Fusion Subspace Clustering: Full & Incomplete Data

Daniel L. Pimentel-Alarcón, Usman Mahmood
Georgia State University
pimentel@gsu.edu, umahmood1@student.gsu.edu

Abstract

Modern inference and learning often hinge on identifying low-dimensional structures that approximate large scale data. Subspace clustering achieves this through a union of linear subspaces. However, in contemporary applications data is increasingly often incomplete, rendering standard (full-data) methods inapplicable. On the other hand, existing incomplete-data methods present major drawbacks, like lifting an already high-dimensional problem, or requiring a super polynomial number of samples. Motivated by this, we introduce a new subspace clustering algorithm inspired by fusion penalties. The main idea is to permanently assign each datum to a subspace of its own, and minimize the distance between the subspaces of all data, so that subspaces of the same cluster get *fused* together. Our approach is entirely new to both, full and missing data, and unlike other methods, it directly allows noise, it requires no liftings, it allows low, high, and even full-rank data, it approaches optimal (information-theoretic) sampling rates, and it does not rely on other methods such as low-rank matrix completion to handle missing data. Furthermore, our extensive experiments on both real and synthetic data show that our approach performs comparably to the state-of-the-art with complete data, and dramatically better if data is missing.

1 Introduction

Inferring low-dimensional structures that explain high-dimensional data has become a cornerstone of discovery in virtually all fields of science. Principal component analysis (PCA), which identifies the low-dimensional linear subspace that best explains a dataset, is arguably the most prominent technique for this purpose. However, in many applications — computer vision, image processing, bioinformatics, linguistics, networks analysis, and more [1–10] — data is often composed of a mixture of several classes, each of which can be explained with a different subspace. Clustering and inferring subspaces that explain data is an important unsupervised learning problem that has received tremendous attention in recent years, producing theory and algorithms to handle outliers, noisy measurements, privacy concerns, and data constraints, among other difficulties [11–22].

However, one major challenge in contemporary problems is that data is often incomplete. For example, in image inpainting, the values of some pixels are missing due to faulty sensors and image contamination [23]; in computer vision features are often missing due to occlusions and tracking algorithms malfunctions [24]; in recommender systems each user only rates a limited number of items [25]; in a network, most nodes communicate in subsets, producing only a handful of all the possible measurements [7].

Prior Work. There are numerous approaches to subspace clustering with missing data. For example, [26] suggests using the standard (full-data) state-of-the-art algorithm sparse subspace clustering (SSC) [9] after filling in the missing entries with a sensible value (e.g., zeros or means) or with low-rank matrix completion (LRMC) [27]. However, the number and dimensions of the subspaces are often large enough that LRMC methods are not applicable, and data filled with zeros or means no longer lie in a union of subspaces (UoS), thus guaranteeing failure even with a modest amount of missing data [32]. On the other hand, [28] gives an algorithm that uses partial neighborhoods and provably works, but requires a super-polynomial amount of samples, which is unusual in most applications. Alternating methods include adaptations of k-subspaces to handle missing data [29], an expectation-maximization (EM) algorithm that models UoS as a gaussian mixture [30], and a group-lasso formulation [31]. However, these alternating algorithms depend heavily on initialization, and can only be guaranteed to converge to a local minimum. More recently, [32–34] use lifting techniques, yet this requires (at the very least) squaring the dimension of an already high-dimensional problem, which severely limits their applicability.

Paper Contributions. In this paper we propose an entirely different approach to subspace clustering (SC), inspired by fusion penalties [35–40]. The main idea is to permanently assign each datum to a subspace of its own, and then *fuse* together nearby subspaces by minimizing (i) the distance between each datum and its subspace (thus guaranteeing that each datum is explained by its subspace), and (ii) the distance between the subspaces from all data, so that subspaces from points that belong together get fused into one. Our algorithm, which we call *fusion subspace clustering* (FSC) is new to both, the full and incomplete data settings, and has the next advantages over the state-of-the-art:

– FSC does not require a super-polynomial number of samples, as [28]. In fact, our experiments show that FSC succeeds with only a few more samples than the strictly necessary [41].

– There is a natural way to extend FSC to handle missing data, unlike other algorithms, including the state-of-the-art for full data, SSC. Similarly, FSC carries unchanged to noisy settings, unlike other results such as [41].

– LRMC methods adapted to multiple subspaces, as well as SC methods adapted to missing data often rely on SSC and LRMC [26, 31–34]. In contrast, FSC is a novel algorithm, rather than an adaptation, and it does not rely on SSC nor any other SC algorithm. In addition to clustering, Fsc provides a subspace estimation method, and a data completion method that do not require LRMC.

- Two-stage methods like [26], which first complete using LRMC, and then cluster using SSC (or vice versa), only work if data lie in a UoS but remains low-rank, allowing only a few number of subspaces of very small dimensions. Fsc works even in the more general setting where data lie in a UoS and is high-rank, or even full-rank, thus allowing more subspaces and of larger dimensions.

- Unlike alternating algorithms such as [30,31], Fsc does not require knowing the number of subspaces nor their dimensions. Similar to hierarchical clustering, Fsc can produce a progressive clustering that gradually fuses subspaces together. Combined with a simple goodness of fit test, like the Akaike information criterion (AIC) [42], this provides a model selection criteria to determine the number of subspaces and their dimensions.

- Recent approaches such as [32–34] exploit the algebraic structure of a UoS. In fact, each UoS describes an algebraic (non-linear) variety that can be used to cluster and complete. However, expressing the polynomials of these varieties requires lifting (or tensorizing) the data, which turns an already high-dimensional problem into an extremely high-dimensional one, thus limiting their applicability. For example, processing a small image of 64×64 pixels requires a vector space of dimension $\mathrm{d} = 64^2 = 4,096$. Even the smallest lifting results in a space of dimension $\binom{\mathrm{d}+1}{2} = 8,390,656$. In contrast, Fsc requires no data liftings.

- Our experiments on real and synthetic data show that with full data, Fsc performs comparably to the state-of-the-art, and dramatically better if data is missing.

2 Problem Statement

Let $\mathbf{X} \in \mathbb{R}^{\mathrm{d}\times\mathrm{n}}$ be a data matrix whose columns lie *approximately* in the union of K low-dimensional subspaces of $\mathbb{R}^{\mathrm{d}}$ (i.e., we allow noise). Assume that we do not know a priori the subspaces, nor how many there are, nor their dimensions, nor which column belongs to which subspace. Let $\mathbf{X}^{\Omega}$ denote the incomplete version of $\mathbf{X}$, observed only in the entries of $\Omega \subset \{1,\dots,\mathrm{d}\} \times \{1,\dots,\mathrm{n}\}$. Given $\mathbf{X}^{\Omega}$, our goals are to cluster the columns of $\mathbf{X}^{\Omega}$ according to the underlying subspaces, infer such subspaces, and complete $\mathbf{X}^{\Omega}$.

