilde{\Lambda}_c$ are sorted in increasing order.

Assume $\tilde{Y} = \tilde{Y}^* + E$, where E models the noise in the compressed data and $\tilde{Y}^* \in \mathcal{LR}(\tilde{P}_{k_r}, \tilde{Q}_{k_c})$. The low-rank matrix $\tilde{X} = \tilde{X}^* + \tilde{E}$ can be recovered by solving the FRPCAG problem as proposed in [36] and re-written below:

$$\min_{\tilde{X}} \phi(\tilde{Y} - \tilde{X}) + \gamma_c \operatorname{tr}(\tilde{X}\tilde{\mathcal{L}}_c\tilde{X}^\top) + \gamma_r \operatorname{tr}(\tilde{X}^\top\tilde{\mathcal{L}}_r\tilde{X}), \quad (4)$$

where ϕ is a proper, positive, convex and lower semi-continuous loss function (possibly l_p norm). From Theorem 1 in [36], the low-rank approximation error comprises the orthogonal projection of $\tilde{X}^*$ on the complement graph eigenvectors $(\bar{\tilde{Q}}_{k_c}, \bar{\tilde{P}}_{k_r})$ and depends on the spectral gaps $\tilde{\lambda}_{k_c}/\tilde{\lambda}_{k_c+1}, \tilde{\lambda}_{k_r}/\tilde{\lambda}_{k_r+1}$ as following:

$$\begin{aligned} &\|\tilde{X}^*\bar{\tilde{Q}}_{k_c}\|_F^2 + \|\bar{\tilde{P}}_{k_r}^\top\tilde{X}^*\|_F^2 = \|\tilde{E}\|_F^2 \\ &\leq \frac{1}{\gamma}\phi(E) + \|\tilde{Y}^*\|_F^2\Big(\frac{\tilde{\lambda}_{k_c}}{\tilde{\lambda}_{k_c+1}} + \frac{\tilde{\lambda}_{k_r}}{\tilde{\lambda}_{k_r+1}}\Big), \end{aligned} \quad (5)$$

where γ depends on the signal-to-noise ratio. Clearly, if $\lambda_{k_c}/\lambda_{k_c+1} \approx 0$ and $\lambda_{k_r}/\lambda_{k_r+1} \approx 0$ and the compressed Laplacians are constructed using the Kron reduction then $\tilde{\lambda}_{k_c}/\tilde{\lambda}_{k_c+1} \approx 0$ and $\tilde{\lambda}_{k_r}/\tilde{\lambda}_{k_r+1} \approx 0$. Thus, exact recovery is attained.

Let $g(Z) = \gamma_c \operatorname{tr}(Z\mathcal{L}_c Z^\top) + \gamma_r \operatorname{tr}(Z^\top\mathcal{L}_r Z)$, then $\nabla_g(Z) = 2(\gamma_c Z\mathcal{L}_c + \gamma_r\mathcal{L}_r Z)$. Also define $\operatorname{prox}_{\lambda h}(Z) = Y + \operatorname{sgn}(Z - Y) \circ \max(|Z - Y| - \lambda, 0)$, where $\circ$ denotes the Hadamard product, λ as the step size (we use $\lambda = \frac{1}{\beta'}$), where $\beta \leq \beta' = 2\gamma_c\|\mathcal{L}_c\|_2 + 2\gamma_r\|\mathcal{L}_r\|_2$ and $\|\mathcal{L}\|_2$ is the spectral norm (or maximum eigenvalue) of $\mathcal{L}$, ϵ as the stopping tolerance and J the maximum number of iterations. Then FRPCAG can be solved by the FISTA in Algorithm 1.

Algorithm 1 FISTA for FRPCAG

INPUT: $Z_1 = Y$, $S_0 = Y$, $t_1 = 1$, $\epsilon > 0$
for $j = 1, \dots J$ **do**
  $S_j = \operatorname{prox}_{\lambda_j h}(Z_j - \lambda_j \nabla g(Z_j))$
  $t_{j+1} = \frac{1+\sqrt{1+4t_j^2}}{2}$
  $Z_{j+1} = S_j + \frac{t_j - 1}{t_{j+1}}(S_j - S_{j-1})$
  if $\|Z_{j+1} - Z_j\|_F^2 < \epsilon\|Z_j\|_F^2$ **then**
    BREAK
  end if
end for
OUTPUT: U_{j+1}

V. Decoders for Low-Rank Recovery

Let $\tilde{X} \in \Re^{\rho_r \times \rho_c}$ be the low-rank solution of (4) with the compressed graph Laplacians $\tilde{\mathcal{L}}_r, \tilde{\mathcal{L}}_c$ and sampled data $\tilde{Y}$. The goal is to decode the low-rank matrix $X \in \Re^{p \times n}$ for the full Y. We assume that $\tilde{X} = M_r\bar{X}M_c + \tilde{E}$, where $\tilde{E} \in \Re^{\rho_r \times \rho_c}$ models the noise incurred by (4).

A. *Ideal Decoder*

A straight-forward way to decode X on the original graphs $\mathcal{L}_r$ and $\mathcal{L}_c$, when one knows the basis P_{k_r}, Q_{k_c} involves solving the following optimization problem:

$$\begin{aligned} &\min_X \|M_r X M_c - \tilde{X}\|_F^2 \\ &\text{s.t: } (X)_i \in span(P_{k_r}),\ (X^\top)_j \in span(Q_{k_c}). \end{aligned} \quad (6)$$

Theorem 2. *Let M_r and M_c be such that (2) holds and X^* be the solution of (6) with $\tilde{X} = M_r\bar{X}M_c + \tilde{E}$, where $\bar{X} \in \mathcal{LR}(P_{k_r}, Q_{k_c})$ and $\tilde{E} \in \Re^{\rho_r \times \rho_c}$. We have:*

$$\|X^* - \bar{X}\|_F \leq 2\sqrt{\frac{np}{\rho_c\rho_r(1-\delta)}}\|\tilde{E}\|_F, \quad (7)$$

where $\sqrt{np/\rho_c\rho_r(1-\delta)}$ is a constant resulting from the norm preservation in (2) and $\|\tilde{E}\|_F^2$ is bounded by eq. (5).

Proof. Please refer to Appendix B. □

Thus, the error of the ideal decoder is only bounded by the error $\tilde{E}$ in the low-rank matrix $\tilde{X}$ obtained by solving (4). In fact $\tilde{E}$ depends on the spectral gaps of $\tilde{\mathcal{L}}_c, \tilde{\mathcal{L}}_r$, as given in eq. (5). Hence, the ideal decoder itself does not introduce any error in the decode stage. The solution for this decoder requires projecting over the eigenvectors P and Q of $\mathcal{L}_r$ and $\mathcal{L}_c$. This is computationally expensive because diagonalization of $\mathcal{L}_r$ and $\mathcal{L}_c$ cost $\mathcal{O}(p^3)$ and $\mathcal{O}(n^3)$. Moreover, the constants k_r, k_c are not known beforehand and require tuning.

B. *Alternate Decoder*

As the ideal decoder is computationally costly, we propose to decode X from $\tilde{X}$ by using a convex and computationally tractable problem which involves the minimization of graph dirichlet energies.

$$\min_X \|M_r X M_c - \tilde{X}\|_F^2 + \bar{\gamma_c} \operatorname{tr}(X\mathcal{L}_c X^\top) + \bar{\gamma_r} \operatorname{tr}(X^\top\mathcal{L}_r X). \quad (8)$$

Theorem 3. *Let M_r and M_c be such that (2) holds and $\gamma > 0$. Let also X^* be the solution of (8) with $\bar{\gamma}_c = \gamma/\lambda_{k_c+1}$, $\bar{\gamma}_r = \gamma/\lambda_{k_r+1}$, and $\tilde{X} = M_r\bar{X}M_c + \tilde{E}$, where $\bar{X} \in \mathcal{LR}(P_{k_r}, Q_{k_c})$ and $\tilde{E} \in \Re^{\rho_r \times \rho_c}$. We have:*

$$\|\bar{X}^* - \bar{X}\|_F \leq \sqrt{\frac{np}{\rho_c\rho_r(1-\delta)}}\Bigg[\Big(2 + \frac{1}{\sqrt{2\gamma}}\Big)\|\tilde{E}\|_F + \Big(\frac{1}{\sqrt{2}} + \sqrt{\gamma}\Big)\sqrt{\Big(\frac{\lambda_{k_c}}{\lambda_{k_c+1}} + \frac{\lambda_{k_r}}{\lambda_{k_r+1}}\Big)}\|\bar{X}\|_F\Bigg], \quad \textit{and}$$

$$\|E^*\|_F \leq \frac{\|\tilde{E}\|_F}{\sqrt{2\gamma}} + \frac{1}{\sqrt{2}}\sqrt{\Big(\frac{\lambda_{k_c}}{\lambda_{k_c+1}} + \frac{\lambda_{k_r}}{\lambda_{k_r+1}}\Big)}\|\bar{X}\|_F, \quad (9)$$

where $\bar{X}^* = \text{Proj}_{\mathcal{LR}(P_{k_r}, Q_{k_c})}(X)$ and $E^* = X^* - \bar{X}^*$. $\text{Proj}_{\mathcal{LR}(P_{k_r}, Q_{k_c})}(.)$ denotes the orthogonal projection onto $\mathcal{LR}(P_{k_r}, Q_{k_c})$ and γ depends on the signal to noise ratio.

Proof. Please refer to Appendix C □

Theorem 3 states that in addition to the error $\tilde{E}$ in $\tilde{X}$ incurred by (4) and characterized by the bound in eq. (5), the error of the alternate decoder (8) also depends on the spectral gaps of the Laplacians $\mathcal{L}_r$ and $\mathcal{L}_c$ respectively. This is the price that one has to pay in order to avoid the expensive ideal decoder. For a k_r, k_c clusterable data Y across the rows and columns, one can expect $\lambda_{k_r}/\lambda_{k_r+1} \approx 0$ and $\lambda_{k_c}/\lambda_{k_c+1} \approx 0$ and the solution is as good as the ideal decoder. Nevertheless, it is possible to reduce this error by using graph filters g such that the ratios $g(\lambda_{k_c})/g(\lambda_{k_c+1})$ and $g(\lambda_{k_r})/g(\lambda_{k_r+1})$ approach zero. However, we do not discuss this approach in our work. It is trivial to solve (8) using a conjugate gradient scheme that costs $\mathcal{O}(Inp\mathcal{K})$, where I is the number of iterations for the algorithm to converge.

C. *Approximate Decoder*

The alternate decoder proposed above has the following disadvantages: 1) It is almost as computationally expensive as FRPCAG 2) It requires tuning two model parameters.

In this section we describe the step-by-step construction of an approximate decoder which overcomes these limitations. The main idea is to breakdown the decode phase of low-rank matrix X into its left and right singular vectors or subspaces. Let $X = U\Sigma V^\top$ and $\tilde{X} = \tilde{U}\tilde{\Sigma}\tilde{V}^\top$ be the SVD of X and $\tilde{X}$. We propose to recover U from $\tilde{U}$ and V from $\tilde{V}$ in 3 steps.

