A TUTORIAL ON SUBSPACE CLUSTERING

René Vidal

Johns Hopkins University

The past few years have witnessed an explosion in the availability of data from multiple sources and modalities. For example, millions of cameras have been installed in buildings, streets, airports and cities around the world. This has generated extraordinary advances on how to acquire, compress, store, transmit and process massive amounts of complex high-dimensional data. Many of these advances have relied on the observation that, even though these data sets are high-dimensional, their intrinsic dimension is often much smaller than the dimension of the ambient space. In computer vision, for example, the number of pixels in an image can be rather large, yet most computer vision models use only a few parameters to describe the appearance, geometry and dynamics of a scene. This has motivated the development of a number of techniques for finding a low-dimensional representation of a high-dimensional data set. Conventional techniques, such as Principal Component Analysis (PCA), assume that the data is drawn from a *single* low-dimensional subspace of a high-dimensional space. Such approaches have found widespread applications in many fields, e.g., pattern recognition, data compression, image processing, bioinformatics, etc.

In practice, however, the data points could be drawn from *multiple subspaces* and the *membership* of the data points to the subspaces might be unknown. For instance, a video sequence could contain several moving objects and different subspaces might be needed to describe the motion of different objects in the scene. Therefore, there is a need to simultaneously cluster the data into multiple subspaces and find a low-dimensional subspace fitting each group of points. This problem, known as *subspace clustering*, has found numerous applications in computer vision (e.g., image segmentation [1], motion segmentation [2] and face clustering [3]), image processing (e.g., image representation and compression [4]) and systems theory (e.g., hybrid system identification [5]).

A number of approaches to subspace clustering have been proposed in the past two decades. A review of methods from the data mining community can be found in [6]. This article will present methods from the machine learning and computer vision communities, including algebraic methods [7, 8, 9, 10], iterative methods [11, 12, 13, 14, 15], statistical methods [16, 17, 18, 19, 20], and spectral clustering-based methods [7, 21, 22, 23, 24, 25, 26, 27]. We review these methods, discuss their advantages and disadvantages, and evaluate their performance on the motion segmentation and face clustering problems.

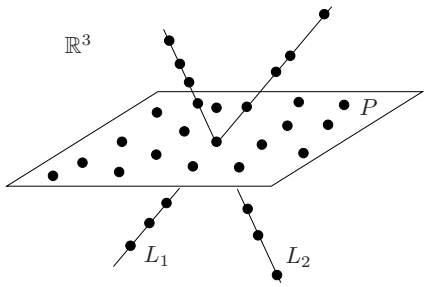


Fig. 1: A set of sample points in $\mathbb{R}^3$ drawn from a union of three subspaces: two lines and a plane.

1. THE SUBSPACE CLUSTERING PROBLEM

Consider the problem of modeling a collection of data points with a union of subspaces, as illustrated in Figure 1. Specifically, let $\{\boldsymbol{x}_j \in \mathbb{R}^D\}_{j=1}^N$ be a given set of points drawn from an unknown union of $n \geq 1$ linear or affine subspaces $\{S_i\}_{i=1}^n$ of unknown dimensions $d_i = \dim(S_i)$, $0 < d_i < D$, $i = 1, \ldots, n$. The subspaces can be described as

$$S_i = \{\boldsymbol{x} \in \mathbb{R}^D : \boldsymbol{x} = \boldsymbol{\mu}_i + U_i \boldsymbol{y}\}, \quad i = 1, \ldots, n, \qquad (1)$$

where $\boldsymbol{\mu}_i \in \mathbb{R}^D$ is an arbitrary point in subspace S_i ($\boldsymbol{\mu}_i = \mathbf{0}$ for linear subspaces), $U_i \in \mathbb{R}^{D \times d_i}$ is a basis for subspace S_i and $\boldsymbol{y} \in \mathbb{R}^{d_i}$ is a low-dimensional representation for point $\boldsymbol{x}$. The goal of *subspace clustering* is to find the number of subspaces n, their dimensions $\{d_i\}_{i=1}^n$, the subspace bases $\{U_i\}_{i=1}^n$, the points $\{\boldsymbol{\mu}_i\}_{i=1}^n$ (in the case of affine subspaces), and the segmentation of the points according to the subspaces.

When the number of subspaces is equal to one, this problem reduces to finding a vector $\boldsymbol{\mu} \in \mathbb{R}^D$, a basis $U \in \mathbb{R}^{D \times d}$, a low-dimensional representation $Y = [\boldsymbol{y}_1, \ldots, \boldsymbol{y}_N] \in \mathbb{R}^{d \times N}$, and the dimension d. This problem is known as Principal Component Analysis (PCA) [28][1] and can be solved in a remarkably simple way: $\boldsymbol{\mu} = \frac{1}{N} \sum_{j=1}^N \boldsymbol{x}_j$ is the mean of the data points, (U, Y) can be obtained from the rank-d singular value decomposition (SVD) of the (mean-subtracted) data matrix $X = [\boldsymbol{x}_1 - \boldsymbol{\mu}, \boldsymbol{x}_2 - \boldsymbol{\mu}, \ldots, \boldsymbol{x}_N - \boldsymbol{\mu}] \in \mathbb{R}^{D \times N}$ as

$$U = \mathcal{U} \quad \text{and} \quad Y = \Sigma \mathcal{V}^\top, \quad \text{where} \quad X = \mathcal{U} \Sigma \mathcal{V}^\top, \qquad (2)$$

[1] The problem of matrix factorization dates back to the work Beltrami [29] and Jordan [30]. In the context of stochastic signal processing, PCA is also known as the Karhunen-Loeve transform [31]. In the applied statistics literature, PCA is also known as the Eckart-Young decomposition [32].

and d can be obtained as $d = \text{rank}(X)$ with noise free data, or using model selection techniques when the data is noisy [28].

When $n > 1$, the subspace clustering problem becomes significantly more difficult due to a number of challenges.

1. First, there is a strong coupling between data segmentation and model estimation. Specifically, if the segmentation of the data were known, one could easily fit a single subspace to each group of points using standard PCA. Conversely, if the subspace parameters were known, one could easily find the data points that best fit each subspace. In practice, neither the segmentation of the data nor the subspace parameters are known and one needs to solve both problems simultaneously.
2. Second, the distribution of the data inside the subspaces is generally unknown. If the data within each subspace is distributed around a cluster center and the cluster centers for different subspaces are far apart, then the subspace clustering problem reduces to the simpler and well studied central clustering problem, where the data is distributed around multiple cluster centers. On the other hand, if the distribution of the data points in the subspaces is arbitrary and there are many points close to the intersection of the subspaces, then the problem cannot be solved with central clustering techniques.
3. Third, the relative position of the subspaces can be arbitrary. When two subspaces intersect or are very close, the subspace clustering problem becomes very hard. However, when the subspaces are *disjoint* or *independent*,[2] the subspace clustering problem is less difficult.
4. The fourth challenge is that the data can be corrupted by noise, missing entries, outliers, etc. Such nuisances can cause the estimated subspaces to be completely wrong. While robust estimation techniques have been developed for the case of a single subspace, the case of multiple subspaces is not as well understood.
5. Last, but not least, is the issue of model selection. In classical PCA, the only parameter is the dimension of the subspace, which can be found by searching for the subspace of smallest dimension that fits the data with a given accuracy. In the case of multiple subspaces, one can fit the data with N different subspaces of dimension one, namely one subspace per data point, or with a single subspace of dimension D. Obviously, neither solution is satisfactory. The challenge is to find a model selection criteria that favors a small number of subspaces of small dimension.