Notations. Throughout the paper, $\mathbf{x}_{\mathrm{i}} \in \mathbb{R}^{\mathrm{d}}$ denotes the ith column of $\mathbf{X}$, $\mathbb{U}_{\mathrm{i}} \subset \mathbb{R}^{\mathrm{d}}$ denotes the subspace assigned to $\mathbf{x}_{\mathrm{i}}$, and $\mathbf{U}_{\mathrm{i}} \in \mathbb{R}^{\mathrm{d}\times\mathrm{r}}$ is a basis of $\mathbb{U}_{\mathrm{i}}$; here $\mathrm{i} = 1,\dots,\mathrm{n}$, and r is an upper bound on the dimension of the subspaces. Given i, we use the superscript ω to indicate the restriction of a subspace, matrix or vector to the observed entries in $\mathbf{x}_{\mathrm{i}}$. For example, if $\mathbf{x}_{\mathrm{i}}$ is observed on

ℓ entries, then $\mathbf{x}_\mathrm{i}^\omega \in \mathbb{R}^\ell$ and $\mathbf{U}_\mathrm{i}^\omega \in \mathbb{R}^{\ell\times \mathrm{r}}$ denote the restrictions of $\mathbf{x}_\mathrm{i}$ and $\mathbf{U}_\mathrm{i}$ to the observed entries in $\mathbf{x}_\mathrm{i}$. Similarly, we denote the projection operators onto $\mathbb{U}_\mathrm{i}$ and $\mathbb{U}_\mathrm{i}^\omega$ (the subspace assigned to $\mathbf{x}_\mathrm{i}$, and its restriction to the observed entries in $\mathbf{x}_\mathrm{i}$) as

$$\mathbf{P}_\mathrm{i} := \mathbf{U}_\mathrm{i}(\mathbf{U}_\mathrm{i}^\mathsf{T}\mathbf{U}_\mathrm{i})^{-1}\mathbf{U}_\mathrm{i}^\mathsf{T} \in \mathbb{R}^{\mathrm{d}\times\mathrm{d}} \qquad \text{and} \qquad \mathbf{P}_\mathrm{i}^\omega := \mathbf{U}_\mathrm{i}^\omega(\mathbf{U}_\mathrm{i}^{\omega\mathsf{T}}\mathbf{U}_\mathrm{i}^\omega)^{-1}\mathbf{U}_\mathrm{i}^{\omega\mathsf{T}} \in \mathbb{R}^{\ell\times\ell}. \tag{1}$$

Finally, $\|\cdot\|_\mathrm{F}^2$ denotes the squared Frobenius norm, given by the sum of squared entries in a matrix.

3 Review of Fusion Penalties

Fusion penalties can be viewed as a convex relaxation of hierarchical clustering [35–40]. Given n points $\mathbf{y}_1,\dots,\mathbf{y}_\mathrm{n} \in \mathbb{R}^\mathrm{d}$ (not necessarily in a UoS), hierarchical clustering is the greedy algorithm that recursively joins nearby points until only K clusters remain, aiming to find:

$$\underset{\boldsymbol{\mu}_1,\dots,\boldsymbol{\mu}_\mathrm{n}}{\arg\min} \sum_{\mathrm{i}=1}^{\mathrm{n}} \|\mathbf{y}_\mathrm{i} - \boldsymbol{\mu}_\mathrm{i}\|_2^2 \quad \text{subject to} \quad \frac{1}{2}\sum_{\mathrm{i}=1}^{\mathrm{n}}\sum_{\mathrm{j}=1}^{\mathrm{n}} \mathbb{1}_{\{\boldsymbol{\mu}_\mathrm{i}\neq\boldsymbol{\mu}_\mathrm{j}\}} \leq \mathrm{K}, \tag{2}$$

where $\mathbb{1}$ denotes the indicator function. Notice that if $\mathrm{K} \geq \mathrm{n}(\mathrm{n}+1)/2$ (the number of distinct pairs of points), then the problem is unconstrained, and the trivial solution is $\mathbf{y}_\mathrm{i} = \boldsymbol{\mu}_\mathrm{i}$ for every i. If $\mathrm{K} = \mathrm{n}(\mathrm{n}+1)/2 - 1$, then (2) forces two centers to *fuse*, which equates to the first step in hierarchical clustering. More generally, if $\mathrm{K} = \mathrm{n}(\mathrm{n}+1)/2 - \ell$, then (2) forces $\ell - 1$ centers to fuse. Since (2) is a difficult combinatorial problem, [38] proposed the following convex relaxation:

$$\underset{\boldsymbol{\mu}_1,\dots,\boldsymbol{\mu}_\mathrm{n}}{\arg\min} \sum_{\mathrm{i}=1}^{\mathrm{n}} \|\mathbf{y}_\mathrm{i} - \boldsymbol{\mu}_\mathrm{i}\|_2^2 \quad \text{subject to} \quad \frac{1}{2}\sum_{\mathrm{i}=1}^{\mathrm{n}}\sum_{\mathrm{j}=1}^{\mathrm{n}} \|\boldsymbol{\mu}_\mathrm{i} - \boldsymbol{\mu}_\mathrm{j}\|_2^2 \leq \mathrm{K}'. \tag{3}$$

Intuitively, (3) aims to minimize the euclidian distance between each point $\mathbf{y}_\mathrm{i}$ and its assigned center $\boldsymbol{\mu}_\mathrm{i}$, while at the same time guaranteeing that the centers from different points are not too far from one an other (which is regulated by K'). The next section uses these insights to address subspace clustering. The crucial difference is that now *centers* will be subspaces rather than points, and so we can no longer use the euclidian distances in (3), as two points in the same subspace can be arbitrarily apart with respect to the euclidian distance.