1) Split the alternate decoder to subspace learning problems.
2) Drop the orthonormality constraints on subspaces.
3) Run an efficient upsampling algorithm to solve the problem of step 2.

The goal of this step-by-step approach is to guide the reader throughout to observe the close relationship between the alternate and approximate decoder. Now we begin to describe these steps in detail.

*1) **Step 1: Splitting the alternate decoder**:* Using the SVD of X and $\tilde{X}$ and the invariance property of the trace under cyclic permutations, we can replace (8) by:

$$\min_{U,V}\|M_rU\Sigma V^\top M_c - \tilde{U}\tilde{\Sigma}\tilde{V}^\top\|_F^2 + \bar{\gamma}_c\,\text{tr}(\Sigma^2V^\top\mathcal{L}_cV) + \bar{\gamma}_r\,\text{tr}(U^\top\mathcal{L}_rU\Sigma^2) \quad \text{s.t: } U^\top U = I_k,\ V^\top V = I_k. \quad (10)$$

The above eq. introduces two new variables based on the SVD of X, i.e, $U \in \mathbb{R}^{p\times k}$ and $V \in \mathbb{R}^{n\times k}$. Clearly, with the introduction of these new variables, one needs to specify k as the dimension of the subspaces U and V. We propose the following strategy for this:

1) First, determine $\tilde{\Sigma}$ by one inexpensive SVD of $\widetilde{X} \in \mathbb{R}^{\rho_r\times\rho_c}$. This costs $\mathcal{O}(\rho_r^2\rho_c)$ for $\rho_r < \rho_c$.
2) Then set k equal to the number of entries in $\tilde{\Sigma}$ which are above a threshold.

It is important to note that so far eq.(10) and the alternate decoder eq.(8) are equivalent. Also note that we did not introduce the singular values Σ as an additional variable in eq.(10) because they are related to the singular values $\tilde{\Sigma}$ of $\tilde{X}$. We argue this as following: If (9) holds for the alternate decoder then $\|\bar{\Sigma}^* - \bar{\Sigma}\|_F$ (where $\bar{\Sigma}^*, \bar{\Sigma}$ are the singular values of $\bar{X}^*, \bar{X}$) is also bounded as argued in the discussion of Appendix C. Thus, the singular values Σ and $\tilde{\Sigma}$ of X and $\tilde{X}$ differ approximately by the normalization constant of theorem 1, i.e,

$$\Sigma = \sqrt{\frac{np}{\rho_r\rho_c(1-\delta)}}\tilde{\Sigma}$$

Note that with the above relationship, the subspaces U, V can be solved independently of each other. Thus eq.(10) can be decoupled as following which separately solves the subspace (U and V) learning problems.

$$\min_U\|M_rU - \tilde{U}\|_F^2 + \gamma^{'}{}_r\,\text{tr}(U^\top\mathcal{L}_rU) \quad \text{s.t:} \quad U^\top U = I_k,$$
$$\min_V\|V^\top M_c - \tilde{V}\|_F^2 + \gamma^{'}{}_c\,\text{tr}(V^\top\mathcal{L}_cV) \quad \text{s.t:} \quad V^\top V = I_k. \quad (11)$$

*2) **Step 2: Dropping Orthonormality Constraints**:* Solving (11) is as expensive as (8) due to the orthonormality constraints (as explained in appendix D). Therefore, we drop the constraints and get

$$\min_U\|M_rU - \tilde{U}\|_F^2 + \gamma^{'}{}_r\,\text{tr}(U^\top\mathcal{L}_rU), \quad (12)$$

$$\min_V\|V^\top M_c - \tilde{V}\|_F^2 + \gamma^{'}{}_c\,\text{tr}(V^\top\mathcal{L}_cV). \quad (13)$$

The solutions to (12) & (13) are not orthonormal anymore. The deviation from the orthonormality depends on the constants $\gamma_r^{'}$ and $\gamma_c^{'}$, but $X = U\Sigma V^\top$ is still a good enough (error characterized in Theorem 4) low-rank representation due to the intuitive explanation that we present here. We argue that the solutions of eqs.(12) &(13) are feasible solutions of the joint non-convex, factorized, and graph regularized low-rank optimization problem like the one presented in [31]. Let A

and B be the subspaces that we want to recover then we can re-write the problem studied in [31] as following:

$$\min_{A,B} \|M_r A B^\top M_c^\top - \tilde{X}\|_F^2 + \gamma_r^{'} \operatorname{tr}(A^\top \mathcal{L}_r A) + \gamma_c^{'} \operatorname{tr}(B^\top \mathcal{L}_c B)$$

The above non-convex problem does not require A and B to be orthonormal, but is still widely used for recovering a low-rank $X = AB^\top$. Our problem setting (eqs.(12) &(13)) is just equivalent except that it is convex as we decouple the learning of two subspaces due to the known Σ that relates U and V. Thus, for any orthonormal U, V and a scaling matrix Σ, $A = U\sqrt{\Sigma}$ and $B = V\sqrt{\Sigma}$ is a feasible solution. Thus, dropping the orthonormality constraints does not effect the final solution X.

*3) **Step 3: Subspace Upsampling**:* Eqs. (12) &(13) require the tuning of two parameters $\gamma_r^{'}$ and $\gamma_c^{'}$ which can be computationally cumbersome. Therefore, our final step in the construction of the approximate decoder is to get rid of the two parameters. But before we present the final construction step we study the problems eqs.(12) &(13) and their solutions more closely.

First, note that solving eqs.(12) &(13) is equivalent to making the following assumptions:

$$\tilde{U} = M_r \bar{U} + \tilde{E}^u \quad \text{and} \quad \tilde{V} = \bar{V} M_c + \tilde{E}^v,$$

where the columns of $\bar{U}$, $\bar{u}_i \in span(P_{k_r})$, $i = 1, \cdots, p$, and the columns of $\bar{V}$ $\bar{v}_j \in span(Q_{k_c})$, $j = 1, \cdots, n$ and $\tilde{E}^u \in \Re^{\rho_r \times \rho_r}$, $\tilde{E}^v \in \Re^{\rho_c \times \rho_c}$ model the noise in the estimate of the subspaces.

Secondly, the closed form solutions of eqs.(12) &(13) are given as following:

$$U = (M_r^\top M_r + \gamma_r^{'} \mathcal{L}_r)^{-1} M_r^\top \tilde{U}, \tag{14}$$

$$V = (M_c M_c^\top + \gamma_c^{'} \mathcal{L}_c)^{-1} M_c \tilde{V}. \tag{15}$$

Thus, problems (12) & (13) decode the subspaces U and V such that they are smooth on their respective graphs $\mathcal{L}_r$ and $\mathcal{L}_c$. *This can also be referred to as 1) simultaneous decoding and 2) subspace denoising stage*. We call it a 'subspace denoising' method because the operator $(M_r^\top M_r + \gamma_r^{'} \mathcal{L}_r)^{-1}$ can be viewed as low-pass filtering the subspace U in the graph fourier domain.

Note that we want to decode and denoise $\tilde{U}$ and $\tilde{V}$ which are in turn determined by the SVD of $\tilde{X}$. Furthermore, $\tilde{X}$ has been determined by solving the FRPCAG problem of eq.(4). FRPCAG is already robust to noise and outliers, therefore, it is safe to assume that the subspaces determined from it, i.e, $\tilde{U}$ and $\tilde{V}$ are also noise and outlier free. Thus, the extra denoising step (performed via graph filtering) of eqs.(12) &(13) is redundant.

Therefore, we can directly upsample $\tilde{U}$ and $\tilde{V}$ to determine U and V without needing a margin for noise. To do this, we reformulate eq.(12) as follows:

$$\min_U \frac{1}{\gamma_r^{'}} \|M_r U - \tilde{U}\|_F^2 + \operatorname{tr}(U^\top \mathcal{L}_r U).$$

For $\gamma_r^{'} \to 0$, $\frac{1}{\gamma_r^{'}} \to \infty$, the emphasis on first term of the objective increases and it turns to an equality constraint $M_r U = \tilde{U}$. The same holds for eq.(13) as well. Thus, the modified problems are:

$$\begin{aligned} &\min_U \operatorname{tr}(U^\top \mathcal{L}_r U) \quad \text{and} \quad \min_V \operatorname{tr}(V^\top \mathcal{L}_c V) \\ &\text{s.t: } M_r U = \tilde{U}, \qquad\qquad \text{s.t: } M_c^\top V = \tilde{V}. \end{aligned} \tag{16}$$

Note that now we have a parameter-free decode stage.

It is important now to study the theoretical guarantees on eqs. (16). To do this, as eqs. (16) are a specific case of eqs. (12) & (13), we first study the guarantees on eqs. (12) & (13) in Theorem 4. Then, based on this study we directly present the guarantees on the *final approximate decoder* of eqs. (16) in Theorem 5.

Theorem 4. *Let M_r and M_c be such that (2) holds and $\gamma_r', \gamma_c' > 0$. Let also U^* and V^* be respectively the solutions of (12) and (13) with $\tilde{U} = M_r \bar{U} + \tilde{E}^u$ and $\tilde{V} = M_c \bar{V} + \tilde{E}^v$, where $\bar{u}_i \in span(P_{k_r})$, $i = 1, \cdots, p$, $\bar{v}_j \in span(Q_{k_c})$, $j = 1, \cdots, n$, $\tilde{E}^u \in \Re^{\rho_r \times \rho_r}$, $\tilde{E}^v \in \Re^{\rho_c \times \rho_c}$. We have:*

$$\|\bar{U}^* - \bar{U}\|_F \leq \sqrt{\frac{2p}{\rho_r(1-\delta)}} \left[\left(2 + \frac{1}{\sqrt{\gamma_r' \lambda_{k_r+1}}} \right) \|\tilde{E}^u\|_F + \left(\sqrt{\frac{\lambda_{k_r}}{\lambda_{k_r+1}}} + \sqrt{\gamma_r' \lambda_{k_r}} \right) \|\bar{U}\|_F \right], \text{ and}$$

$$\|E^*\|_F \leq \sqrt{\frac{2}{\gamma_r' \lambda_{k_r+1}}} \|\tilde{E}^u\|_F + \sqrt{2 \frac{\lambda_{k_r}}{\lambda_{k_r+1}}} \|\bar{U}\|_F.$$

where $\bar{U}^* = P_{k_r} P_{k_r}^\top X$ and $E^* = U^* - \bar{U}^*$. The same inequalities with slight modification also hold for V^*, which we omit because of space constraints.