In what follows, we present a number of subspace clustering algorithms and show how they try to address these challenges.

[2] n linear subspaces are disjoint if every two subspaces intersect only at the origin. n linear subspaces are independent if the dimension of their sum is equal to the sum of their dimensions. Independent subspaces are disjoint, but the converse is not always true. n affine subspaces are disjoint (independent) if so are the corresponding linear subspaces in homogeneous coordinates.

2. SUBSPACE CLUSTERING ALGORITHMS

2.1. Algebraic Algorithms

We first review two algebraic algorithms for clustering noise free data drawn from multiple linear subspaces, i.e., $\boldsymbol{\mu}_i = \mathbf{0}$. The first algorithm is based on linear algebra, specifically matrix factorization, and is applicable only to independent subspaces. The second one is based on polynomial algebra and is applicable to any kind of subspaces. While these algorithms are designed for linear subspaces, in the case of noiseless data they can also be applied to affine subspaces by considering an affine subspace of dimension d in $\mathbb{R}^D$ as a linear subspace of dimension $d+1$ in $\mathbb{R}^{D+1}$. Also, while these algorithms operate under the assumption of noise free data, they provide great insights into the geometry and algebra of the subspace clustering problem. Moreover, they can be extended to handle moderate amounts of noise, as we shall see.

Matrix factorization-based algorithms. These algorithms obtain the segmentation of the data from a low-rank factorization of the data matrix X. Hence, they are a natural extension of PCA from one to multiple independent linear subspaces.

Specifically, let $X_i \in \mathbb{R}^{D \times N_i}$ be the matrix containing the N_i points in subspace i. The columns of the data matrix can be sorted according to the n subspaces as $[X_1, X_2, \ldots, X_n] = X\Gamma$, where $\Gamma \in \mathbb{R}^{N \times N}$ is an unknown permutation matrix. Because each matrix X_i is of rank d_i, it can be factorized as

$$X_i = U_i Y_i \quad i = 1, \ldots, n, \tag{3}$$

where $U_i \in \mathbb{R}^{D \times d_i}$ is an orthogonal basis for subspace i and $Y_i \in \mathbb{R}^{d_i \times N_i}$ is the low-dimensional representation of the points with respect to U_i. Therefore, if the subspaces are independent, then $r = \text{rank}(X) = \sum_{i=1}^{n} d_i \leq \min\{D, N\}$ and

$$X\Gamma = \begin{bmatrix} U_1, U_2, \cdots, U_n \end{bmatrix} \begin{bmatrix} Y_1 & & & \\ & Y_2 & & \\ & & \ddots & \\ & & & Y_n \end{bmatrix} \triangleq UY, \tag{4}$$

where $U \in \mathbb{R}^{D \times r}$ and $Y \in \mathbb{R}^{r \times N}$. The subspace clustering problem is then equivalent to finding a permutation matrix Γ such that $X\Gamma$ admits a rank-r factorization into a matrix U and a *block diagonal* matrix Y. This idea is the basis for the algorithms of Boult and Brown [7], Costeira and Kanade [8] and Gear [9], which compute Γ from the SVD of X [7, 8] or from the row echelon canonical form of X [9].

Specifically, the Costeira and Kanade algorithm proceeds as follows. Let $X = \mathcal{U}\Sigma\mathcal{V}^\top$ be the rank-r SVD of the data matrix, i.e., $\mathcal{U} \in \mathbb{R}^{D \times r}$, $\Sigma \in \mathbb{R}^{r \times r}$ and $\mathcal{V} \in \mathbb{R}^{N \times r}$. Also, let

$$Q = \mathcal{V}\mathcal{V}^\top \in \mathbb{R}^{N \times N}. \tag{5}$$

As shown in [33, 2], the matrix Q is such that

$$Q_{jk} = 0 \quad \text{if points } j \text{ and } k \text{ are in different subspaces.} \tag{6}$$

In the absence of noise, equation (6) can be immediately used to obtain the segmentation of the data by thresholding and sorting the entries of Q.[3] For instance, [8] obtains the segmentation by maximizing the sum of the squared entries of Q in different groups, while [34] finds the groups by thresholding the most discriminant rows of Q. However, as noted in [35, 33], this thresholding process is very sensitive to noise. Also, the construction of Q requires knowledge of the rank of X and using the wrong rank can lead to very poor results [9].

Wu et al. [35] use an agglomerative process to reduce the effects of noise. The entries of Q are first thresholded to obtain an initial over-segmentation of the data. A subspace is then fit to each group G_i and two groups are merged when some distance between their subspaces is below a threshold. Kanatani [33, 36] uses the Geometric Akaike Information Criterion [37] (G-AIC) to decide when to merge two groups. Specifically, the G-AIC of G_i and G_j as separate groups, $\text{G-AIC}_{G_i,G_j}$, is compared to their G-AIC as a single group, $\text{G-AIC}_{G_i \cup G_j}$, and used to scale the entries of Q as follows

$$\hat{Q}_{jk} = \frac{\text{G-AIC}_{G_j,G_k}}{\text{G-AIC}_{G_j \cup G_k}} \max_{\boldsymbol{x}_l \in G_j, \boldsymbol{x}_m \in G_k} |Q_{lm}| . \tag{7}$$

While these approaches indeed reduce the effect of noise, in practice they are not effective because the equation $Q_{jk} = 0$ holds only when the subspaces are independent. In the case of dependent subspaces, one can use the subset of the columns of $\mathcal{V}$ that do not span the intersections of the subspaces. Unfortunately, we do not know which columns to choose a priori. Zelnik-Manor and Irani [38] propose to use the top columns of $\mathcal{V}$ to define Q. However, this heuristic is not provably correct. Another issue with factorization-based algorithms is that, with a few exceptions, they do not provide a method for computing the number of subspaces, n, and their dimensions, $\{d_i\}_{i=1}^n$. The first exception is when n is known. In this case, d_i can be computed from each group after the segmentation has been obtained. The second exception is for independent subspaces of equal dimension d. In this case $\text{rank}(X) = nd$, hence we may determine n when d is known or vice versa.