4 Fusion Subspace Clustering

We now present our novel algorithm: Fusion Subspace Clustering (FSC). The main idea is to permanently assign each column $\mathbf{x}_\mathrm{i}$ to a subspace of its own that is close to $\mathbf{x}_\mathrm{i}$, and close to subspaces assigned to other columns, so that the subspaces that belong together get *fused*. To do this, we will minimize (i) the residual of $\mathbf{x}_i$ when projected onto $\mathbb{U}_i$, to measure the distance between $\mathbf{x}_i$ and its assigned subspace $\mathbb{U}_i$, and (ii) the difference between the projector operators of $\mathbb{U}_i$ and $\mathbb{U}_j$, to measure the distance (over the Grassmannian) between the subspaces assigned to columns i and j, i.e.,

$$\boxed{\arg\min_{\mathbf{U}_1,\ldots,\mathbf{U}_n} \sum_{i=1}^{n} \|\mathbf{x}_i - \mathbf{P}_i\mathbf{x}_i\|_2^2 \; + \; \frac{\lambda}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \|\mathbf{P}_i - \mathbf{P}_j\|_F^2, \qquad (4)}$$

where the dependency on each $\mathbf{U}_i$ is hidden in $\mathbf{P}_i$ (see (1)), and $\lambda \geq 0$ is a proxy of K that regulates how clusters fuse together. The larger λ, the more we penalize subspaces being apart, which results in subspaces getting closer (fused); more details in Section 6. Recall that $\mathbf{U}_i \in \mathbb{R}^{d \times r}$, where r is an upper bound on the subspaces dimension. Section 6 also includes more details about estimating r.

In order to solve (4), in our experiments we use a random initialization and standard gradient descent, where the gradient of $\mathbf{U}_i$ is given by:

$$\begin{aligned} \boldsymbol{\nabla}_i \; &= \; -2\mathbf{x}_i\mathbf{x}_i^{\mathsf{T}}\mathbf{U}_i(\mathbf{U}_i^{\mathsf{T}}\mathbf{U}_i)^{-1} \; + \; (\mathbf{U}_i\mathbf{U}_i^{\mathsf{T}})^2\mathbf{x}_i\mathbf{x}_i^{\mathsf{T}}\mathbf{U}_i(\mathbf{U}_i^{\mathsf{T}}\mathbf{U}_i)^{-1} \\ &+ \; \mathbf{U}_i(\mathbf{U}_i^{\mathsf{T}}\mathbf{U}_i)^{-1}\mathbf{U}_i^{\mathsf{T}}\mathbf{x}_i\mathbf{x}_i^{\mathsf{T}}\mathbf{U}_i\mathbf{U}_i^{\mathsf{T}}\mathbf{U}_i \\ &+ \; 8\sum_{j\neq i}(\mathbf{U}_i(\mathbf{U}_i^{\mathsf{T}}\mathbf{U}_i)^{-1}\mathbf{U}_i^{\mathsf{T}} - \mathbf{I})\mathbf{U}_j(\mathbf{U}_j^{\mathsf{T}}\mathbf{U}_j)^{-1}\mathbf{U}_j^{\mathsf{T}}\mathbf{U}_i(\mathbf{U}_i^{\mathsf{T}}\mathbf{U}_i)^{-1}. \qquad (5) \end{aligned}$$

However, we can also solve (4) using more sophisticated techniques, like an alternating direction method of multipliers (ADMM) formulation [43, 44] and smarter initializations, for example setting the initial subspaces to be orthogonal or with small principal angles between them.

The solution to (4) will be a sequence of subspace bases $\mathbf{U}_1, \ldots, \mathbf{U}_n$, one for each column in $\mathbf{X}$. Due to the second term in (4), we expect subspace $\mathbb{U}_i := \text{span}\{\mathbf{U}_i\}$ to be close to $\mathbb{U}_j := \text{span}\{\mathbf{U}_j\}$ if columns $\mathbf{x}_i$ and $\mathbf{x}_j$ belong together, and far otherwise. It remains to group together subspaces that are close, or equivalently, assign a label to each subspace $\mathbb{U}_i$. To this end, use spectral clustering, which shows remarkable performance in many modern problems, and is widely used as the final step in many subspace clustering algorithms, including the state-of-the-art SSC. Spectral clustering receives a similarity matrix $\mathbf{S} \in \mathbb{R}^{n \times n}$ between n points, and runs a standard clustering method (like *k*-means) on the relevant eigenvectors of the Laplacian matrix of $\mathbf{S}$ []. We can build a similarity matrix $\mathbf{S}$ between subspaces $\mathbb{U}_1, \ldots, \mathbb{U}_n$ whose $(i, j)^{\text{th}}$ entry is equal to $1/\|\mathbf{P}_i - \mathbf{P}_j\|_F^2$. At this point we can run spectral clustering on $\mathbf{S}$ to assign a label $k_i \in \{1, \ldots, K\}$ to each subspace $\mathbb{U}_i$, or equivalently, to each column $\mathbf{x}_i$, thus providing a clustering of $\mathbf{X}$, as desired. The entire process of FSC is summarized in Algorithm 1.

5 A Natural Generalization to Missing Data

Section 4 introduces Fsc in its full-data setting. If data is missing, the only difference is that each subspace only needs to explain the observed entries of its assigned column, resulting in the following:

$$\boxed{\quad \underset{\mathbf{U}_1,\dots,\mathbf{U}_n}{\arg\min} \ \sum_{i=1}^{n} \|\mathbf{x}_i^{\omega} - \mathbf{P}_i^{\omega}\mathbf{x}_i^{\omega}\|_2^2 \ + \ \frac{\lambda}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \|\mathbf{P}_i - \mathbf{P}_j\|_F^2. \qquad (6)\quad}$$

In words, (6) simply ignores all the unobserved entries in the first term of (4). Notice that if columns $\mathbf{x}_i^{\omega}$ and $\mathbf{x}_j^{\omega}$ belong together, we still want the whole subspaces $\mathbb{U}_i$ and $\mathbb{U}_j$ to be close, so the second term in (4) remains unchanged. As consequence, this time the gradient of $\mathbf{U}_i$ is equal to

$$\boldsymbol{\nabla}_i \ = \ \boldsymbol{\nabla}_i' \ + \ 8\sum_{j\neq i}(\mathbf{U}_i(\mathbf{U}_i^{\mathsf{T}}\mathbf{U}_i)^{-1}\mathbf{U}_i^{\mathsf{T}} - \mathbf{I})\mathbf{U}_j(\mathbf{U}_j^{\mathsf{T}}\mathbf{U}_j)^{-1}\mathbf{U}_j^{\mathsf{T}}\mathbf{U}_i(\mathbf{U}_i^{\mathsf{T}}\mathbf{U}_i)^{-1},$$

where $\boldsymbol{\nabla}_i'$ is equal to $\mathbf{0}$ for the rows not in ω, and equal to