Proof. Please refer to Appendix E. □

Theorem 5. *Let M_r and M_c be such that (2) holds. Let also U^* and V^* be the solutions of (16) with $\tilde{U} = M_r \bar{U}$ and $\tilde{V} = M_c \bar{V}$, where $\bar{u}_i \in span(P_{k_r})$, $i = 1, \cdots, p$, $\bar{v}_j \in span(Q_{k_c})$, $j = 1, \cdots, n$. We have:*

$$\|U^* - \bar{U}\|_F \leq \sqrt{\frac{2p}{\rho_r(1-\delta)}} \sqrt{\frac{\lambda_{k_r}}{\lambda_{k_r+1}}} \|\bar{U}\|_F$$

where $U^* = P_{k_r} P_{k_r}^\top X$. The same inequalities with slight modification also hold for V^*, which we omit because of space constraints.

Proof. The proof directly follows from the proof of Theorem 4 by using $\tilde{E}^u = 0$ and $\gamma_r^{'} = 0$. □

As $\bar{X} = \bar{U} \bar{\Sigma} \bar{V}^\top$, we can say that the error with eqs.(16) is upper bounded by the product of the errors of the individual subspace decoders. Also note that the error again depends on the spectral gaps defined by the ratios $\lambda_{k_c}/\lambda_{k_c+1}$ and $\lambda_{k_r}/\lambda_{k_r+1}$.

The solution to the above problems is simply a graph upsampling operation as explained in Lemma 1.

Lemma 1. *Let $S \in \Re^{c\times r}$ and $R \in \Re^{d\times r}$ be the two matrices such that $d < r$ and $d < c$. Furthermore, let $M \in \Re^{d\times c}$ be a sampling matrix as constructed in (1) and $\mathcal{L} \in \Re^{c\times c}$ be a symmetric positive semi-definite matrix. We can write $S = [S_a^\top | S_b^\top]^\top$, where $S_b \in \Re^{d\times r}$ and $S_a \in \Re^{(c-d)\times r}$ are the known and unknown submatrices of S. Then the exact and unique solution to the following problem:*

$$\min_{S_a} \operatorname{tr}(S^\top \mathcal{L} S), \quad s.t: \quad MS = R \tag{17}$$

is given by $S_a = -\mathcal{L}_{aa}^{-1}\mathcal{L}_{ab}R$.

Proof. Please refer to Appendix F. □

Using Lemma 1 and the notation of Section IV-A we can write:

$$U = \begin{bmatrix} -\mathcal{L}_r^{-1}(\bar{\Omega}_r, \bar{\Omega}_r)\mathcal{L}_r(\bar{\Omega}_r, \Omega_r)\tilde{U} \\ \tilde{U} \end{bmatrix}$$
$$V = \begin{bmatrix} -\mathcal{L}_c^{-1}(\bar{\Omega}_c, \bar{\Omega}_c)\mathcal{L}_c(\bar{\Omega}_c, \Omega_c)\tilde{V} \\ \tilde{V} \end{bmatrix}. \tag{18}$$

Eqs. (18) involves solving a sparse linear system. If each connected component of the graph has at least one labeled element, $\mathcal{L}_r(\bar{\Omega}_r, \bar{\Omega}_r)$ is full rank and invertible. If the linear system above is not large then one can directly use eq. (18). However, to avoid inverting the big matrix we can use the standard Preconditioned Conjugate Gradient (PCG) method to solve it. Note that the eqs. (18) and even PCG can be implemented in parallel for every column of U and V. This gives a significant advantage over the alternate decoder in terms of computation time. The cost of this decoder is $\mathcal{O}(O_l \mathcal{K} k n)$ where O_l is the number of iterations for the PCG method. The columns of U and V are not normalized with the above solution, therefore, a unit norm normalization step is needed at the end. Once U, V are determined, one can use $X = U\tilde{\Sigma}V^\top \sqrt{np/\rho_r\rho_c(1-\delta)}$ to determine the required low-rank matrix X. The decoder for approximate recovery is presented in Algorithm 2.

Algorithm 2 Subspace Upsampling based Approximate Decoder for low-rank recovery

INPUT: $\tilde{X} \in \Re^{\rho_c\times\rho_r}$, $\mathcal{L}_r \in \Re^{p\times p}$, $\mathcal{L}_c \in \Re^{n\times n}$
1. do $SVD(\tilde{X}) = \tilde{U}\tilde{\Sigma}\tilde{V}^\top$
2. find k such that $\tilde{\Sigma}_{k,k}/\tilde{\Sigma}_{1,1} < 0.1$
3. Solve eqs. (16) for every column of U, V as following:
for $i = 1, \dots k$ **do**
solve $\min_{u_i} u_i^\top \mathcal{L}_r u_i$ s.t $M_r u_i = \tilde{u}_i$ using PCG
solve $\min_{v_i} v_i^\top \mathcal{L}_c v_i$ s.t $M_c^\top v = \tilde{v}_i$ using PCG
end for
4. Set $u_i = u_i/\|u_i\|_F, v_i = v_i/\|v_i\|_F, \forall i = 1, \cdots, k$
5. Set $\Sigma = \sqrt{\frac{np}{\rho_r\rho_c(1-\delta)}}\tilde{\Sigma}$
6. Set $X = U\Sigma V^\top$
OUTPUT: The full low-rank $X \in \Re^{p\times n}$

Two other approximate decoders for low-rank recovery are presented in Appendix G.

VI. Decoder for clustering

As already mentioned earlier, PCA has been widely used for two types of applications: 1) low-rank recovery and 2) clustering. Therefore, in this section, we present a method to perform clustering using our framework.

For the clustering application we do not need the full low-rank matrix X. Thus, we propose to do k-means on the low-rank representation of the sampled data $\tilde{X}$ obtained using (4), extract the cluster labels $\tilde{C}$ and then decode the cluster labels C for X on the graphs $\mathcal{L}_r$ and $\mathcal{L}_c$.

Let $\tilde{C} \in \{0,1\}^{\rho_c\times k}$ be the cluster labels of $\tilde{X}$ (for k clusters) which are obtained by performing k-means. Then,

$$\tilde{C}_{ij} = \begin{cases} 1 & \text{if } \tilde{x}_i \in \ j^{th} \text{ cluster} \\ 0 & \text{otherwise.} \end{cases}$$

Note that each of the columns $\tilde{c}_i$ of $\tilde{C}$ is the cluster indicator for one of the k clusters. The goal now is to decode the cluster indicator matrix $C \in \{0,1\}^{n\times k}$. We refer to the Compressive Spectral Clustering (CSC) framework [42], where the authors solve a similar problem by arguing that each of the columns of C can be obtained by assuming that it lies close to the $span(Q_{k_c})$, where Q_{k_c} are the first k_c Laplacian eigenvectors of the graph G_c. This requires solving the following convex minimization problem:

$$\min_C \|M_c^\top C - \tilde{C}\|_F^2 + \gamma \operatorname{tr}(C^\top \mathcal{L}_c C) \tag{19}$$

The above problem can be solved independently for each of the columns of C, thus,

$$\min_{c_i} \|M_c^\top c_i - \tilde{c}_i\|_2^2 + \gamma c_i^\top \mathcal{L}_c c_i \tag{20}$$

Furthermore, note that the graph G_r is not required for this process. Eq.(20) gives a faithful solution for c_i if the sampling operator M_c satisfies the restricted isometry property RIP. Thus, for any $\delta_c, \epsilon_c \in (0,1)$, with probability at least $1-\epsilon_c$,

$$(1-\delta_c)\|w\|_2^2 \le \frac{n}{\rho_c}\|w^\top M_c\|_2^2 \le (1+\delta_c)\|w\|_2^2 \tag{21}$$

for all $w \in span(Q_{k_c})$ provided that

$$\rho_c \ge \frac{3}{\delta_c^2}\nu_{k_c}^2 \log\left(\frac{2k_c}{\epsilon_c}\right). \tag{22}$$

This holds true as a consequence of Theorem 1 (eq.(30) in the proof of Theorem 1 and Theorem 5 in [28]).

Eq.(20) requires the tuning of a model parameter γ which we want to avoid. Therefore, we use the same strategy as for the approximate low-rank decoder in Section V-C. The cluster labels $\tilde{c}_i$ are not noisy because they are obtained by running k-means on the result of FRPCAG eq.(4), which is robust to outliers. Thus, we set $\gamma = 0$ in eq.(20) and propose to solve the following problem:

$$\min_{c_i} c_i^\top \mathcal{L}_c c_i \quad \text{s.t:} \quad M_c^\top c_i = \tilde{c}_i. \tag{23}$$

According to Lemma 1, the solution is given by:

$$c_i = \begin{bmatrix} -\mathcal{L}_c^{-1}(\bar{\Omega}_c, \bar{\Omega}_c)\mathcal{L}_c(\bar{\Omega}_c, \Omega_c)\tilde{c}_i \\ \tilde{c}_i \end{bmatrix}. \tag{24}$$

Ideally, every row of the matrix C should have 1 in exactly one of the k columns, indicating the cluster membership of that data sample. However, the solution $C \in \mathbb{R}^{n\times k}$ obtained by solving the above problem is not binary. Thus, to finalize the cluster membership (one of the k columns), we perform a maximum pooling for each of the rows of C, i.e,

$$C_{ij} \leftarrow \begin{cases} 1 & \text{if } C_{ij} = \max\{C_{ij} \ \forall \ j = 1 \cdots k\} \\ 0 & \text{otherwise.} \end{cases}$$

Algorithm 3 summarizes this procedure.

Algorithm 3 Approximate Decoder for clustering

INPUT: $\tilde{X} \in \Re^{\rho_c \times \rho_r}$, $\mathcal{L}_c \in \Re^{n\times n}$
1. do k-means on $\tilde{X}$ to get the labels $\tilde{C} \in \{0,1\}^{\rho_c \times k}$
2. Solve eqs. (23) for every column of C as following:
for $i = 1, \ldots k$ **do**
solve $\min_{c_i} c_i^\top \mathcal{L}_c c_i$ s.t $M_c^\top c_i = \tilde{c}_i$ using PCG
end for
3. Set $C_{ij} = 1$ if $\max\{C_{ij} \ \forall \ j = 1 \cdots k\}$ and 0 otherwise.
OUTPUT: cluster indicators for X: $C \in \{0,1\}^{n\times k}$

Theorem 6. *Let M_c be such that (21) holds. Let also c_i^* be the solution of (23) with $\tilde{c}_i = M_c^\top \bar{c}_i$, where $\bar{c}_i \in span(Q_{k_c})$, $i = 1, \cdots, n$. We have:*

$$\|{c_i}^* - \bar{c}_i\|_2 \leq \sqrt{\frac{n}{\rho_c(1-\delta_c)}}\sqrt{\frac{\lambda_{k_c}}{\lambda_{k_c+1}}}\|\bar{c}_i\|_2$$

where $c_i^* = Q_{k_c} Q_{k_c}^\top c_i$.