Generalized PCA (GPCA). GPCA (see [10, 39]) is an algebraic-geometric method for clustering data lying in (not necessarily independent) linear subspaces. The main idea behind GPCA is that one can fit a union of n subspaces with a set of polynomials of degree n, whose derivatives at a point give a vector normal to the subspace containing that point. The segmentation of the data is then obtained by grouping these normal vectors using several possible techniques. More specifically, the GPCA algorithm proceeds as follows.

The first step, which is not strictly needed, is to project the data points onto a subspace of $\mathbb{R}^D$ of dimension $r = d_{\max}+1$, where $d_{\max} = \max\{d_1, \ldots, d_n\}$.[4] The rationale behind this step is as follows. Since the maximum dimension of each subspace is $d_{\max}$, with probability one a projection onto a generic subspace of $\mathbb{R}^D$ of dimension $d_{\max}+1$ preserves the number and dimensions of the subspaces. As a byproduct, the dimensionality of the problem is reduced to clustering subspaces of dimension at most $d_{\max}$ in $\mathbb{R}^{d_{\max}+1}$. As we shall see, this will be very important to reduce the computational complexity of the GPCA algorithm. With an abuse of notation, we will denote both the original and projected subspaces as S_i, and both the original and projected data matrix as

$$X = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N] \in \mathbb{R}^{D \times N} \text{ or } \mathbb{R}^{r \times N}. \tag{8}$$

The second step is to fit a homogeneous polynomial of degree n to the (projected) data. The rationale behind this step is as follows. Imagine, for instance, that the data came from the union of two planes in $\mathbb{R}^3$, each one with normal vector $\boldsymbol{b}_i \in \mathbb{R}^3$. The union of the two planes can be represented as the set of points such that $p(\boldsymbol{x}) = (\boldsymbol{b}_1^\top \boldsymbol{x})(\boldsymbol{b}_2^\top \boldsymbol{x}) = 0$. This equation is nothing but the equation of a conic of the form

$$c_1 x_1^2 + c_2 x_1 x_2 + c_3 x_1 x_3 + c_4 x_2^2 + c_5 x_2 x_3 + c_6 x_3^2 = 0. \tag{9}$$

Imagine now that the data came from the plane $\boldsymbol{b}^\top \boldsymbol{x} = 0$ or the line $\boldsymbol{b}_1^\top \boldsymbol{x} = \boldsymbol{b}_2^\top \boldsymbol{x} = 0$. The union of the plane and the line is the set of points such that $p_1(\boldsymbol{x}) = (\boldsymbol{b}^\top \boldsymbol{x})(\boldsymbol{b}_1^\top \boldsymbol{x}) = 0$ and $p_2(\boldsymbol{x}) = (\boldsymbol{b}^\top \boldsymbol{x})(\boldsymbol{b}_2^\top \boldsymbol{x}) = 0$. More generally, data drawn from the union of n subspaces of $\mathbb{R}^r$ can be represented with polynomials of the form $p(\boldsymbol{x}) = (\boldsymbol{b}_1^\top \boldsymbol{x}) \cdots (\boldsymbol{b}_n^\top \boldsymbol{x}) = 0$, where the vector $\boldsymbol{b}_i \in \mathbb{R}^r$ is orthogonal to S_i. Each polynomial is of degree n in $\boldsymbol{x}$ and can be written as $\boldsymbol{c}^\top \nu_n(\boldsymbol{x})$, where $\boldsymbol{c}$ is the vector of coefficients and $\nu_n(\boldsymbol{x})$ is the vector of all monomials of degree n in $\boldsymbol{x}$. There are $M_n(r) = \binom{n+r-1}{n}$ monomials.

In the case of noiseless data, the vector of coefficients $\boldsymbol{c}$ of each polynomial can be computed from

$$\boldsymbol{c}^\top [\nu_n(\boldsymbol{x}_1),\ \nu_n(\boldsymbol{x}_2), \cdots, \nu_n(\boldsymbol{x}_N)] = \boldsymbol{c}^\top \boldsymbol{V}_n = \boldsymbol{0}^\top \tag{10}$$

and the number of polynomials is simply the dimension of the null space of $\boldsymbol{V}_n$. While in general the relationship between the number of subspaces, n, their dimensions, $\{d_i\}_{i=1}^n$, and the number of polynomials involves the theory of Hilbert functions [40], in the particular case where all the dimensions are equal to d, and $r = d+1$, there is a unique polynomial that fits the data. This fact can be exploited to determine both n and d. For example, given d, n can be computed as

$$n = \min\{i : \text{rank}(\boldsymbol{V}_i) = M_i(r) - 1\}. \tag{11}$$

In the case of data contaminated with small to moderate amounts of noise, the polynomial coefficients (10) can be found using least squares – the vectors $\boldsymbol{c}$ are the left singular vectors of $\boldsymbol{V}_n$ corresponding to the smallest singular values. To handle larger amounts of noise in the estimation of the polynomial coefficients, one can resort to techniques from robust statistics [20] or rank minimization [41]. Model selection techniques can be used to determine the rank of $\boldsymbol{V}_n$ and

[3] Boult and Brown [7] use instead the eigenvectors of Q to find the segmentation by using spectral clustering, as we will see in Section 2.4.

[4] The value of r is determined using model selection techniques when the subspace dimensions are unknown.

hence the number of polynomials, as shown in [42]. Model selection techniques can also be used to determine the number of subspaces of equal dimensions in (11), as shown in [10]. However, determining n and $\{d_i\}_{i=1}^n$ for subspaces of different dimensions from noisy data remains very challenging. The reader is referred to [43] for a model selection criteria called *minimum effective dimension*, which measures the complexity of fitting n subspaces of dimensions $\{d_i\}_{i=1}^n$ to a given dataset within a certain tolerance, and to [42, 40] for algebraic relationships among n, $\{d_i\}_{i=1}^n$ and the number of polynomials that could be used for model selection purposes.

The last step is to compute the normal vectors $\boldsymbol{b}_i$ from the vector of coefficients $\boldsymbol{c}$. This can be done by taking the derivatives of the polynomials at a data point. For example, if $n = 2$ we have $\nabla p(\boldsymbol{x}) = (\boldsymbol{b}_2^\top \boldsymbol{x})\boldsymbol{b}_1 + (\boldsymbol{b}_1^\top \boldsymbol{x})\boldsymbol{b}_2$. Thus if $\boldsymbol{x}$ belongs to the first subspace, then $\nabla p(\boldsymbol{x}) \sim \boldsymbol{b}_1$. More generally, in the case of n subspaces we have $p(\boldsymbol{x}) = (\boldsymbol{b}_1^\top \boldsymbol{x}) \cdots (\boldsymbol{b}_n^\top \boldsymbol{x})$ and $\nabla p(\boldsymbol{x}) \sim \boldsymbol{b}_i$ if $\boldsymbol{x} \in S_i$. We can use this result to obtain the set of all normal vectors to S_i from the derivatives of all the polynomials at $\boldsymbol{x} \in S_i$. This gives us a basis for the orthogonal complement of S_i from which we can obtain a basis U_i for S_i. Therefore, if we knew one point per subspace, then we could immediately compute the n subspace bases from the derivatives of the polynomials. Given the subspace basis, we could obtain the segmentation by assigning each data point to its closest subspace. There are several ways of choosing one point per subspace. A simple method is to choose any point in the dataset as the first point. The basis for this subspace can hence be computed as well as the points that belong to this subspace. Such points can then be removed from the data and a second point can be chosen and so on. In Section 2.4 we will describe an alternative method based on spectral clustering.