$$\begin{aligned}\boldsymbol{\nabla}_i^{\omega} \ = \ & -2\mathbf{x}_i^{\omega}\mathbf{x}_i^{\omega\mathsf{T}}\mathbf{U}_i^{\omega}(\mathbf{U}_i^{\omega\mathsf{T}}\mathbf{U}_i^{\omega})^{-1} \ + \ (\mathbf{U}_i^{\omega}\mathbf{U}_i^{\omega\mathsf{T}})^2\mathbf{x}_i^{\omega}\mathbf{x}_i^{\omega\mathsf{T}}\mathbf{U}_i^{\omega}(\mathbf{U}_i^{\omega\mathsf{T}}\mathbf{U}_i^{\omega})^{-1} \\ & + \ \mathbf{U}_i^{\omega}(\mathbf{U}_i^{\omega\mathsf{T}}\mathbf{U}_i^{\omega})^{-1}\mathbf{U}_i^{\omega\mathsf{T}}\mathbf{x}_i^{\omega}\mathbf{x}_i^{\omega\mathsf{T}}\mathbf{U}_i^{\omega}\mathbf{U}_i^{\omega\mathsf{T}}\mathbf{U}_i^{\omega}.\end{aligned}$$

for the rows in ω. Intuitively, the incomplete-data gradient is equal to (5), only ignoring the first line for the rows not in ω. Using this new gradient, we can solve (6) same as (4), using random or orthogonal initializations and gradient descent, or more sophisticated methods, like ADMM [43,44]. We point out that even though $\mathbf{X}^{\Omega}$ has missing data, the solution to (6) will be a sequence of (full) subspace bases $\mathbf{U}_1,\dots,\mathbf{U}_n$, same as in the full-data case. Hence, as the final step we can assign labels using spectral clustering, same as before, thus clustering $\mathbf{X}^{\Omega}$.

Algorithm 1: Fusion Subspace Clustering.

FULL DATA:	INCOMPLETE DATA:
1. **Input:** Columns $\mathbf{x}_1,\dots,\mathbf{x}_n$.	1. **Input:** Columns $\mathbf{x}_1^{\omega},\dots,\mathbf{x}_n^{\omega}$.
2. Solve (4) to obtain $\mathbf{U}_1,\dots,\mathbf{U}_n$.	2. Solve (6) to obtain $\mathbf{U}_1,\dots,\mathbf{U}_n$.

3. Compute $\mathbf{S}$, the similarity matrix of $\mathbf{U}_1,\dots,\mathbf{U}_n$.
4. Spectral cluster $\mathbf{S}$ to obtain labels $k_1,k_2,\dots,k_n$.
5. **Output:** The ith column corresponds to cluster k_i.

5.1 Estimating Subspaces and Completing Data

Recall that our goals are to: (i) cluster the columns of $\mathbf{X}^\Omega$, (ii) infer the underlying subspaces, and (iii) complete $\mathbf{X}^\Omega$. So far we have only achieved (i). However, that is the difficult step. In fact, once $\mathbf{X}^\Omega$ is clustered, there are several straightforward ways to achieve (ii) and (iii).

Common approaches concatenate all the columns of $\mathbf{X}^\Omega$ that correspond to the same cluster into a single matrix $\mathbf{X}_\mathrm{k}^\Omega$, and complete it into a matrix $\hat{\mathbf{X}}_\mathrm{k}$ using LRMC (because its columns now lie in a single subspace), thus achieving (iii). To accomplish (ii) one can compute the leading singular vectors of $\hat{\mathbf{X}}_\mathrm{k}$ to produce an subspace basis estimate $\hat{\mathbf{U}}_\mathrm{k}$. We can do this as well. However Fsc offers natural estimation and completion methods that do not rely on LRMC, which may fail if the subspaces are coherent (somewhat aligned with the canonical axes) or samples are not uniformly spread [27].

Our LRMC-free approach is as follows: since the bases $\mathbf{U}_1, \dots, \mathbf{U}_\mathrm{n}$ produced by (6) have no missing data, we can normalize and concatenate all the bases that correspond to the k^{th} cluster into a single matrix $\mathbf{W}_\mathrm{k}$, and compute its leading singular vectors to produce an *average* estimate $\hat{\mathbf{U}}_\mathrm{k}$, thus achieving (ii). Next we can estimate the coefficient of each incomplete column $\mathbf{x}_\mathrm{i}^\omega$ with respect to its corresponding subspace basis $\hat{\mathbf{U}}_{\mathrm{k_i}}$ as $\hat{\boldsymbol{\theta}}_\mathrm{i} := (\hat{\mathbf{U}}_{\mathrm{k_i}}^{\omega\mathsf{T}} \hat{\mathbf{U}}_{\mathrm{k_i}}^{\omega})^{-1} \hat{\mathbf{U}}_{\mathrm{k_i}}^{\omega\mathsf{T}} \mathbf{x}_\mathrm{i}^\omega$. Since the coefficient of $\mathbf{x}_\mathrm{i}^\omega$ is the same as the coefficient of $\mathbf{x}_\mathrm{i}$ we can complete $\mathbf{x}_\mathrm{i}^\omega$ as $\hat{\mathbf{x}}_\mathrm{i} = \hat{\mathbf{U}}_{\mathrm{k_i}} \hat{\boldsymbol{\theta}}_\mathrm{i}$, thus achieving (iii).

6 Model Selection

In this section we discuss how Fsc provides a natural way for model selection, namely determining the number of subspaces K that best explain the data, and their dimensions. Intuitively, the first terms in (4) and (6) guarantee that each subspace $\mathbb{U}_\mathrm{i}$ is close to its assigned column, and the second terms guarantee that subspaces from different columns are close to one another. The tradeoff between these two quantities is determined by $\lambda \geq 0$ (see Figure 1). If $\lambda = 0$, then the second term is ignored, and there is a trivial solution where each subspace exactly contains its assigned column (thus attaining the minimum, zero, in the first term). If $\lambda > 0$, the second term forces subspaces from different columns to get closer, even if they no longer contain exactly their assigned columns. As λ grows, subspaces get closer and closer, up to the point where some subspaces *fuse* into one. This is verified in our experiments (see Figure 2). The extreme case ($\lambda = \infty$) forces all subspaces to fuse into one (to attain zero in the second term), meaning we only have one subspace to explain all data, which is precisely PCA (for full data) and LRMC (for incomplete data). In other words, Fsc is a generalization of PCA and LRMC which are the particular cases of (4) and (6) with $\lambda = \infty$.