Proof. The proof directly follows from the proof of Theorem 3.2 in [28]. These steps have been repeated in the proof of Theorem 4 in Appendix E as well. Using $c_i^* = \bar{u}_i^*$, $\bar{c}_i = \bar{u}_i$, $n = p, \rho_c = \rho_r$ in eq.(41) one can get theoretical guarantees for eq. (20). Then, by using $\tilde{e}_i^u = 0$ and $\gamma = 0$ we get the result of above theorem. □

VII. Complexity, Memory & Convergence

Computational Complexity: A summary of all the decoders and their computational complexities is presented in Table X of Appendix H. The complete CPCA algorithm and the computational complexities of different steps are presented in Table I. For $\mathcal{K}, k, \rho_r, \rho_c, p \ll n$ CPCA algorithm scales as $O(nk\mathcal{K})$ per iteration. Thus, assuming that the row and column graphs are available from external source, a speed-up of p/k per iteration is obtained over FRPCAG and p^2/k over RPCA. A detailed explanation regarding the calculation of complexities of CPCA and other models is presented in Table XI and Appendix H.

Memory Requirements: We compare the memory requirements of CPCA with FRPCAG. For a matrix $Y \in \mathbb{R}^{p\times n}$, FRPCAG and CPCA require the construction of two graphs G_r, G_c whose Laplacians $L_r \in \mathbb{R}^{p\times p}, L_c \in \mathbb{R}^{n\times n}$ are used in the core algorithm. However, these Laplacians are sparse, therefore the memory requirement for L_r, L_c is $\mathcal{O}(\mathcal{K}(|\mathcal{E}_r| + |\mathcal{E}_c|))$ respectively. The core algorithm of FRPCAG requires operation on the full matrix Y and the graph Laplacians L_r, L_c. As $|\mathcal{E}_r| \approx \mathcal{K}p$ and $|\mathcal{E}_c| \approx \mathcal{K}n$ therefore, the memory requirement for the regularization terms $\mathrm{tr}(XL_cX^\top)$ and $\mathrm{tr}(X^\top L_r X)$ is $\mathcal{O}(\mathcal{K}np)$. For the CPCA algorithm, assuming $n > p$ and letting $\rho_r = p/b$ and $\rho_c = n/a$, the complexity of FRPCAG on the sampled data is $\mathcal{O}(\mathcal{K}np/(ab))$ and the approximate decode stage for subspaces of dimension k is $\mathcal{O}(\mathcal{K}nk)$. Thus the overall memory requirement of CPCA is $\mathcal{O}(\mathcal{K}n(p/(ab) + k))$. As compared to FRPCAG, an improvement of $pab/(p + kab)$ is obtained. For example for $n = 1000, p = 200, a = 10, b = 1, k = 10$, a reduction of approximately 6.6 times is obtained.

Convergence of CPCA: The CPCA based algorithm (Table I) has two main steps: 1) FRPCAG on the compressed data matrix and 2) low-rank matrix or cluster label decoding. FRPCAG is solved by the FISTA (Algorithm 1) and the decode step is solved using the PCG method. Both of these methods have been well studied in terms of their convergence guarantees. More specifically, one can refer to [3] for a detailed study on FISTA and [2] for PCG. Let $\tilde{X}_j$ represent the iterates of the FRPCAG problem 4, $\tilde{X}^*$ the optimal solution, $F(\tilde{X})$ the objective function and $2\gamma_r\|\tilde{\mathcal{L}}_r\|_2 + 2\gamma_c\|\tilde{\mathcal{L}}_c\|_2$ the Lipshitz constant, then the Algorithm 1 (FISTA) converges as following:

$$F(\tilde{X}_j) - F(\tilde{X}^*) \leq \frac{4\alpha(\gamma_r\|\tilde{\mathcal{L}}_r\|_2 + \gamma_c\|\tilde{\mathcal{L}}_c\|_2)\|\tilde{X}_j - \tilde{X}^*\|_F^2}{(j+1)^2},$$

where α is the step size and j denotes the iteration number. The PCG method on the other hand can theoretically be viewed as a direct method, as it produces the exact solution after a finite number of iterations [2].

Tables VI & IX show detailed computational time and iterations for our CPCA based algorithms to converge, for a wide variety and sizes of datasets.

VIII. Experimental Results

We perform two types of experiments corresponding to two applications of PCA 1) Data clustering and 2) Low-rank recovery using two open-source toolboxes: the UNLocBoX [26] and the GSPBox [25].

A. Clustering

1) Experimental Setup: **Datasets**: We perform our clustering experiments on 5 benchmark databases (as in [35], [36]): CMU PIE, ORL, YALE, MNIST and USPS. For the USPS and ORL datasets, we further run two types of experiments 1) on subset of datasets and 2) on full datasets. The experiments on the subsets of the datasets take less time so they are used to show the efficiency of our model for a wide variety of noise types. The details of all datasets used are provided in Table XII of Appendix H.

Noise & Errors: CPCA is a memory and computationally efficient alternative for FRPCAG. An important property of

Table I: Summary of CPCA and its computational complexity for a dataset $Y \in \Re^{p \times n}$. Throughout we assume that $\mathcal{K}, k, \rho_r, \rho_c, p \ll n$.

Steps	The Complete CPCA Algorithm	Complexity
1	Construct graph Laplacians between the rows $\mathcal{L}_r$ and columns $\mathcal{L}_c$ of Y using Section II.	$\mathcal{O}(np\log(n))$
2	Construct row and column sampling matrices $M_r \in \Re^{\rho_r \times p}$ and $M_c \in \Re^{n \times \rho_c}$ satisfying (1) and theorem 1	–
3	Sample the data matrix Y as $\tilde{Y} = M_r Y M_c$	–
4	Construct the new graph Laplacians between the rows $\tilde{\mathcal{L}}_r$ and columns $\tilde{\mathcal{L}}_c$ of $\tilde{Y}$ using Section IV-A.	$\mathcal{O}(O_l \mathcal{K} n)$
5	Solve FRPCAG (4) using Algorithm 1 to get the low-rank $\tilde{X}$	$\mathcal{O}(I \rho_r \rho_c \mathcal{K})$
6	**For low-rank recovery**: Decode X from $\tilde{X}$ using the approximate decoder Algorithm 2	$\mathcal{O}(O_l n k \mathcal{K})$
7	**For clustering**: Decode the cluster labels C for X using the semi-supervised label propagation (Algorithm 3)	$\mathcal{O}(O_l n k)$

Table II: Clustering error of USPS datasets for different PCA based models. The best results per column are highlighted in bold and the 2nd best in blue. NMF and GNMF require non-negative data so they are not evaluated for USPS because USPS is also negative.

Dataset	Model	no noise	Gaussian noise				Laplacian noise				Sparse noise			
			5%	10%	15%	20%	5%	10%	15%	20%	5%	10%	15%	20%
USPS small ($n = 3500$ $p = 256$)	k-means	0.31	0.31	0.31	0.33	0.32	0.32	0.30	0.36	0.37	0.40	0.45	0.53	0.73
	LLE	0.40	0.34	0.32	0.35	0.24	0.40	0.40	0.33	0.36	0.23	0.30	0.33	0.37
	LE	0.38	0.38	0.38	0.36	0.35	0.38	0.38	0.38	0.38	0.32	0.33	0.36	0.48
	PCA	0.27	0.29	0.25	0.28	0.26	0.29	0.29	0.28	0.24	0.29	0.26	0.26	0.28
	MMF	0.21	**0.20**	0.21	0.22	0.21	0.21	0.21	0.22	0.21	0.27	0.23	0.25	0.27
	GLPCA	**0.20**	**0.20**	0.21	0.23	0.23	0.21	0.21	0.22	0.21	0.26	0.24	0.24	0.28
	RPCA	0.26	0.25	0.23	0.24	0.22	0.26	0.26	0.25	0.24	0.26	0.24	0.23	0.30
	RPCAG	**0.20**	**0.20**	0.21	0.20	0.21	**0.20**	0.21	0.21	0.21	**0.21**	**0.22**	0.23	0.25
	FRPCAG	**0.20**	**0.20**	**0.20**	**0.19**	**0.20**	**0.20**	**0.19**	**0.17**	**0.17**	**0.21**	**0.22**	**0.22**	**0.23**
	CPCA (2,1)	**0.20**	**0.20**	0.21	0.20	0.22	**0.20**	0.22	0.22	0.21	0.23	0.23	0.25	0.28
USPS large ($n = 10000$ $p = 256$)	k-means	0.26	0.26	0.26	0.26	0.28	0.27	0.26	0.26	0.26	0.26	0.25	0.34	0.30
	LLE	0.51	0.29	0.22	0.21	0.22	0.22	0.22	0.22	0.21	0.22	**0.19**	0.26	0.31
	LE	0.33	0.32	0.32	0.27	0.27	0.32	0.34	0.31	0.34	0.35	0.44	0.49	0.53
	PCA	0.21	0.21	0.21	0.22	0.21	0.21	0.21	0.22	0.21	0.22	0.23	0.23	0.23
	MMF	0.24	0.23	0.23	0.24	0.24	0.19	0.23	0.22	0.23	0.24	0.25	0.26	0.26
	GLPCA	0.16	0.16	0.17	0.17	0.16	0.17	0.15	0.15	0.17	**0.18**	**0.19**	**0.21**	0.23
	FRPCAG	**0.15**	0.16	0.17	**0.15**	**0.14**	0.16	0.16	0.16	0.17	**0.18**	0.21	**0.21**	**0.21**
	CPCA (10,1)	**0.15**	**0.14**	**0.14**	**0.15**	**0.14**	**0.14**	**0.14**	**0.13**	**0.14**	**0.18**	**0.19**	0.22	0.24
MNIST small (n = 1000 p = 784)	K-means	0.51	0.51	0.52	0.51	0.58	0.52	0.51	0.52	0.52	0.80	0.88	0.88	0.88
	PCA	0.43	0.43	0.41	0.43	0.42	0.43	0.42	0.38	0.43	0.42	0.42	0.42	0.44
	MMF	0.33	0.34	0.34	0.34	**0.33**	0.34	**0.33**	0.37	0.34	0.40	0.38	0.38	0.42
	GLPCA	0.38	0.36	0.37	0.35	0.38	0.39	0.36	0.36	0.36	0.37	0.39	0.38	**0.39**
	PCAG-(1,0)	0.40	0.40	0.41	0.40	0.40	0.40	0.40	0.41	0.40	0.37	0.37	0.39	0.44
	FRPCAG	**0.32**	**0.33**	**0.32**	**0.33**	**0.33**	**0.32**	**0.33**	**0.32**	**0.32**	**0.33**	**0.36**	**0.35**	**0.39**
	CPCA (5,1)	0.39	0.38	0.37	0.38	0.38	0.39	0.39	0.36	0.37	0.39	0.40	0.41	0.50

FRPCAG is its robustness to noise and outliers, just like RPCA. Therefore, it is important to study the performance of CPCA under noise and corruptions similar to those for FRPCAG and RPCA. To do so we add 3 different types of noise in all the samples of datasets in different experiments: 1) Gaussian noise and 2) Laplacian noise with standard deviation ranging from 5% to 20% of the original data 3) Sparse noise (randomly corrupted pixels) occupying 5% to 20% of each data sample.