The first advantage of GPCA is that it is an algebraic algorithm, thus it is computationally cheap when n and d are small. Second, intersections between subspaces are automatically allowed, hence GPCA can deal with both independent and dependent subspaces. Third, in the noiseless case, it does not require the number of subspaces or their dimensions to be known beforehand. Specifically, the theory of Hilbert functions [40] may be used to determine n and $\{d_i\}$.

The first drawback of GPCA is that its complexity increases exponentially with the n and $\{d_i\}$. Specifically, each vector $\boldsymbol{c}$ is of dimension $\mathrm{O}(M_n(r))$, while there are only $\mathrm{O}(r\sum(r - d_i))$ unknowns in the n sets of normal vectors. Second, the vector $\boldsymbol{c}$ is computed using least-squares, thus the computation of $\boldsymbol{c}$ is sensitive to outliers. Third, the least-squares fit does not take into account nonlinear constraints among the entries of $\boldsymbol{c}$ (recall that p must factorize as a product of linear factors). These issues cause the performance of GPCA to deteriorate as n increases. Fourth, the method in [40] to determine n and $\{d_i\}$ does not handle noisy data. Fifth, while GPCA can be applied to affine subspaces by using the data in homogeneous coordinates, in practice it does not work very well when the data is contaminated with noise.

2.2. Iterative Methods

A very simple way of improving the performance of algebraic algorithms in the case of noisy data is to use iterative refinement. Intuitively, given an initial segmentation, we can fit a subspace to each group using classical PCA. Then, given a PCA model for each subspace, we can assign each data point to its closest subspace. By iterating these two steps till convergence, we can obtain a refined estimate of the subspaces and of the segmentation. This is the basic idea behind the K-planes [11] and K-subspaces [12, 13] algorithms, which are generalizations of the K-means algorithm [44] from data distributed around cluster centers to data drawn from hyperplanes and affine subspaces of any dimensions, respectively.

The K-subspaces algorithm proceeds as follows. Let $w_{ij} = 1$ if point j belongs to subspace i and $w_{ij} = 0$ otherwise. Referring back to (1), assume that the number of subspaces n and the subspace dimensions $\{d_i\}$ are known. Our goal is to find the points $\{\boldsymbol{\mu}_i \in \mathbb{R}^D\}_{i=1}^n$, the subspace bases $\{U_i \in \mathbb{R}^{D\times d_i}\}_{i=1}^n$, the low-dimensional representations $\{Y_i \in \mathbb{R}^{d_i \times N_i}\}_{i=1}^n$ and the segmentation of the data $\{w_{ij}\}_{i=1,\ldots,n}^{j=1,\ldots,N}$. We can do so by minimizing the sum of the squared distances from each data point to its own subspace

$$\begin{aligned} \min_{\{\mu_i\},\{U_i\},\{\boldsymbol{y}_i\},\{w_{ij}\}} \quad & \sum_{j=1}^{N}\sum_{i=1}^{n} w_{ij}\|\boldsymbol{x}_j - \boldsymbol{\mu}_i - U_i\boldsymbol{y}_j\|^2 \\ \text{subject to} \quad & w_{ij} \in \{0,1\} \text{ and } \sum_{i=1}^{n} w_{ij} = 1. \end{aligned} \tag{12}$$

Given $\{\boldsymbol{\mu}_i\}$, $\{U_i\}$, $\{\boldsymbol{y}_j\}$, the optimal value for w_{ij} is

$$w_{ij} = \begin{cases} 1 & \text{if } i = \arg\min_{k=1,\ldots,n} \|\boldsymbol{x}_j - \boldsymbol{\mu}_k - U_k\boldsymbol{y}_j\|^2 \\ 0 & \text{else} \end{cases}. \tag{13}$$

Given $\{w_{ij}\}$, the cost function in (12) decouples as the sum of n cost functions, one per subspace. The optimal values for $\boldsymbol{\mu}_i$, U_i, $\boldsymbol{y}_j$ are hence obtained by applying PCA to each group of points. The K-subspaces algorithm then proceeds by alternating between assigning points to subspaces and re-estimating the subspaces. Since the number of possible assignments of points to subspaces is finite, the algorithm is guaranteed to converge to a local minimum in a finite number of iterations.

The main advantage of K-subspaces is its simplicity, since it alternates between assigning points to subspaces and estimating the subspaces via PCA. Another advantage is that it can handle both linear and affine subspaces explicitly. The third advantage is that it converges to a local optimum in a finite number of iterations. However, K-subspaces suffers from a number of drawbacks. First, its convergence to the global optimum depends on good initialization. If a random initialization is used, several restarts are often needed to find the global optimum. In practice, one may use any of the algorithms described in this paper to reduce the number of restarts

needed. We refer the reader to [45, 22] for two additional initialization methods. Second, K-subspaces is sensitive to outliers, partly due to the use of the 2-norm. This issue can be addressed by using a robust norm, such as the 1-norm, as done by the median K-flats algorithm [15]. However, this results in a more complex algorithm, which requires solving a robust PCA problem at each iteration. Alternative, one can resort to nonlinear minimization techniques, which are only guaranteed to converge to a local minimum. Third, K-subspaces requires n and $\{d_i\}$ to be known beforehand. One possible avenue to be explored is to use the model selection criteria for mixtures of subspaces proposed in [43]. We refer the reader to [46] for a more detailed analysis of some of this issues and to [45] for a theoretical study on the conditions for the existence of a solution to the optimization problem in (12).

2.3. Statistical Methods

The approaches described so far seek to cluster the data according to multiple subspaces by using mostly algebraic and geometric properties of a union of subspaces. While these approaches can handle noise in the data, they do not make explicit assumptions about the distribution of the data inside the subspaces or about the distribution of the noise. Therefore, the estimates they provide are not optimal, e.g., in a maximum likelihood (ML) sense. To address this issue, we need to define a proper generative model for the data in the subspaces.