This way, $\lambda = \infty$ will produce a single subspace cluster. Iteratively decreasing λ will result in more and more clusters, until $\lambda = 0$ produces n, analogous to the *clusterpath* produced in [38] for euclidian clustering. The more subspaces,

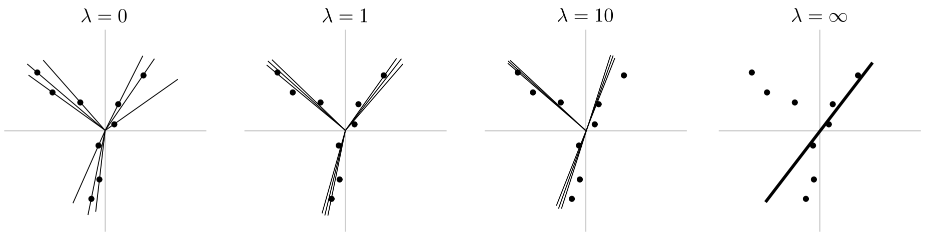

Figure 1: In (4) and (6), $\lambda \geq 0$ regulates how clusters fuse together. If $\lambda = 0$, each point is assigned to a subspace that exactly contains it (overfitting). The larger λ, the more we penalize subspaces being apart, which results in subspaces getting closer (as with $\lambda = 1$), up to the point that some subspaces fuse (as with $\lambda = 10$). In the extreme ($\lambda = \infty$), all subspaces fuse together, and we need to explain all data with a single subspace (which may not be enough). Notice that it is not always evident how many subspaces one should use to explain a dataset. In this illustration, should we choose $\lambda = 1$, which would result in 3 subspaces, or $\lambda = 10$, which would result in 2? Section 6 discusses how to choose λ, which in turn determines the number of subspaces K that best explain the data, and their dimensions.

the more accuracy, but the more degrees of freedom (overfitting). For each λ that provides a different clustering, we can compute a goodness of fit test (like the Akaike information criterion, AIC [42]) that quantifies the tradeoff between accuracy and degrees of freedom, to determine the best number of subspaces K. For example, this test can be in the form of K, and the residuals of the projections of each $\mathbf{x}_{\mathrm{i}}^{\omega}$ onto its corresponding $\hat{\mathbf{U}}_{\mathrm{k}}^{\omega}$, as defined in Section 5.1. Similarly, we can iteratively increase r to find all the columns that lie in 1-dimensional subspaces, then all the columns that lie in 2-dimensional subspaces, and so on (pruning the data at each iteration). After this we will have an estimate of the number of subspaces K, and their dimensions.

7 Experiments

In this section we study the performance of Fsc on both, synthetic and real data. For reference, we compare against the following subspace clustering algorithms that allow missing data: (*a*) Entry-wise zero-filling followed by SSC (EWZF-SSC) [26]. (*b*) Matrix Completion plus SSC (MC+SSC) [26]. (*c*) SSC-Lifting (SSCL) [32]. (*c*) Algebraic variety high-rank matrix completion (AVHRMC) [33]. (*e*) *k*-subspaces with missing data (*k*S) [29]. (*f*) Expectation-maximization (EM) [30]. (*g*) Group-sparse subspace clustering (GSSC) [31]. (*h*) Mixture subspace clustering (MSC) [31]. We chose these algorithms based on [31,32], where they show comparable state-of-the-art performance. With full data, (*a*)-(*b*) simplify to the acclaimed state-of-the-art SSC. (*c*)-(*d*) are both lifting schemes. Same as (*a*)-(*b*), they are two-step procedures that at some point require using SSC. (*e*)-(*h*) are alternating algorithms that depend heavily on initialization; according to [31], they produce best results when initialized with the output of 1, and so indirectly they also depend on SSC. To measure performance we compute clustering error (fraction of misclassified points).

7.1 Simulations

Since Fsc is an entirely new subspace clustering approach to both, full and incomplete data, we present a thorough series of experiments to study its behavior as a function of the penalty parameter λ, the ambient dimension d, the number of subspaces involved K, their dimensions r, the noise variance σ^2, the number of data points in each cluster n_k, and of course, the fraction of observed entries p. Unless otherwise stated, we use the following default settings: $d = 100$, $K = 4$, $r = 5$, $\sigma = 0$, $n_k = 20$, and $p = 1$. We run 30 trials of each experiment, and show the average results of Fsc and the best result of each experiment amongst algorithms (a)-(h) above.

In all our simulations we first generate K matrices $\mathbf{U}_k^\star \in \mathbb{R}^{d\times r}$ with i.i.d. $\mathcal{N}(0,1)$ entries, to use as bases of the *true* subspaces. For each k we generate a matrix $\mathbf{\Theta}_k^\star \in \mathbb{R}^{r\times n_k}$, also with i.i.d. $\mathcal{N}(0,1)$ entries, to use as coefficients of the columns in the kth subspace. We then form $\mathbf{X}$ as the concatenation $[\mathbf{U}_1{}^\star\mathbf{\Theta}_1^\star \quad \mathbf{U}_2{}^\star\mathbf{\Theta}_2^\star \quad \cdots \quad \mathbf{U}_K^\star\mathbf{\Theta}_K^\star]$, plus a $d \times n$ noise matrix with i.i.d. $\mathcal{N}(0,\sigma^2)$ entries. To create Ω, we sample each entry independently with probability p.

Effect of the penalty parameter. In our first experiment we study the number of clusters obtained by Fsc as a function of λ, with the default settings above. Figure 2 shows, consistent with our discussion in Section 6, that if $\lambda = 0$, Fsc assigns each point to its own cluster. As λ increases, subspaces start fusing together up to the point where if λ is too large, Fsc fuses all subspaces into one, and all data gets clustered together. Next we study performance. Figure 3 shows that there is a wide range of values of λ that produce low error, which shows that Fsc is quite stable. Notice that the error increases if λ is too small or too large. This is consistent with our previous experiment, showing that this is because such extreme values of λ produce too few or too many clusters.

Effect of noise. Figure 3 shows that Fsc performs as well as the state-of-the-art with low-noise, but considerably better in the high-noise regime. Recall that λ quantifies the tradeoff between how accurately we want to represent each point $\mathbf{x}_i$ (the first term in (4) and (6)), and how close subspaces from

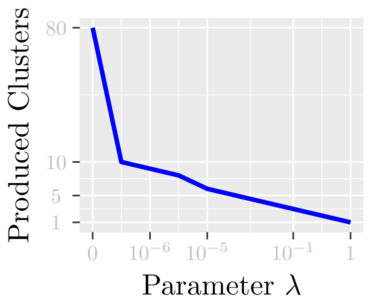

Figure 2: Number of clusters obtained by Fsc as a function of the parameter λ in (4) and (6).