Comparison with other methods: We compare the clustering performance of CPCA with 11 other models including: 1) k-means on original data 2) Laplacian Eigenmaps (LE) [4] 3) Locally Linear Embedding (LLE) [32] 4) Standard PCA 5) Graph Laplacian PCA (GLPCA) [14] 6) Manifold Regularized Matrix Factorization (MMF) [46] 7) Non-negative Matrix Factorization (NMF) [17] 8) Graph Regularized Non-negative Matrix Factorization (GNMF) [6] 9) Robust PCA (RPCA) [7] 10) Robust PCA on Graphs (RPCAG) [35] and 11) Fast Robust PCA on Graphs (FRPCAG) [36]. RPCA and RPCAG are not used for the evaluation of MNIST, USPS large and ORL large datasets due to computational complexity of these models.

Pre-processing: All datasets are transformed to zero-mean and unit standard deviation along the features / rows. For MMF the samples are additionally normalized to unit-norm. For NMF and GNMF only the unit-norm normalization is applied to all the samples of the dataset as NMF based models can only work with non-negative data.

Evaluation Metric: We use *clustering error* as a metric to compare the clustering performance of various models. The clustering error for LE, PCA, GLPCA, MMF, NMF and GNMF is evaluated by performing k-means on the principal components V (note that these models explicitly learn V, where $X = U\Sigma V^\top$). The clustering error for RPCA, RPCAG and FRPCAG is determined by performing k-means directly on the low-rank X. For our CPCA method, k-means is performed on the small low-rank $\tilde{X}$ and then the labels for full X are decoded using Algorithm 3.

Parameter Selection: To perform a fair validation for each of the models we use a range of values for the model

Table III: Clustering error of ORL datasets for different PCA based models. The best results per column are highlighted in bold and the 2nd best in blue.

Data set	Model	no noise	Gaussian noise			
			5%	10%	15%	20%
ORL small	k-means	0.40	0.43	0.43	0.45	0.44
	LLE	0.26	0.26	0.26	0.26	0.19
	LE	0.21	0.18	0.19	0.19	0.19
	PCA	0.30	0.30	0.32	0.34	0.32
	MMF	0.21	0.20	0.18	0.18	0.17
	GLPCA	**0.14**	**0.13**	**0.13**	**0.13**	**0.14**
	NMF	0.31	0.34	0.29	0.31	0.34
	GNMF	0.29	0.29	0.29	0.31	0.29
	RPCA	0.36	0.34	0.33	0.35	0.36
	RPCAG	0.17	0.17	0.17	0.16	0.16
	FRPCAG	0.15	0.15	0.15	0.14	0.16
	CPCA (2,2)	0.23	0.23	0.23	0.24	0.25
	CPCA (1,2)	0.17	0.17	0.17	0.14	0.17
ORL large	k-means	0.49	0.50	0.51	0.51	0.51
	LLE	0.28	0.27	0.27	0.24	0.25
	LE	0.24	0.25	0.25	0.24	0.25
	PCA	0.35	0.34	0.35	0.36	0.36
	MMF	0.23	0.23	0.23	0.24	0.24
	GLPCA	0.18	0.18	0.18	0.19	0.19
	NMF	0.36	0.33	0.36	0.32	0.36
	GNMF	0.34	0.37	0.36	0.39	0.39
	FRPCAG	**0.17**	**0.17**	**0.17**	**0.17**	**0.17**
	CPCA (2,2)	0.21	0.21	0.21	0.23	0.22

Table IV: Clustering error of CMU PIE and YALE datasets for different PCA based models. The best results per column are highlighted in bold and the 2nd best in blue.

Data set	Model	no noise	Gaussian noise			
			5%	10%	15%	20%
CMU PIE	k-means	0.76	0.76	0.76	0.75	0.76
	LLE	0.47	0.50	0.52	0.50	0.55
	LE	0.60	0.58	0.59	0.58	0.60
	PCA	0.27	0.27	0.27	0.27	0.29
	MMF	0.67	0.67	0.67	0.66	0.67
	GLPCA	0.37	0.39	0.37	0.38	0.38
	NMF	0.24	0.27	**0.24**	0.25	0.27
	GNMF	0.58	0.59	0.56	0.58	0.59
	RPCA	0.39	0.38	0.41	0.41	0.38
	RPCAG	0.24	**0.24**	**0.24**	**0.24**	0.25
	FRPCAG	**0.23**	**0.24**	**0.24**	**0.24**	**0.24**
	CPCA (2,1)	0.26	0.26	0.26	0.28	0.27
	CPCA (2,2)	0.28	0.29	0.29	0.30	0.30
YALE	k-means	0.76	0.76	0.76	0.76	0.77
	LLE	0.51	0.47	0.46	0.49	0.50
	LE	0.52	0.56	0.55	0.54	0.54
	PCA	0.53	0.52	0.53	0.56	0.55
	MMF	0.58	0.59	0.56	0.57	0.58
	GLPCA	0.47	0.46	0.47	0.45	0.48
	NMF	0.57	0.58	0.59	0.57	0.59
	GNMF	0.59	0.59	0.61	0.59	0.60
	RPCA	0.45	0.48	0.46	0.46	0.48
	RPCAG	**0.40**	0.40	0.41	0.41	0.41
	FRPCAG	**0.40**	**0.37**	**0.40**	**0.40**	**0.40**
	CPCA (2,2)	0.43	0.45	0.42	0.46	0.43

parameters as presented in Table XIII of Appendix H. For a given dataset, each of the models is run for each of the parameter tuples in this table and the best clustering error is reported. Furthermore, PCA, GLPCA, MMF, NMF and GNMF are non-convex models so they are run 10 times for each of the parameter tuple. RPCA, RPCAG, FRPCAG and CPCA based models are convex so they are run only once. Although our CPCA approach is convex, it involves a bit of randomness due to the sampling step. Due to the extensive nature of the experimental setup, most of the clustering experiments are performed under one sampling condition. However, it is interesting to study the variation of error under different sampling scenarios. For this purpose we perform some additional experiments on the USPS dataset. For our proposed CPCA, we use a convention $CPCA(a, b)$, where a and b denote the downsampling factors on the columns and rows respectively. A uniform sampling strategy is always used for CPCA.

Graph Construction: The $\mathcal{K}$-nearest neighbor graphs G_r, G_c are constructed using FLANN [22] as discussed in Section II. The small graphs $\tilde{G}_r, \tilde{G}_c$ can also be constructed using FLANN or the Kron reduction strategy of Section IV-A. For all the experiments reported in this paper we use $\mathcal{K}$-nearest neighbors = 10 and Gaussian kernel for the adjacency matrices W. The smoothing parameters σ^2 for the Gaussian kernels are automatically set to the average distance of the $\mathcal{K}$-nearest neighbors.

*2) **Discussion on clustering performance**:* We point out here that the purpose of our clustering experiments is three-fold:

- To show the efficiency of CPCA for a wide variety of noise and errors and downsampling.
- To study the conditions under which CPCA performs worse than the other models.
- To study the variation of performance under different sampling scenarios.

For this purpose, we test CPCA under a variety of downsampling for different datasets. Cases with $p \ll n$ and $n \ll p$ carry special interest. Therefore, we present our discussion below in the light of the above goals.

Tables II, III, IV & V present the clustering results for USPS small, USPS large, MNIST large, MNIST small, ORL small, ORL large, CMU PIE and YALE datasets. Note that not all the models are run for all the datasets due to computational constraints. The best results are highlighted in bold and the second best in blue. From Table II for the USPS dataset, it is clear that our proposed CPCA model attains comparable clustering results to the state-of-the-art RPCAG and FRPCAG models and better than the others in most of the cases. Similar observation can be made about the MNIST large dataset from Table V in comparison to FRPCAG.

It is important to note that for the USPS and MNIST datasets $p \ll n$. Thus, for the USPS dataset, the compression is only applied along the columns (n) of the dataset. This compression results in clustering error which is comparable to the other state-of-the-art algorithms. As $p = 256$ for the USPS dataset, it was observed that even a 2 times downsampling on p results in a loss of information and the clustering quality deteriorates. The same observation can be made about ORL small, ORL large and YALE datasets from Tables III, IV for CPCA (2,2). i.e, two times downsampling on both rows and columns. On the other hand the performance of CPCA (1,2) is reasonable for the ORL small dataset. Recall that CPCA (a,b) means a downsampling by a and b across columns and rows (samples and features). Also note that for ORL dataset $n \ll p$.

Finally, we comment about the results on MNIST small dataset ($p = 784, n = 1000$) from Table II. It is clear that FRPCAG (no compression) results in the best performance. CPCA (5,1) results in a highly undersampled dataset which does not capture enough variations in the MNIST small dataset to deliver a good clustering performance. This particular case supports the fact that compression does not always yield a good performance at the advantage of reduced complexity. Therefore, we study this phenomena below.

The above findings for the MNIST dataset are intuitive as it only makes sense to compress both rows and columns in our CPCA based framework if a reasonable speed-up can be obtained without compromising the performance, i.e, *if both n and p are large*. If either n or p is small then one might only apply compression along the larger dimension, as the compression on the smaller dimension would not speed up the computations significantly. For example, for the USPS dataset, a speed-up of $p/k = 256/10 \approx 25$ times would be obtained over FRPCAG by compressing along the samples (columns n) only without a loss of clustering quality. Our experiments showed that this speed up increased upto 30 by compressing along the features but with a loss of performance (The results are not presented for brevity).

Tables II, III, IV & V also show that CPCA is quite robust to a variety of noise and errors in the dataset. Even in the presence of higher levels of Gaussian and Laplacian noise, CPCA performs comparable to other methods for the USPS dataset. Thus, CPCA tends to preserve the robustness property of FRPCAG. This will also be clear from the low-rank recovery experiments in the next section.

*3) **Computation Time vs Performance & Sampling**:* It is interesting to compare 1) the time needed for FRPCAG and CPCA to perform clustering 2) the corresponding clustering error and 3) the sub-sampling rates in CPCA. Table V shows such a comparison for 70,000 digits of MNIST with (10, 2) times downsampling on the (columns, rows) respectively for CPCA. The time needed by CPCA is an order of magnitude lower than FRPCAG. Note that the time reported here does not include the construction of graphs G_r, G_c as both methods use the same graphs. Furthermore, these graphs can be constructed in the order of a few seconds if parallel processing is used. The time for CPCA includes steps 2 to 5 and 7 of Table I. For the information about the graph construction time, please refer to Table VI and the discussion thereof.