Mixtures of Probabilistic PCA (MPPCA). Resorting back to the geometric PCA model (1), Probabilistic PCA (PPCA) [47] assumes that the data within a subspace S is generated as

$$\boldsymbol{x} = \boldsymbol{\mu} + U\boldsymbol{y} + \boldsymbol{\epsilon}, \tag{14}$$

where $\boldsymbol{y}$ and $\boldsymbol{\epsilon}$ are independent zero-mean Gaussian random vectors with covariance matrices I and $\sigma^2 I$, respectively. Therefore, $\boldsymbol{x}$ is also Gaussian with mean $\boldsymbol{\mu}$ and covariance matrix $\Sigma = UU^\top + \sigma^2 I$. It can be shown that the ML estimate of $\boldsymbol{\mu}$ is the mean of the data, and the ML estimates of U and σ can be obtained from the SVD of the data matrix X.

PPCA can be naturally extended to be a generative model for a union of subspaces $\cup_{i=1}^n S_i$ by using a Mixture of PPCA (MPPCA) [16] model. Let $G(\boldsymbol{x}; \boldsymbol{\mu}, \Sigma)$ be the probability density function of a D-dimensional Gaussian with mean $\boldsymbol{\mu}$ and covariance matrix Σ. MPPCA uses a mixture of Gaussians

$$p(\boldsymbol{x}) = \sum_{i=1}^{n} \pi_i G(\boldsymbol{x}; \boldsymbol{\mu}_i, U_i U_i^\top + \sigma_i^2 I), \tag{15}$$

where the parameter π_i, called the mixing proportion, represents the a priori probability of drawing a point from subspace S_i. The ML estimates of the parameters of this mixture model can be found using Expectation Maximization (EM) [48]. EM is an iterative procedure that alternates between data segmentation and model estimation. Specifically, given initial values for the model parameters $\theta_i = (\boldsymbol{\mu}_i, U_i, \sigma_i, \pi_i)$, in the E-step the probability that $\boldsymbol{x}_j$ belongs to subspace i is computed as

$$p_{ij} = \frac{G(\boldsymbol{x}_j; \boldsymbol{\mu}_i, U_i U_i^\top + \sigma_i^2 I)\pi_i}{p(\boldsymbol{x}_j)}, \tag{16}$$

and in the M-step the p_{ij}'s are used to recompute the subspace parameters θ_i using PPCA. Specifically, π_i and μ_i are updated as

$$\tilde{\pi}_i = \frac{1}{N}\sum_{j=1}^{N} p_{ij}, \; \tilde{\boldsymbol{\mu}}_i = \frac{1}{N\tilde{\pi}_i}\sum_{j=1}^{N} p_{ij}\boldsymbol{x}_j, \tag{17}$$

and σ_i and U_i are updated from the SVD of

$$\tilde{\Sigma}_i = \frac{1}{N\tilde{\pi}_i}\sum_{j=1}^{N} p_{ij}(\boldsymbol{x}_j - \tilde{\boldsymbol{\mu}}_i)(\boldsymbol{x}_j - \tilde{\boldsymbol{\mu}}_i)^\top. \tag{18}$$

These two steps are iterated until convergence to a local maxima of the log-likelihood. Notice that MPPCA can be seen as a probabilistic version of K-subspaces that uses soft assignments $p_{ij} \in [0, 1]$ rather than hard assignments $w_{ij} = \{0, 1\}$.

As in the case of K-subspaces, the main advantage of MPPCA is that it is a simple and intuitive method, where each iteration can be computed in closed form by using PPCA. Moreover, the MPPCA model is applicable to both linear and affine subspaces, and can be extended to accommodate outliers [49] and missing entries in the data points [50]. However, an important drawback of MPPCA is that the number and dimensions of the subspaces need to be known beforehand. One way to address this issue is by putting a prior on these parameters, as shown in [51]. Also, MPPCA is not optimal when the distribution of the data inside each subspace or the noise are not Gaussian. Another issue with MPPCA is that it often converges to a local maximum, hence good initialization is critical. The initialization problem can be addressed by using any of the methods described earlier for K-subspaces. For example, the Multi-Stage Learning (MSL) algorithm [17] uses the factorization method of [8] followed by the agglomerative refinement steps of [33, 36] for initialization.

Agglomerative Lossy Compression (ALC). The ALC algorithm [18] assumes that the data is drawn from a mixture of degenerate Gaussians. However, unlike MPPCA, ALC does not aim to obtain a ML estimate of the parameters of the mixture model. Instead, it looks for the segmentation of the data that minimizes the coding length needed to fit the points with a mixture of degenerate Gaussians up to a given distortion.

Specifically, the number of bits needed to optimally code N i.i.d. samples from a zero-mean D-dimensional Gaussian, i.e., $X \in \mathbb{R}^{D\times N}$, up to a distortion ε can be approximated as $\frac{N+D}{2}\log_2\det(I + \frac{D}{\varepsilon^2 N}XX^\top)$. Thus, the total number of bits for coding a mixture of Gaussians can be approximated as

$$\sum_{i=1}^{n} \frac{N_i+D}{2}\log_2\det\Big(I + \frac{D}{\varepsilon^2 N_i}X_i X_i^\top\Big) - N_i\log_2\big(\frac{N_i}{N}\big), \tag{19}$$

where $X_i \in \mathbb{R}^{N_i \times D}$ is the data from subspace i and the last term is the number of bits needed to code (losslessly) the membership of the N samples to the n groups.

The minimization of (19) over all possible segmentations of the data is, in general, an intractable problem. ALC deals with this issue by using an agglomerative clustering method. Initially, each data point is considered as a separate group. At each iteration, two groups are merged if doing so results in the greatest decrease of the coding length. The algorithm terminates when the coding length cannot be further decreased. Similar agglomerative techniques have been used in [52, 53], though with a different criterion for merging subspaces.

ALC can naturally handle noise and outliers in the data. Specifically, it is shown in [18] that outliers tend to cluster either as a single group or as small separate groups depending on the dimension of the ambient space. Also, in principle, ALC does not need to know the number of subspaces and their dimensions. In practice, however, the number of subspaces is directly related to the parameter ε. When ε is chosen to be very large, all the points could be merged into a single group. Conversely, when ε is very small, each point could end up as a separate group. Since ε is related to the variance of the noise, one can use statistics on the data to determine ε (see e.g., [33, 22] for possible methods). In cases the number of subspaces is known, one can run ALC for several values of ε, discard the values of ε that give the wrong number of subspaces, and choose the ε that results in the segmentation with the smallest coding length. This typically increases the computational complexity of the method. Another disadvantage of ALC, perhaps the major one, is that there is no theoretical proof for the optimality of the agglomerative procedure.

Random Sample Consensus (RANSAC). RANSAC [54] is a statistical method for fitting a model to a cloud of points corrupted with outliers in a statistically robust way. More specifically, if d is the minimum number of points required to fit a model to the data, RANSAC randomly samples d points from the data, fits a model to these d points, computes the residual of each data point to this model, and chooses the points whose residual is below a threshold as the inliers. The procedure is then repeated for another d sample points, until the number of inliers is above a threshold, or enough samples have been drawn. The outputs of the algorithm are the parameters of the model and the labeling of inliers and outliers.