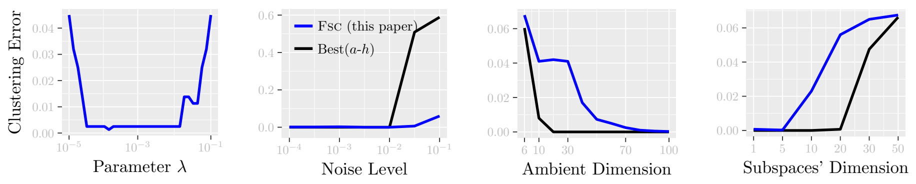

Figure 3: Clustering error of Fsc and the best algorithm among (a)-(h) in each trial. Notice the different scales: even in the worst-case (subspace's dimension $\mathrm{r} = 20$), the gap between Fsc and the best algorithm (on each trial) among (a)-(h) is less than 6%.

different points will be (second term), which in turn determines how subspaces fuse together, or equivalently, how many subspaces we will obtain. If data is completely noiseless, we expect to represent each point very accurately, and so we can use a smaller λ (giving more weight to the first term). On the other hand, if data is noisy, we expect to represent each point within the noise level, and so we can use a larger λ. As a rule of thumb, we can use λ inversely proportional to the noise level σ.

Effect of dimensionality. It is well-documented that data in lower-dimensional subspaces are easier to cluster [28, 31–34, 41]. In the extreme case, clustering 1-dimensional subspaces requires a simple co-linearity test, and is theoretically possible with as little as 2 samples per column [41]. In contrast, no existing algorithm can successfully cluster $(\mathrm{d}-1)$-dimensional subspaces (hyperplanes), which is actually impossible even if one entry per column is missing [41]. Of course, being low-dimensional is relative to the ambient dimension: a 10-dimensional subspace is a hyperplane in $\mathbb{R}^{11}$, but low-dimensional in $\mathbb{R}^{1000}$. In this experiment we test Fsc as a function of the *low-dimensionality* of the subspaces, i.e., the gap between the ambient dimension d and the subspaces' dimension r. First we fix $\mathrm{r} = 5$, and compute error as a function of d. As d grows, this subspace becomes lower and lower-dimensional. Then we turn things around, fixing $\mathrm{d} = 100$ and varying r. As r grows, the subspace becomes higher and higher-dimensional. The results are in Figure 3. Unfortunately, Fsc seems more sensitive to high-dimensionality than the state-of-the-art. However, pay attention to the scale: even in the worst-case ($\mathrm{r} = 20$), the gap between Fsc and the best algorithm (on each trial) among (a)-(h) is less than 6%.

Effect of the number of subspaces and data points. Figure 4 shows that like the state-of-the-art, Fsc is very robust to the number of subspaces K. Recall that in our default settings, $\mathrm{r} = 5$, so $\mathrm{K} \geq 20$ produces a full-rank data matrix $\mathbf{X}$. Figure 4 also evaluates the performance of Fsc as a function of the columns per subspace $\mathrm{n_k}$. Since $\mathrm{r} = 5$, $\mathrm{n_k} = 6$ is information-theoretically necessary for subspace clustering. Fsc only requires little more than that to perform as well as the state-of-the-art.

Effect of missing data. There is a tradeoff between the number of columns per subspace $\mathrm{n_k}$ and the samples per column p required for subspace clustering [41]. The larger $\mathrm{n_k}$, the lower p may be, and vice versa. Figure 4 evaluates

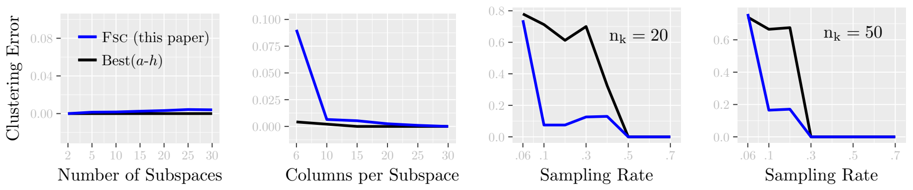

Figure 4: Clustering error of Fsc and the best algorithm among (a)-(h) in each trial. Notice the different scales. With full-data Fsc is rarely outperformed by other algorithms, and only by a marginal amount. In contrast, when data is missing, Fsc outperforms other algorithms by a wide margin. For example, with $n_k = 20$ and $p = 0.1$, Fsc achieves 7.5% error, while the next best algorithm achieves 71.25%.

the performance of Fsc as a function of p with $n_k = 20, 50$ (few and many columns). Notice that if the sampling rate p is high (few missing data), then Fsc performs as well as the state-of-the-art, and much better if the sampling rate p is low (many missing data); see for example $n_k = 20$ and $p = 0.1$, where the best among algorithms (a)-(h) gets 71.25% error, which is as good as random guessing (because there are $K = 4$ subspaces in our default settings). In contrast, Fsc gets 7.5% error. Notice that $p = 0.1$ is very close to the exact information-theoretic minimum sampling rate $p = (r+1)/d = 0.06$ [41]. Similar to noise, if there is much missing data the first term in (4) and (6) will carry less weight, which we can compensate by making λ smaller.

7.2 Real Data Experiments

Motion Segmentation. It is well-known that the locations over time of a rigidly moving object approximately lie in a 3-dimensional affine subspace [2, 3] (which can be thought as a 4-dimensional subspace whose fourth component accounts for the offset). Hence, by tracking points in a video, and subspace clustering them, we can segment the multiple moving objects appearing in the video. In this experiment we test Fsc on this task, using the Hopkins 155 dataset [45], containing sequences of points tracked over time in 155 videos; Figure 5 shows a sample frame. Each video contains either $K = 2$ or $K = 3$ objects. On average, each object is tracked on $n_k = 133$ points (described by two coordinates) over 29 frames, producing vectors in ambient dimension $d = 58$.