Table VI presents the computational time and number of iterations for the convergence of CPCA, FRPCAG, RPCAG & RPCA on different sizes and dimensions of the datasets. We also present the time needed for the graph construction. The computation is done on a single core machine with a 3.3 GHz processor without using any distributed or parallel computing tricks. An ∞ in the table indicates that the algorithm did not converge in 4 hours. It is notable that our model requires a very small number of iterations to converge irrespective of the size of the dataset. Furthermore, the model is orders of magnitude faster than RPCA and RPCAG. This is clearly observed from the experiments on MNIST dataset where our proposed model is 100 times faster than RPCAG. Specially for MNIST dataset with 25000 samples, RPCAG and RPCA did not converge even in 4 hours whereas CPCA converged in less than a minute.

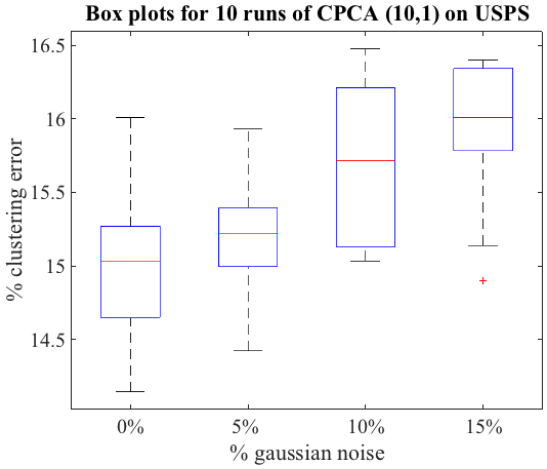

Figure 2: Box plots for clustering error over 10 random sampling runs of CPCA (10,1) for the full USPS dataset (256×10000) with increasing levels of Gaussian noise. For each run CPCA is evaluated for the full parameter grid $\gamma_r^{'}, \gamma_c^{'} \in (0, 30)$ and the minimum clustering error is considered. A slight increase in the average clustering error with Gaussian noise shows that CPCA is quite robust to sampling and noise.

Table V: Clustering error and computational times of FRPCAG and CPCA on MNIST large dataset ($784 \times 70{,}000$).

Model	**FRPCAG**	**CPCA (10,2)**
Error	0.25	0.24
time (secs)	350	58

*4) **Effect of random sampling**:* An interesting observation can be made from Table V for the MNIST dataset: the error of CPCA is also lower than FRPCAG. Such cases can also be observed in USPS dataset (Table II). As the downsampling step is random, it might remove some spurious samples sometimes and the clustering scheme (Section VI) becomes robust to these samples. For the clustering application, the spurious samples mostly lie on the cluster borders and deteriorate the clustering decision.

To incorporate the extensive nature of the experiments presented in this work, Tables II & V correspond to one run of the CPCA for one specific sampling case. However, it is imperative to study the effect of random sampling on the performance of CPCA. Therefore, in order to study the effect of random sampling on the clustering performance, we perform an experiment on the full USPS dataset (256×10000) with different levels of artificial noise. The results in Fig. 2 correspond to 10 runs of the CPCA under different uniform sampling scenarios. For this experiment, we downsample 10 times along the columns (digits), whereas no downsampling is used across the features and then add different levels of Gaussian noise from 0 to 15% in the dataset. Thus, we downsample the dataset, run FRPCAG to get a low-rank $\tilde{X} \in \mathbb{R}^{256 \times 1000}$, perform k-means ($k = 10$) and then use

Table VI: Computation times (in seconds) for graphs G_r, G_c, FRPCAG, CPCA, RPCAG, RPCA and the number of iterations to converge for different datasets. The computation is done on a single core machine with a 3.3 GHz processor without using any distributed or parallel computing tricks. ∞ indicates that the algorithm did not converge in 4 hours.

Dataset	Samples	Features	Classes	Graphs		FRPCAG		CPCA			RPCAG		RPCA	
				G_r	G_c	time	Iters	(a,b)	time	Iters	time	Iters	time	Iters
MNIST	5000	784	10	10.8	4.3	13.7	27	(5,1)	5	30	1345	325	1090	378
MNIST	15000	784	10	32.5	13.3	35.4	23	(5,1)	13	25	3801	412	3400	323
MNIST	25000	784	10	40.7	22.2	58.6	24	(10,1)	20	37	∞	∞	∞	∞
ORL	300	10304	30	1.8	56.4	24.7	12	(2,1)	14	15	360	301	240	320
USPS	3500	256	10	5.8	10.8	21.7	16	(10,1)	12	31	900	410	790	350

Table VII: Preservation of the rank of the datasets in the compressed low-rank $\tilde{X}$ determined by solving FRPCAG (4).

Dataset	downsampling factor	actual rank	Rank after FRPCAG on sampled matrix					
	$(columns, rows)$		Gaussian noise			Laplacian noise		
			5%	10%	15%	5%	10%	15%
ORL large	(2,1)	40	41	41	41	41	41	42
USPS large	(10,2)	10	10	11	11	11	11	11
MNIST	(10,2)	10	11	11	11	11	11	11
CMU PIE	(2,1)	30	31	31	31	31	31	32
YALE	(2,1)	11	11	11	11	11	11	11

the approximate clustering decoder (Algorithm 3) to decode the labels for the full dataset. This process is repeated over the whole parameter grid $\gamma_r^{'}, \gamma_c^{'} \in (0, 30)$ and the minimum error over the grid is considered. Each of the boxplots in this figure summarize the clustering error over 10 runs of CPCA(10,1) for different levels of Gaussian noise. The mean clustering error is 15.05% for the case of no noise and 15.2%, 15.6% and 16% for 5%, 10% and 15% Gaussian noise respectively. Furthermore, the standard deviation for each of the boxplots varies between 0.4% to 0.55%. This result clearly shows that the CPCA performance is quit robust to random sampling. Similar results were observed for other datasets and are not reported here for the purpose of brevity.

It is interesting to study the reduction in the total time attained by using CPCA as compared to FRPCAG. Table VI can be used to perform this comparison as well. For example, for the MNIST dataset with 5000 samples, the total time (including graph construction) for FRPCAG is 28.2 secs and that for CPCA is 20.1 secs. Thus, a speed-up of 1.4 times is obtained over FRPCAG. The time required for the construction of the graph between the samples or features is often more than that required for the CPCA to converge. This is a small computational bottleneck of the graph construction algorithm. While graph learning or graph construction is an active and interesting field of research, it is not a part of this work. The state-of-the-art results [16], [24], [34] do not provide any scalable solutions yet.

*5) **Rank Preservation Under Downsampling**:* An interesting question about CPCA is if it preserves the underlying rank of the dataset under the proposed sampling scheme. Table VII shows that the rank is preserved even in the presence of noise. For this experiment, we take different datasets and corrupt them with different types of noise and perform cross-validation for clustering using the parameter range for CPCA mentioned in Table XIII (see Appendices). Then, we report the rank of $\tilde{X}$ for the parameter corresponding to the minimum clustering error. As $\tilde{X}$ is approximately low-rank so we use the following strategy to determine the rank k: $\tilde{\Sigma}_{k,k}/\tilde{\Sigma}_{1,1} < 0.1$. FRPCAG assumes that the number of clusters $\approx$ rank of the dataset. Our findings show that this assumption is almost satisfied for the sampled matrices even in the presence of various types of noise. Thus, the rank is preserved under the proposed sampling strategy. For clustering experiments, the lowest error with CPCA occurs when the rank $\approx$ number of clusters.

Table VIII: Variation of clustering error of CPCA with different uniform downsampling schemes / factors across rows and columns of the USPS dataset ($256 \times 10,000$).

downsampling (rows / cols)	1	5	10	15	20
1	0.16	0.16	0.21	0.21	0.21
2	0.21	0.23	0.23	0.24	0.25
4	0.26	0.30	0.31	0.31	0.31

*6) **Clustering error vs downsampling rate**:* Table VIII shows the variation of clustering error of CPCA with different downsampling factors across rows and columns of the USPS dataset ($256 \times 10,000$). Obviously, higher downsampling results in an increase in the clustering error. However, note that we can downsample the samples (columns) by a factor of 5 without observing an error increase. The downsampling of features results in an error increase because the number of features for this dataset is only 256 and downsampling results in a loss of information. Similar behavior can also be observed for the ORL small and ORL large datasets in Table III where the performance of CPCA is slightly worse than FRPCAG because the number of samples n for ORL is only 400.

B. Low-rank recovery

In order to demonstrate the effectiveness of our model to recover low-rank static background from videos we perform experiments on 1000 frames of 3 videos available online [7].

[7] https://sites.google.com/site/backgroundsubtraction/test-sequences

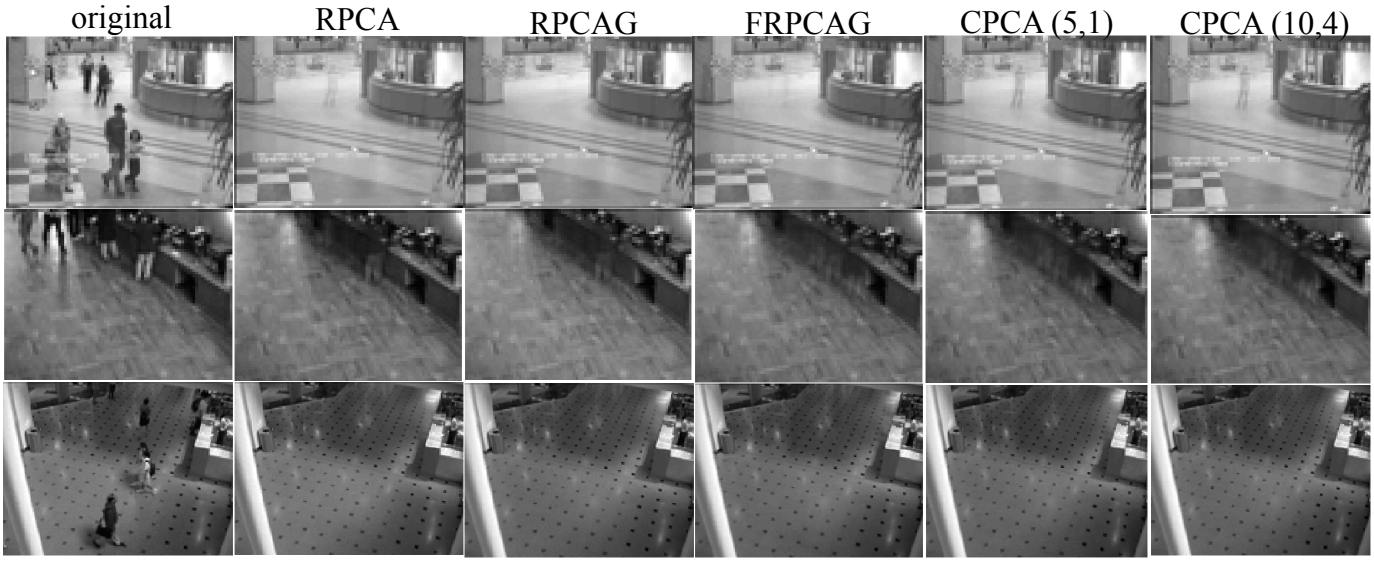

Figure 3: Static background separation from videos using different PCA based models. The first row corresponds to a frame from the video of a shopping mall lobby, the second row to a restaurant food counter and the third row to an airport lobby. The leftmost plot in each row shows the actual frame, the other 5 show the recovered low-rank using RPCA, RPCAG, FRPCAG and CPCA with two different uniform downsampling schemes.