In the case of clustering subspaces of equal dimension d, the model to be fit by RANSAC is a subspace of dimension d. Since there are multiple subspaces, RANSAC proceeds in a greedy fashion to fit one subspace at a time as follows:

1. Apply RANSAC to the original data set and recover a basis for the first subspace along with the set of inliers. All points in other subspaces are considered as outliers.
2. Remove the inliers from the current data set and repeat step 1 until all the subspaces are recovered.
3. For each set of inliers, use PCA to find an optimal basis for each subspace. Segment the data into multiple subspaces by assigning each point to its closest subspace.

The main advantage of RANSAC is its ability to handle outliers explicitly. Also, notice that RANSAC does not require the subspaces to be independent, because it computes one subspace at a time. Moreover, RANSAC does not need to know the number of subspaces beforehand. In practice, however, determining the number of subspaces depends on user defined thresholds. An important drawback of RANSAC is that its performance deteriorates quickly as the number of subspaces n increases, because the probability of drawing d inliers reduces exponentially with the number of subspaces. Therefore, the number of trials needed to find d points in the same subspace grows exponentially with the number and dimension of the subspaces. This issue can be addressed by modifying the sampling strategy so that points in the same subspace are more likely to be chosen than points in different subspaces, as shown in [55]. Another critical drawback of RANSAC is that it requires the dimension of the subspaces to be known and equal. In the case of subspaces of different dimensions, one could start from the largest to the smallest dimension or vice versa. However, those procedures suffer from a number of issues, as discussed in [20].

2.4. Spectral Clustering-Based Methods

Spectral clustering algorithms (see [56] for a review) are a very popular technique for clustering high-dimensional data. These algorithms construct an *affinity matrix* $A \in \mathbb{R}^{N \times N}$, whose jk entry measures the similarity between points j and k. Ideally, $A_{jk} = 1$ if points j and k are in the same group and $A_{jk} = 0$ if points j and k are in different groups. A typical measure of similarity is $A_{jk} = \exp(-\text{dist}_{jk}^2)$, where dist_{jk} is some distance between points j and k. Given A, the segmentation of the data is obtained by applying the K-means algorithm to the eigenvectors of a matrix $L \in \mathbb{R}^{N \times N}$ formed from A. Specifically, if $\{\boldsymbol{v}_j\}_{j=1}^N$ are the eigenvectors of L, then a subset of $n \ll N$ eigenvectors are chosen and stacked into a matrix $V \in \mathbb{R}^{N \times n}$. The K-means algorithm is then applied to the rows of V. Typical choices for L are the affinity matrix itself $L = A$, the Laplacian $L = \text{diag}(A\mathbf{1}) - A$, where $\mathbf{1}$ is the vector of all 1's, and the normalized Laplacian $L_{sym} = \text{diag}(A\mathbf{1})^{-1/2} A \text{diag}(A\mathbf{1})^{-1/2}$. Typical choices for the eigenvectors are the top n eigenvectors of the affinity or the bottom n eigenvectors of the (normalized) Laplacian, where n is the number of groups.

One of the main challenges in applying spectral clustering to the subspace clustering problem is how to define a good affinity matrix. This is because two points could be very close to each other, but lie in different subspaces (e.g., near the intersection of two subspaces). Conversely, two points could be far from each other, but lie in the same subspace. As a consequence, one cannot use the typical distance-based affinity.

In what follows, we describe several methods for building an affinity between pairs points lying in multiple subspaces. The first two methods (factorization and GPCA) are designed for linear subspaces, though they can be applied to affine subspaces by using homogeneous coordinates. The remaining methods can handle either linear or affine subspaces.

Factorization-based affinity. Interestingly, one of the first subspace clustering algorithms is based on both matrix factorization and spectral clustering. Specifically, the algorithm of Boult and Brown [7] obtains the segmentation of the data from the eigenvectors of the matrix $Q = \mathcal{V}\mathcal{V}^\top$ in (6). Since these eigenvectors are the singular vectors of X, the segmentation is obtained by clustering the rows of $\mathcal{V}$. However, recall that the affinity $A_{jk} = Q_{jk}$ has a number of issues. First, it is not necessarily the case that $A_{jk} \approx 1$ when points i and j are in the same subspace. Second, the equation $Q_{jk} = 0$ is sensitive to noise and it is valid only for independent subspaces.

GPCA-based affinity. The GPCA algorithm can also be used to define an affinity between pairs of points. Recall that the derivatives of the polynomials $p(\boldsymbol{x}_j)$ at a point $\boldsymbol{x}_j \in S_i$ provide an estimate of the normal vectors to subspace S_i. Therefore, one can use the angles between the subspaces to define an affinity as $A_{jk} = \prod_{m=1}^{\min(d_j,d_k)} \cos^2(\theta_{jk}^m)$, where θ_{jk}^m is the mth subspace angle between the bases of the estimated subspaces at points j and k, $\hat{S}_j$ and $\hat{S}_k$, respectively, for $j, k = 1, \ldots, N$. The segmentation of the data is then found by applying spectral clustering to the normalized Laplacian.

Local Subspace Affinity (LSA) and Spectral Local Best-fit Flats (SLBF). The LSA [21] and SLBF [22] algorithms are based on the observation that a point and its nearest neighbors (NNs) often belong to the same subspace. Therefore, we can fit an affine subspace $\hat{S}_j$ to each point j and its d-NNs using, e.g., PCA. In practice, we can choose $K \geq d$ NNs, hence d does not need to be known exactly: we only need an upper bound. Then, if two points j and k lie in the same subspace S_i, their locally estimated subspaces $\hat{S}_j$ and $\hat{S}_k$ should be the same, while if the two points lie in different subspaces $\hat{S}_j$ and $\hat{S}_k$ should be different. Therefore, we can use a distance between $\hat{S}_j$ and $\hat{S}_k$ to define an affinity between the two points.

The first (optional) step of the LSA and SLBF algorithms is to project the data points onto a subspace of dimension $r = \text{rank}(X)$ using the SVD of X. With noisy data, the value of r is determined using model selection techniques. In the case data drawn from linear subspaces, the LSA algorithm projects the resulting points in $\mathbb{R}^r$ onto the hypersphere $\mathbb{S}^{r-1}$.

The second step is to compute the K-NNs of each point j and to fit a local affine subspace $\hat{S}_j$ to the point and its neighbors. LSA assumes that K is specified by the user. The K-NNs are then found using the angle between pairs of data points or the Euclidean distance as a metric. PCA is then used to fit the local subspace $\hat{S}_j$. The subspace dimension d_j is determined using model selection techniques. SLBF determines both the number of neighbors K_j and the subspace $\hat{S}_j$ for point j automatically. It does so by searching for the smallest value of K_j that minimizes a certain fitting error.