First we test Fsc with full data. Figure 6 shows the results of 10 videos. We can see a consistent behavior between Fsc and the state-of-the-art. Pay attention to the scale, showing that Fsc's accuracy is only about 3% lower than the best among (a)-(h). Figure 6 also shows the average clustering error (of all videos) as a function of the amount of missing data (induced uniformly at random). Consistent with our simulations, Fsc dramatically outperforms the state-of-the-art in the low-sampling regime (many missing data); for example, with $p = 0.1$, the best among algorithms (a)-(h) gets 52.95% error, which is close to random guessing (because on average, there are $K = 2$ subspaces in each video). In contrast, Fsc achieves 15.03% error. Notice that $p = 0.1$ is very close

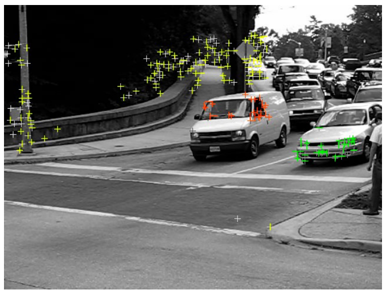

Figure 5: Points tracked in a frame from Hopkins 155 [45].

to the exact information-theoretic minimum sampling rate $p = (r+1)/d = 0.086$ [41].

Face Clustering. It has been shown that the vectorized images of the same person under different illuminations lie near a 9-dimensonal subspace [4]. In this experiment we evaluate the performance of FSC at clustering faces of multiple individuals, using the Yale B dataset [46], containing a total of 2432 images, each of size 48×42, evenly distributed amongst 38 individuals; Figure 7 contains a few samples. To compare things vis à vis, before clustering, we use robust PCA [47] on each cluster, to remove outliers; this is a widely used preprocessing step [9,26,30,32]. In each of 30 trials, we select K people uniformly at random, and record the clustering error. Figure 6 shows that FSC is quite competitive. Notice that there is only a small gap of 5% error between FSC and the best algorithm (for each trial) amongst (a)-(h), which in most cases was SSC. Figure 6 also shows the average clustering error as a function of the amount of missing data (induced uniformly at random), with K fixed to 6 people. Again, FSC outperforms the state-of-the-art in the low-sampling regime (many missing data). For example, with $p = 0.1$ FSC gets 25.7% error, while the next best algorithm gets 69.79%. Notice that $p = 0.1$ is quite close to the exact information-theoretic necessary $p = (r+1)/d = 0.005$ [41].

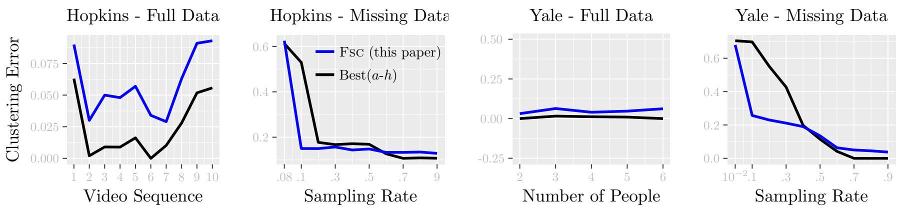

Figure 6: Clustering error on real datasets. Comparing FSC vs. the best algorithm among (a)-(h) in each trial. Notice the different scales. With full-data FSC is rarely outperformed by other algorithms, and only by a marginal amount (only about 3% lower than the best among (a)-(h)). In contrast, when data is missing, FSC outperforms other algorithms by a wide margin. For example, in the Yale B experiment with $p = 0.1$, FSC gets 25.7% error, while the next best algorithm gets 69.79%.

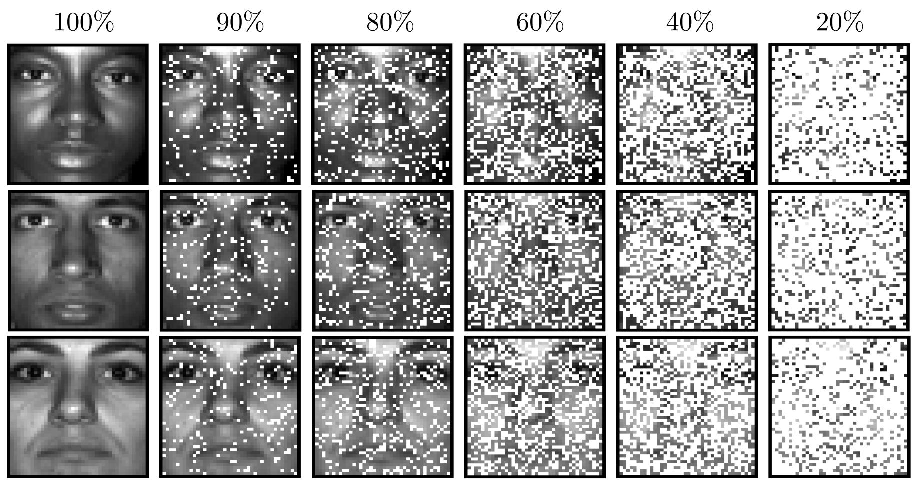

Figure 7: Sample images from the Yale B dataset [46] under decreasing sampling rates p. Fsc achieves ≈ 75% clustering accuracy with as little as p = 10% observations. The next best algorithm only achieves ≈ 30%.

8 Future Directions

This paper introduces Fsc, and shows its great potential to handle missing data. However, the term $\mathbf{U}_i^\mathsf{T}\mathbf{U}_i$ in $\mathbf{P}_i$ may become ill-conditioned (especially if the rank is over-estimated), and its inversion is computationally expensive. Also notice that the storage and computational complexity of Fsc is polynomial in the number of data points n. Exciting directions for future work that are out of the scope of this paper will address these caveats using updates over the Grassmannian, as is done in [48] for subspace tracking, as well as greedy, adaptive, and data-driven variants that quickly fuse promising subspaces within a confidence interval, in order to improve complexity.

References

[1] T. Hastie and P. Simard, *Metrics and models for handwritten character recognition*, Statistical Science, 1998.

[2] C. Tomasi and T. Kanade, *Shape and motion from image streams under orthography*, International Journal of Computer Vision, 1992.

[3] K. Kanatani, *Motion segmentation by subspace separation and model selection*, IEEE International Conference in Computer Vision, 2001.

[4] R. Basri and D. Jacobs, *Lambertian reflection and linear subspaces*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 2003.

[5] J. Rennie and N. Srebro, *Fast maximum margin matrix factorization for collaborative prediction*, International Conference on Machine Learning, 2005.

[6] G. Chen and G. Lerman, *Spectral curvature clustering (SCC)* International Journal of Computer Vision, 2009.

[7] B. Eriksson, P. Barford, J. Sommers and R. Nowak, *DomainImpute: Inferring unseen components in the Internet*, IEEE INFOCOM Mini-Conference, 2011.

[8] A. Zhang, N. Fawaz, S. Ioannidis and A. Montanari, *Guess who rated this movie: Identifying users through subspace clustering*, Uncertainty in Artificial Intelligence, 2012.