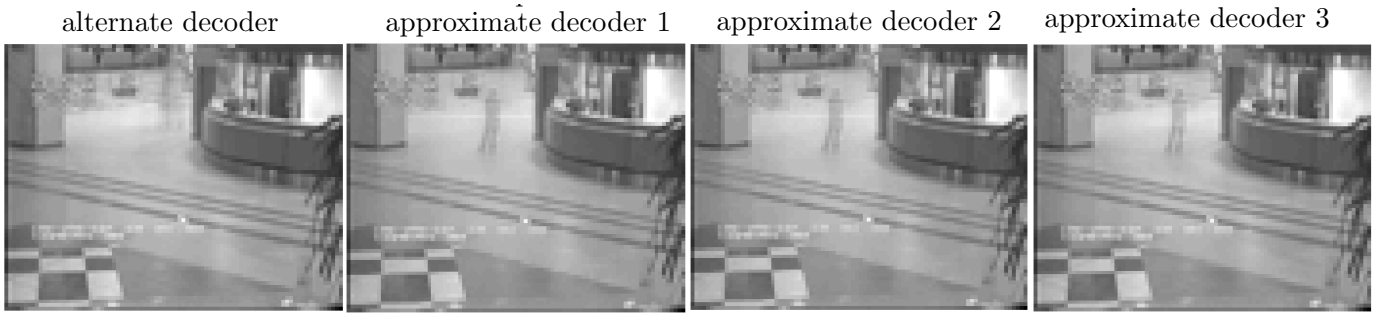

Figure 4: A quality comparison of various low-rank decoders discussed in this work.

All the frames are vectorized and arranged in a data matrix Y whose columns correspond to frames. The graph G_c is constructed between the columns of Y and the graph G_r is constructed between the rows of Y following the methodology of Section II. Fig. 3 shows the recovery of low-rank frames for one actual frame of each of the videos. The first row corresponds to a frame from the video of a shopping mall lobby, the second row to a restaurant food counter and the third row to an airport lobby. The leftmost plot in each row shows the actual frame, the other 5 show the recovered low-rank representations using RPCA, RPCAG, FRPCAG and CPCA with two different uniform downsampling rates. For CPCA Algorithm 2 is used and the rank k for the approximate decoder is set such that $\tilde{\Sigma}_{k,k}/\tilde{\Sigma}_{1,1} < 0.1$, where $\tilde{\Sigma}$ are the singular values of $\tilde{X}$.

For the 2nd and 3rd rows of Fig. 3 it can be seen that our proposed model is able to separate the static backgrounds very accurately from the moving people which do not belong to the static ground truth. However, the quality is slightly compromised in the 1st row where the shadow of the person appears in the low-rank frames recovered with CPCA. In fact, this person remains static for a long time in the video and the uniform sampling compromises the quality slightly.

Table IX: Computational time in seconds of RPCA, RPCAG, FRPCAG and CPCA for low-rank recovery of different videos in Fig. 3.

Videos	RPCA	RPCAG	FRPCAG	CPCA (5,1)	CPCA (10,4)
1	2700	3550	120	21	8
2	1650	2130	85	15	6
3	3650	4100	152	32	11

Table IX presents the computational time in seconds of RPCA, RPCAG, FRPCAG and CPCA for low-rank recovery of different videos in Fig. 3. The time reported here corresponds to steps 2 to 6 of Table I, Algorithm 1 of [36] for FRPCAG, [7] for RPCA and [35] for RPCAG, excluding the construction of graphs G_r, G_c. The speed-up observed for these experiments from Table IX is 10 times over FRPCAG and 100 times over RPCA and RPCAG.

Fig. 4 presents a comparison of the quality of the low-rank static background extracted using the alternate (8) and approximate decoders discussed in (18) for a video (1st row) of Fig. 3. Clearly, the alternate decoder performs slightly better than the approximate decoders but at the price of tuning of two

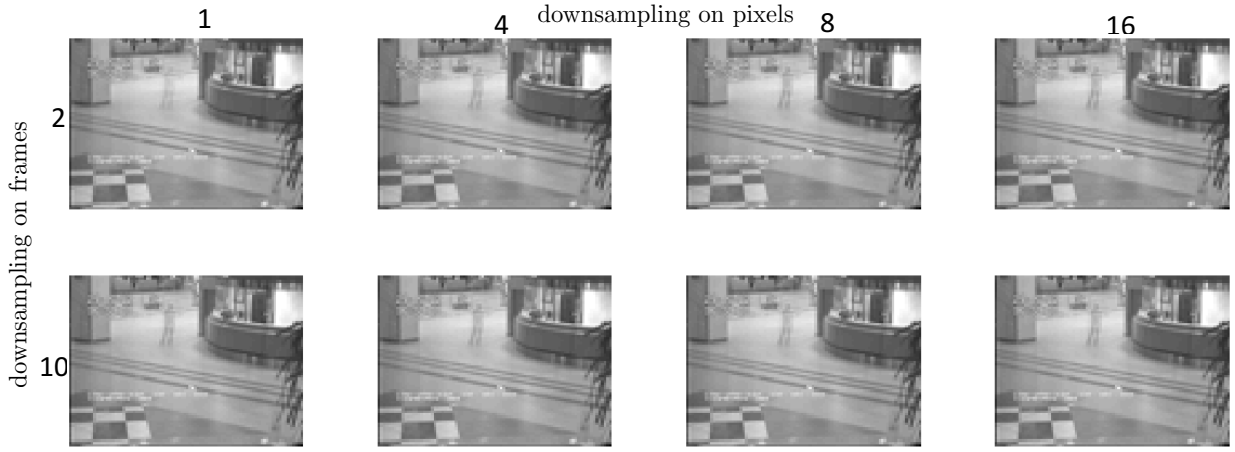

Figure 5: A comparison of the quality of low-rank frames for the shopping mall video (1st row of Fig. 3) extracted using the approximate decoder (18) for different downsampling factors on the pixels and frames. It is obvious that the quality of low-rank remains intact even with higher downsampling factors.

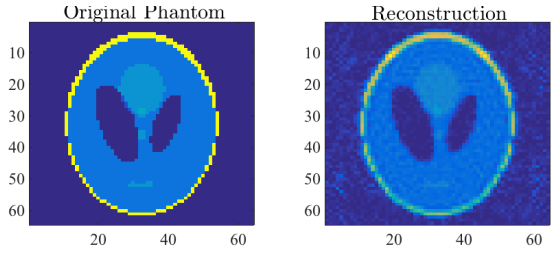

Figure 6: The reconstruction of a 64×64 Modified Shepp-Logan phantom from 20% projections using eq.(25).

model parameters.

Fig. 5 presents a comparison of the quality of low-rankness for the same video extracted using the approximate decoder (18) using different downsampling factors on the pixels and frames. It is obvious that the quality of low-rankness remains intact even with higher downsampling factors.

C. *Low-Rank Recovery from Random Projections*

Throughout this work we assume that the graphs $\mathcal{G}_c$ and $\mathcal{G}_r$ for the complete data matrix Y are either available or can be constructed directly from Y itself using the standard graph construction algorithms. This is a reasonable assumption if one wants to reduce the computational burden by downsampling on the datasets. However, often the complete data matrix Y is not available and the goal is to obtain an estimate of Y from some side information. Typical examples include Magnetic Resonance Imaging (MRI), Computational Tomography (CT) and Electron Tomography (ET) where one only has access to the projections b of Y acquired through a known projection operator A. The purpose here is not to reduce the computational burden but to acquire a good enough estimate of Y from b. Furthermore, for such applications, there is no notion of row or column projection / sampling operators. Nevertheless, one might want to exploit the row and column smoothness assumption for the purpose of reconstruction. While, this is not a significant part of our current work, it is still an obvious open question and the answer comes from an extension of this work. Therefore, to be complete, we propose a framework for such problems which might require a low-rank reconstruction from a few projections. It is important to emphasize though that the goal is not to compare and evaluate the performance rigorously with the state-of-the-art. In fact we mention this here just to give a flavour of how the current framework can be extended for such problems.

Assume that a CT sample, for example, a Shepp-Logan phantom of the size $X \in \mathbb{R}^{p \times n}$ needs to be reconstructed from its projections $b \in \mathbb{R}^m$, obtained via a line projection matrix $A \in \mathbb{R}^{m \times np}$. Thus, $b = Avec(X) + e$, where $e \in \mathbb{R}^m$ models the noise in the projections. We propose to reconstruct the sample X by solving the following optimization problem:

$$\min_X \|Avec(X) - b\|_2^2 + \gamma_r \operatorname{tr}(X^\top \mathcal{L}_r X) + \gamma_c \operatorname{tr}(X \mathcal{L}_c X^\top), \tag{25}$$

where $\mathcal{L}_r, \mathcal{L}_c$ are the row and column graph Laplacians between the rows and columns of X. Since, these graphs are not available in the beginning, one can obtain an initial estimate of X by running a standard compressed sensing problem for a few iterations and then construct these graphs from this estimate. The estimated graphs can also be improved after every few iterations from the more refined X.

Fig. 6 shows the reconstruction of a 64×64 Modified Shepp-Logan phantom from 20% projections using eq.(25). The initial estimate of the graphs $\mathcal{G}_r, \mathcal{G}_c$ between the rows and columns of the phantom is obtained from the first 3 iterations of the compressed sensing based recovery problem and then these graphs are updated every 5 iterations. Our future work will focus on a detailed study of this method.

IX. Conclusion

We present Compressive PCA on Graphs (CPCA) which approximates a recovery of low-rank matrices on graphs from their sampled measurements. It is supported by the proposed restricted isometry property (RIP) which is related to the coherence of the eigenvectors of graphs between the rows and columns of the data matrix. Accompanied with several efficient, parallel, parameter free and low-cost decoders for

low-rank recovery and clustering, the presented framework gains a several orders of magnitude speed-up over the low-rank recovery methods like Robust PCA. Our theoretical analysis reveals that CPCA targets exact recovery for low-rank matrices which are clusterable across the rows and columns. Thus, the error depends on the spectral gaps of the graph Laplacians. Extensive clustering experiments on 5 datasets with various types of noise and comparison with 11 state-of-the-art methods reveal the efficiency of our model. CPCA also achieves state-of-the-art results for background separation from videos.