The third step of LSA is to compute an affinity matrix as

$$A_{jk} = \exp\big(-\sum_{m=1}^{\min(d_j,d_k)} \sin^2(\theta_{jk}^m)\big), \tag{20}$$

where the θ_{jk}^m is the mth principal angle between the bases of subspaces $\hat{S}_j$ and $\hat{S}_k$. In the case of data drawn from affine subspaces, A_{jk} would need to be modified to also incorporate a distance between points j and k. SLBF uses an affinity matrix that is applicable to both linear and affine subspaces as

$$A_{jk} = \exp(-\hat{d}_{jk}/2\sigma_j^2) + \exp(-\hat{d}_{jk}/2\sigma_k^2), \tag{21}$$

where $\hat{d}_{jk} = \sqrt{\text{dist}(\boldsymbol{x}_j, \hat{S}_k)\text{dist}(\boldsymbol{x}_k, \hat{S}_j)}$ and $\text{dist}(\boldsymbol{x}, S)$ is the Euclidean distance from point $\boldsymbol{x}$ to subspace S. The segmentation of the data is then found by applying spectral clustering to the normalized Laplacian.

The LSA and SLBF algorithms have two main advantages when compared to GPCA. First, outliers are likely to be "rejected", because they are far from all the points and so they are not considered as neighbors of the inliers. Second, LSA requires only $\text{O}(nd_{\max})$ data points, while GPCA needs $\text{O}(M_n(d_{\max}+1))$. On the other hand, LSA has two main drawbacks. First, the neighbors of a point could belong to a different subspace. This is more likely to happen near the intersection of two subspaces. Second, the selected neighbors may not span the underlying subspace. Thus, K needs to be small enough so that only points in the same subspace are chosen and large enough so that the neighbors span the local subspace. SLBF resolves these issues by choosing the size of the neighborhood automatically.

Notice also that both GPCA and LSA are based on a linear projection followed by spectral clustering. While in principle both algorithms can use any linear projection, GPCA prefers to use the smallest possible dimension $r = d_{\max} + 1$, so as to reduce the computational complexity. On the other hand, LSA uses a slightly larger dimension $r = \text{rank}(X) \leq \sum d_i$, because if the dimension of the projection is too small (less than $\text{rank}(X)$), the projected subspaces are not independent and LSA has problems near the intersection of two subspaces. Another major difference is that LSA fits a subspace *locally* around each projected point, while GPCA uses the gradients of a polynomial that is *globally* fit to the projected data.

Locally Linear Manifold Clustering (LLMC). The LLMC algorithm [23] is also based on fitting a local subspace to a point and its K-NNs. Specifically, every point j is written as an affine combination of all other points $k \neq j$. The coefficients w_{jk} are found in closed form by minimizing the cost

$$\sum_{j=1}^{N} \|\boldsymbol{x}_j - \sum_{k \neq j} w_{jk}\boldsymbol{x}_k\|^2 = \|(I - W)X^\top\|_F^2, \tag{22}$$

subject to $\sum_{k\neq j} w_{jk} = 1$ and $w_{jk} = 0$ if $\boldsymbol{x}_k$ is not a K-NN of $\boldsymbol{x}_j$. Then, the affinity matrix and the matrix L are built as

$$A = W + W^\top - W^\top W \text{ and } L = (I - W)^\top (I - W). \quad (23)$$

It is shown in [23] that when every point and its K-NNs are always in the same subspace, then there are vectors $\boldsymbol{v}$ in the null space of L with the property that $v_j = v_k$ when points j and k are in the same subspace. However, these vectors are not the only vectors in the null space and spectral clustering is not directly applicable. In this case, a procedure for properly selecting linear combinations of the eigenvectors of L is needed, as discussed in [23].

A first advantage of LLMC is its robustness to outliers This is because, as in the case of LSA and SLBF, outliers are often far from the inliers, hence it is unlikely that they are chosen as neighbors of the inliers. Another important advantage of LLMC is that it is also applicable to non-linear subspaces, while all the other methods discussed so far are only applicable to linear (or affine) subspaces. However, LLMC suffers from the same disadvantage of LSA, namely that it is not always the case that a point and its K-NNs are in the same subspace, especially when the subspaces are not independent. Also, properly choosing the number of nearest neighbors is a challenge. These issues could be resolved by choosing the neighborhood automatically, as done by SLBF.

Sparse Subspace Clustering (SSC). SSC [24, 25] is also based on the idea of writing a data point as a linear (affine) combination of neighboring data points. However, the key difference with LSA, SLBF and LLMC is that, instead of choosing neighbors based on the angular or Euclidean distance between pairs of points (which can lead to errors in choosing the neighbors), the neighbors can be any other points in the data set. In principle, this leads to an ill-posed problem with many possible solutions. To resolve this issue, the principle of *sparsity* is invoked. Specifically, every point is written as a *sparse* linear (affine) combination of all other data points by minimizing the number of nonzero coefficients w_{jk} subject to $\boldsymbol{x}_j = \sum_{k\neq j} w_{jk}\boldsymbol{x}_k$ (and $\sum w_{jk} = 1$ in the case of affine subspaces). Since this problem is combinatorial, a simpler ℓ_1 optimization problem is solved

$$\min_{\{w_{jk}\}} \sum_{k\neq j} |w_{jk}| \text{ s.t. } \boldsymbol{x}_j = \sum_{k\neq j} w_{jk}\boldsymbol{x}_k \text{ (and } \sum_{k\neq j} w_{jk} = 1\text{)}. \quad (24)$$

It is shown in [24] and [25] that when the subspaces are either independent or disjoint, the solution to the optimization problem in (24) is such that $w_{jk} = 0$ only if points j and k are in different subspaces. In other words, the *sparsest* representation is obtained when each point is written as a linear (affine) combination of points in its own subspace.

In the case of data contaminated by noise, the SSC algorithm does not attempt to write a data point as an exact linear (affine) combination of other points. Instead, a penalty in the 2-norm of the error is added to the ℓ_1 norm. Specifically, the sparse coefficients are found by solving the problem

$$\min_{\{w_{jk}\}} \sum_{k\neq j} |w_{jk}| + \mu\|\boldsymbol{x}_j - \sum_{k\neq j} w_{jk}\boldsymbol{x}_k\|^2 \text{ (s.t. } \sum_{k\neq j} w_{jk} = 1\text{)}, \quad (25)$$

where $\mu > 0$ is a parameter. Obviously, different solutions for $\{w_{jk}\}$ will be obtained for different choices of the parameter μ. However, we are not interested in the specific values of w_{jk}: all what matters is that, for each point j, the top nonzero coefficients come from points in the same subspace.