[9] E. Elhamifar and R. Vidal, *Sparse subspace clustering: Algorithm, theory, and applications* IEEE Transactions on Pattern Analysis and Machine Intelligence, 2013.

[10] G. Mateos and K. Rajawat, *Dynamic network cartography: Advances in network health monitoring*, IEEE Signal Processing Magazine, 2013.

[11] G. Liu, Z. Lin and Y. Yu, *Robust subspace segmentation by low-rank representation*, International Conference on Machine Learning, 2010.

[12] R. Vidal, *Subspace clustering*, IEEE Signal Processing Magazine, 2011.

[13] G. Liu, Z. Lin, S. Yan, J. Sun, Y. Yu and Y. Ma, *Robust recovery of subspace structures by low-rank representation*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 2013.

[14] M. Soltanolkotabi, E. Elhamifar and E. Candès, *Robust subspace clustering*, Annals of Statistics, 2014.

[15] C. Qu and H. Xu, *Subspace clustering with irrelevant features via robust Dantzig selector*, Advances in Neural Information Processing Systems, 2015.

[16] X. Peng, Z. Yi and H. Tang, *Robust subspace clustering via thresholding ridge regression*, AAAI Conference on Artificial Intelligence, 2015.

[17] Y. Wang and H. Xu, *Noisy sparse subspace clustering*, International Conference on Machine Learning, 2013.

[18] Y. Wang, Y. Wang and A. Singh, *Differentially private subspace clustering*, Advances in Neural Information Processing Systems, 2015.

[19] H. Hu, J. Feng and J. Zhou, *Exploiting unsupervised and supervised constraints for subspace clustering*, IEEE Pattern Analysis and Machine Intelligence, 2015.

[20] C. You, D. Robinson and R. Vidal, *Scalable sparse subspace clustering by orthogonal matching pursuit*, IEEE Conference on Computer Vision and Pattern Recognitionm 2016.

[21] Y. Yang, J. Feng, N. Jojic, J. Yang and T. Huang, *ℓ_0-sparse subspace clustering*, European Conference on Computer Vision, 2016.

[22] B. Xin, Y. Wang, W. Gao and D. Wipf, *Data-dependent sparsity for subspace clustering*, Uncertainty in Artificial Intelligence, 2017.

[23] J. Mairal, F. Bach, J. Ponce and G. Sapiro, *Online dictionary learning for sparse coding*, International Conference on Machine Learning, 2009.

[24] R. Vidal, R. Tron and R. Hartley, *Multiframe motion segmentation with missing data using Power Factorization and GPCA* International Journal of Computer, 2008.

[25] D. Park, J. Neeman, J. Zhang, S. Sanghavi and I. Dhillon, *Preference completion: Large-scale collaborative ranking from pairwise comparisons*, International Conference on Machine Learning, 2015.

[26] C. Yang, D. Robinson and R. Vidal, *Sparse subspace clustering with missing entries*, International Conference on Machine Learning, 2015.

[27] E. Candès and B. Recht, *Exact matrix completion via convex optimization*, Foundations of Computational Mathematics, 2009.

[28] B. Eriksson, L. Balzano and R. Nowak, *High-rank matrix completion and subspace clustering with missing data*, Artificial Intelligence and Statistics, 2012.

[29] L. Balzano, R. Nowak, A. Szlam and B. Recht, *k-Subspaces with missing data*, IEEE Statistical Signal Processing, 2012.

[30] D. Pimentel-Alarcón, L. Balzano and R. Nowak, *On the sample complexity of subspace clustering with missing data*, IEEE Statistical Signal Processing, 2014.

[31] D. Pimentel-Alarcón, L. Balzano, R. Marcia, R. Nowak and R. Willett, *Group-sparse subspace clustering with missing data*, IEEE Statistical Signal Processing, 2016.

[32] E. Elhamifar, *High-rank matrix completion and clustering under self-expressive models*, Neural Information Processing Systems, 2016.

[33] G. Ongie, R. Willett, R. Nowak and L. Balzano, *Algebraic variety models for high-rank matrix completion*, International Conference on Machine Learning, 2017.

[34] D. Pimentel-Alarcón, G. Ongie, L. Balzano, R. Willett and R. Nowak, *Low algebraic dimension matrix completion*, Allerton Conference on Communication, Control, and Computing, 2017.

[35] S. Land and J. Friedman, *Variable fusion: a new method of adaptive signal regression*, Technical Report, Department of Statistics, Stanford University, 1996.

[36] R. Tibshirani, S. Rosset, J. Zhu and K. Knight, *Sparsity and smoothness via the fused lasso*, Journal of the Royal Statistical Society, 2005.

[37] X. Shen and H. Huang, *Grouping pursuit through a regularization solution surface*, Journal of the American Statistical Association, 2010.

[38] T. Hocking, A. Joulin and F. Bach, *Clusterpath: An algorithm for clustering using convex fusion penalties* International Conference on Machine Learning, 2011.

[39] F. Lindsten, H. Ohlsson and L. Ljung, *Clustering using sum-of-norms regularization: with application to particle filter output computation*, Statistical Signal Processing, 2011.

[40] S. Poddar and M. Jacob, *Clustering of data with missing entries using non-convex fusion penalties*, arXiv preprint, 2017.

[41] D. Pimentel-Alarcón and R. Nowak, *The information-theoretic requirements of subspace clustering with missing data*, International Conference on Machine Learning, 2016.

[42] H. Akaike, *Information theory and an extension of the maximum likelihood principle*, IEEE International Symposium on Information Theory, 1973.

[43] D. Gabay and B. Mercier, *A dual algorithm for the solution of nonlinear variational problems via finite-element approximations*, Computers & Mathematics with Applications, 1976.

[44] S. Boyd, N. Parikh, E. Chu, B. Peleato and J. Eckstein, *Distributed optimization and statistical learning via the alternating direction method of multipliers*, Foundations and Trends in Machine Learning, 2010.

[45] R. Tron and R. Vidal, *A benchmark for the comparison of 3-D motion segmentation algorithms*, IEEE Conference on Computer Vision and Pattern Recognition, 2007.

[46] A. Georghiades, P. Belhumeur and D. Kriegman, *From few to many: Illumination cone models for face recognition under variable lighting and pose*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 2001.

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

[48] L. Balzano, R. Nowak and B. Recht, *Online identification and tracking of subspaces from highly incomplete information*, Allerton Conference on Communication, Control and Computing, 2010.