REFERENCES

[1] A. Aravkin, S. Becker, V. Cevher, and P. Olsen. A variational approach to stable principal component pursuit. arXiv preprint arXiv:1406.1089, 2014. 1

[2] O. Axelsson and G. Lindskog. On the rate of convergence of the preconditioned conjugate gradient method. Numerische Mathematik, 48(5):499–523, 1986. 9

[3] A. Beck and M. Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. SIAM Journal on Imaging Sciences, 2(1):183–202, 2009. 9

[4] M. Belkin and P. Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. Neural computation, 15(6):1373–1396, 2003. 1, 10, 23, 24

[5] C. Boutsidis, M. W. Mahoney, and P. Drineas. An improved approximation algorithm for the column subset selection problem. In Proceedings of the twentieth Annual ACM-SIAM Symposium on Discrete Algorithms, pages 968–977. Society for Industrial and Applied Mathematics, 2009. 1

[6] D. Cai, X. He, J. Han, and T. S. Huang. Graph regularized nonnegative matrix factorization for data representation. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 33(8):1548–1560, 2011. 2, 10, 23, 24

[7] E. J. Candès, X. Li, Y. Ma, and J. Wright. Robust principal component analysis? Journal of the ACM (JACM), 58(3):11, 2011. 1, 10, 14, 23, 24

[8] M. Davenport, P. T. Boufounos, M. B. Wakin, R. G. Baraniuk, et al. Signal processing with compressive measurements. Selected Topics in Signal Processing, IEEE Journal of, 4(2):445–460, 2010. 1

[9] F. Dorfler and F. Bullo. Kron reduction of graphs with applications to electrical networks. Circuits and Systems I: Regular Papers, IEEE Transactions on, 60(1):150–163, 2013. 4

[10] C. Elkan. Using the triangle inequality to accelerate k-means. In ICML, volume 3, pages 147–153, 2003. 22, 23

[11] S. Gao, I.-H. Tsang, and L.-T. Chia. Laplacian sparse coding, hypergraph Laplacian sparse coding, and applications. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 35(1):92–104, 2013. 2

[12] W. Ha and R. F. Barber. Robust PCA with compressed data. In Advances in Neural Information Processing Systems, pages 1927–1935, 2015. 1

[13] N. Halko, P.-G. Martinsson, and J. A. Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. SIAM review, 53(2):217–288, 2011. 1

[14] B. Jiang, C. Ding, and J. Tang. Graph-laplacian PCA: Closed-form solution and robustness. In IEEE Conference on Computer Vision and Pattern Recognition, pages 3492–3498, 2013. 2, 10, 23, 24

[15] T. Jin, J. Yu, J. You, K. Zeng, C. Li, and Z. Yu. Low-rank matrix factorization with multiple hypergraph regularizers. Pattern Recognition, 2014. 2

[16] V. Kalofolias. How to learn a graph from smooth signals. In th International Conference on Artificial Intelligence and Statistics AISTATS, Cadiz, Spain, 2016. 13

[17] D. D. Lee and H. S. Seung. Learning the parts of objects by non-negative matrix factorization. Nature, 401(6755):788–791, 1999. 10, 23, 24

[18] X. Li and J. Haupt. Identifying outliers in large matrices via randomized adaptive compressive sampling. Signal Processing, IEEE Transactions on, 63(7):1792–1807, 2015. 1

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

[20] X. Lu, Y. Wang, and Y. Yuan. Graph-regularized low-rank representation for destriping of hyperspectral images. IEEE transactions on geoscience and remote sensing, 51(7):4009–4018, 2013. 1

[21] M. Muja and D. Lowe. Scalable nearest neighbour algorithms for high dimensional data. 2014. 3

[22] M. Muja and D. G. Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. VISAPP (1), 2, 2009. 11

[23] T.-H. Oh, Y. Matsushita, Y.-W. Tai, and I. S. Kweon. Fast randomized singular value thresholding for nuclear norm minimization. In IEEE Conference on Computer Vision and Pattern Recognition, pages 4484–4493, 2015. 1

[24] E. Pavez and A. Ortega. Generalized Laplacian precision matrix estimation for graph signal processing. In 2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pages 6350–6354, 2016. 13

[25] N. Perraudin, J. Paratte, D. Shuman, V. Kalofolias, P. Vandergheynst, and D. K. Hammond. GSPBOX: A toolbox for signal processing on graphs. ArXiv e-prints, #aug# 2014. 9

[26] N. Perraudin, D. Shuman, G. Puy, and P. Vandergheynst. UNLocBoX A matlab convex optimization toolbox using proximal splitting methods. ArXiv e-prints, #feb# 2014. 9

[27] N. Perraudin and P. Vandergheynst. Stationary signal processing on graphs. ArXiv e-prints, #jan# 2016. 2

[28] G. Puy, N. Tremblay, R. Gribonval, and P. Vandergheynst. Random sampling of bandlimited signals on graphs. arXiv preprint arXiv:1511.05118, 2015. 3, 4, 5, 8, 9, 18, 20

[29] M. Rahmani and G. Atia. High dimensional low rank plus sparse matrix decomposition. arXiv preprint arXiv:1502.00182, 2015. 1

[30] M. Rahmani and G. K. Atia. Randomized robust subspace recovery for big data. In Machine Learning for Signal Processing (MLSP), 2015 IEEE 25th International Workshop on, pages 1–6. IEEE, 2015. 1

[31] N. Rao, H.-F. Yu, P. K. Ravikumar, and I. S. Dhillon. Collaborative filtering with graph information: Consistency and scalable methods. In Advances in Neural Information Processing Systems, pages 2107–2115, 2015. 6, 7

[32] S. T. Roweis and L. K. Saul. Nonlinear dimensionality reduction by locally linear embedding. Science, 290(5500):2323–2326, 2000. 10, 23, 24

[33] J. Sankaranarayanan, H. Samet, and A. Varshney. A fast all nearest neighbor algorithm for applications involving large point-clouds. Computers Graphics, 31(2):157–174, 2007. 3

[34] S. Segarra, A. G. Marques, G. Mateos, and A. Ribeiro. Network topology identification from spectral templates. arXiv preprint arXiv:1604.02610, 2016. 13

[35] N. Shahid, V. Kalofolias, X. Bresson, M. Bronstein, and P. Vandergheynst. Robust principal component analysis on graphs. arXiv preprint arXiv:1504.06151, 2015. 1, 9, 10, 14, 23, 24

[36] N. Shahid, N. Perraudin, V. Kalofolias, G. Puy, and P. Vandergheynst. Fast robust PCA on graphs. IEEE Journal of Selected Topics in Signal Processing, 10(4):740–756, 2016. 2, 3, 5, 9, 10, 14, 19, 23, 24

[37] F. Shang, L. Jiao, and F. Wang. Graph dual regularization non-negative matrix factorization for co-clustering. Pattern Recognition, 45(6):2237–2250, 2012. 2

[38] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst. The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. Signal Processing Magazine, IEEE, 30(3):83–98, 2013. 1, 3

[39] A. Susnjara, N. Perraudin, D. Kressner, and P. Vandergheynst. Accelerated filtering on graphs using lanczos method. arXiv preprint arXiv:1509.04537, 2015. 5, 23

[40] A. Talwalkar and A. Rostamizadeh. Matrix coherence and the nystrom method. arXiv preprint arXiv:1004.2008, 2010. 1

[41] L. Tao, H. H. Ip, Y. Wang, and X. Shu. Low rank approximation with sparse integration of multiple manifolds for data representation. Applied Intelligence, pages 1–17, 2014. 2

[42] N. Tremblay, G. Puy, R. Gribonval, and P. Vandergheynst. Compressive spectral clustering. arXiv preprint arXiv:1602.02018, 2016. 8

[43] J. A. Tropp. On the conditioning of random subdictionaries. Applied and Computational Harmonic Analysis, 25(1):1–24, 2008. 1

[44] R. Witten and E. Candes. Randomized algorithms for low-rank matrix factorizations: sharp performance bounds. Algorithmica, pages 1–18, 2013. 1

[45] C. You, D. Robinson, and R. Vidal. Scalable sparse subspace clustering by orthogonal matching pursuit. In IEEE Conference on Computer Vision and Pattern Recognition, volume 1, 2016. 1

[46] Z. Zhang and K. Zhao. Low-rank matrix approximation with manifold regularization. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 35(7):1717–1729, 2013. 2, 10, 23, 24

N.Shahid studied BSc Electrical Engineering from UET Lahore and MSc Computer Engineering from LUMS, Pakistan, where he specialized in Signal Processing and Machine Learning. In 2013, he started his PhD at LTS2 (laboratoire de traitement du signal) at EPFL under the theme "Signal Processing on Graphs". His current work focuses on fast and approximate methods for Principal Component Analysis on Graphs.

N. Perraudin studied electrical engineering at EPFL ("Ecole polytechnique federal de Lausanne") where he specialized in signal processing. After his degree he spent six months as research assistant at the Acoustic Research Institute at the Austrian Academy of Science. In 2013 he started his PhD at LTS2 (laboratoire de traitement du signal) at EPFL. His current research focuses on the intersection of different domains: spectral graph theory, optimization and machine learning. He is the founder and the maintainer of two open-source projects, the UNLocBoX (a convex optimization toolbox, https://lts2.epfl.ch/unlocbox/) and the GSPBox (a graph signal processing toolbox, https://lts2.epfl.ch/gspbox/).

G. Puy received the M.Sc. degree in Electrical and Electronics Engineering from Ecole Polytechnique Fédérale de Lausanne (EFFL), Switzerland, and the Engineering Diploma from Supelec, France, in 2009. He obtained the Ph.D. degree from EPFL in 2014. He received the EPFL Chorafas Foundation award in 2013 and the EPFL Doctorate award in 2015. From 2014 to 2016, he was a postdoctoral researcher at INRIA, Rennes, in the PANAMA research team. He is now a researcher at Technicolor R&D France in Rennes. His research interests are in the field of signal processing and machine learning. He works on the design of efficient sampling methods for high dimensional data (compressive sampling), low-dimensional representations, and non-linear reconstruction methods using convex and non-convex optimization, with applications in imaging problems.

P. Vandergheynst received the M.S. degree in physics and the Ph.D. degree in mathematical physics from the Université catholique de Louvain, Louvain-la-Neuve, Belgium, in 1995 and 1998, respectively. From 1998 to 2001, he was a Postdoctoral Researcher with the Signal Processing Laboratory, Swiss Federal Institute of Technology (EPFL), Lausanne, Switzerland. He was Assistant Professor at EPFL (2002-2007), where he is now a Full Professor. His research focuses on harmonic analysis, sparse approximations, mathematical image processing and most importantly Signal Processing on Graphs. He has been on the Technical Committee of various conferences and was Co-General Chairman of the EUSIPCO 2008 conference. Pierre Vandergheynst is the author or co-author of more than 50 journal papers, one monograph and several book chapters. He's a laureate of the Apple ARTS award and of the 2010-2011 De Boelpaepe prize from the Royal Academy of Sciences of Belgium. Pierre is strongly involved in technology transfer: he co-founded two start-ups and holds numerous patents.