In the case of data contaminated with outliers, the SSC algorithm algorithm assumes that $\boldsymbol{x}_j = \sum_{k\neq j} w_{jk}\boldsymbol{x}_k + \boldsymbol{e}_j$, where the vector of outliers $\boldsymbol{e}_j$ is also sparse. The sparse coefficients and the outliers are found by solving the problem

$$\min_{\{w_{jk}\},\{\boldsymbol{e}_j\}} \sum_{k\neq j} |w_{jk}| + \|\boldsymbol{e}_j\|_1 + \mu\|\boldsymbol{x}_j - \sum_{k\neq j} w_{jk}\boldsymbol{x}_k - \boldsymbol{e}_j\|^2 \quad (26)$$

subject to $\sum_{k\neq j} w_{jk} = 1$ in the case of affine subspaces.

Given a sparse representation for each data point, the graph affinity matrix is defined as

$$A = |W| + |W^\top|. \quad (27)$$

The segmentation is then obtained by applying spectral clustering to the Laplacian.

The SSC algorithm presents several advantages with respect to all the algorithms discussed so far. With respect to factorization-based methods, the affinity in (27) is very robust to noise. This is because the solution changes continuously with the amount of noise. Specifically, with moderate amounts of noise the top nonzero coefficients will still correspond to points in the same subspace. With larger amounts of noise, some of the nonzero coefficients will come from other subspaces. These mistakes can be handled by spectral clustering, which is also robust to noise (see [56]). With respect to GPCA, SSC is more robust to outliers because, as in the case of LSA, SLBF and LLMC, it is very unlikely that a point in a subspace will write it self as a linear combination of a point that is very far from the all the subspaces. Also, the computational complexity of SSC does not grow exponentially with the number of subspaces and their dimensions. Nonetheless, it requires solving N optimization problems in N variables, as per (24), (25) or (26), hence it can be slow. With respect to LSA and LLMC, the great advantage of SSC is that the neighbors of a point are automatically chosen, without having to specify the value of K. Indeed, the number of nonzero coefficients should correspond to the dimension of the subspace. More importantly, the SSC algorithm is provably correct for independent and disjoint subspaces, hence its performance is not affected when the NNs of a point (in the traditional sense) do not come from the same subspace containing that point. Another advantage of SCC over GPCA is that does not require the data to be projected onto a low-dimensional subspace. A possible disadvantage of SSC is that it is provably

correct only in the case of independent or disjoint subspaces. However, the experiments will show that SSC performs well also for non-disjoint subspaces.

Low-Rank Representation (LRR). This algorithm [26] is very similar to SSC, except that it aims to find a low-rank representation instead of a sparse representation. Before explaining the connection further, let us first rewrite the SSC algorithm in matrix form. Specifically, recall that SSC requires solving N optimization problems in N variables, as per (24). As it turns out, these optimization problems can be written as a single optimization problem in $\mathrm{O}(N^2)$ variables as

$$\min_{\{w_{jk}\}} \sum_{j=1}^{N} \sum_{k \neq j} |w_{jk}| \text{ s.t. } \boldsymbol{x}_j = \sum_{k \neq j} w_{jk} \boldsymbol{x}_k \text{ (and } \sum_{k \neq j} w_{jk} = 1). \tag{28}$$

This problem can be rewritten in matrix form as

$$\min_{W} \|W\|_1 \text{ s.t. } X = XW^\top, \ \mathrm{diag}(W) = \mathbf{0} \text{ (and } W\mathbf{1} = \mathbf{1}). \tag{29}$$

Similarly, in the case of data contaminated with noise, the N optimization problems in (25) can be written as

$$\begin{aligned} \min_{W,E} \quad & \|W\|_1 + \mu \|E\|_F^2 \\ \text{s.t.} \quad & X = XW^\top + E, \ \mathrm{diag}(W) = \mathbf{0} \text{ (and } W\mathbf{1} = \mathbf{1}). \end{aligned} \tag{30}$$

The LRR algorithm aims to minimize $\mathrm{rank}(W)$ instead of $\|W\|_1$. Since this rank-minization problem is NP hard, the authors replace the rank of W by its nuclear norm $\|W\|_* = \sum \sigma_i(W)$, where $\sigma_i(W)$ is the ith singular value of W. In the case of noise free data drawn from linear (affine) subspaces, this leads to the following (convex) optimization problem

$$\min_{W} \|W\|_* \quad \text{s.t.} \quad X = XW^\top \text{ (and } W\mathbf{1} = \mathbf{1}). \tag{31}$$

It can be shown that when the data is noise free and drawn from independent linear subspaces, the optimal solution to (31) is given by the matrix Q of the Costeira and Kanade algorithm, as defined in (5). Recall that this matrix is such that $Q_{jk} = 0$ when points j and k are in different subspaces (see (6)), and can be used to build an affinity matrix.

In the case of data contaminated with noise or outliers, the LRR algorithm solves the (convex) optimization problem

$$\min_{W} \|W\|_* + \mu \|E\|_{2,1} \text{ s.t. } X = XW^\top + E \text{ (and } W\mathbf{1} = \mathbf{1}.), \tag{32}$$

where $\|E\|_{2,1} = \sum_{k=1}^{N} \sqrt{\sum_{j=1}^{N} |E_{jk}|^2}$ is the $\ell_{2.1}$ norm of the matrix of errors E. Notice that this problem is analogous to (30), except that the ℓ_1 and the Frobenious norms are replaced by the nuclear and the $\ell_{2,1}$ norms, respectively.

The LRR algorithm proceeds by solving the optimization problem in (32) using an augmented Lagrangian method. The optimal W is used to define an affinity matrix A as in (27). The segmentation of the data is then obtained by applying spectral clustering to the normalized Laplacian.

One of the main attractions of the LRR algorithm is that it results on very interesting connections between the Costeira and Kanade algorithm and the SSC algorithm. A second advantage is that, similarly to SSC, the optimization problem is convex. Perhaps the main drawback of LRR is that the optimization problem involves $\mathrm{O}(N^2)$ variables.

Spectral Curvature Clustering (SCC). The methods discussed so far choose a data point plus d NNs (LSA, SLBF, LLMC) or d "sparse" neighbors (SSC), fit an affine subspace to each of these N groups of $d+1$ points, and build a pairwise affinity by comparing these subspaces. In contrast, multiway clustering techniques such as [57, 58, 27] are based on the observation that a minimum of $d+1$ points are needed to define an affine subspace of dimension d (d for linear subspaces). Therefore, they consider $d+2$ points, build a measure of how likely these points are to belong to the same subspace, and use this measure to construct an affinity between pairs of points.

Specifically, let $X_{d+2} = \{\boldsymbol{x}_{j_\ell}\}_{\ell=1}^{d+2}$ be $d+2$ randomly chosen data points. One possible affinity is the volume spanned by the $(d+1)$-simplex formed by these points, $\mathrm{vol}(X_{d+2})$, which is equal to zero if the points are in the same subspace. However, one issue with this affinity is that it is not invariant to data transformations, e.g., scaling of the $d+2$ points. The SCC algorithm [27] is based on the concept of *polar curvature*, which is also zero when the points are in the same subspace. The multi-way affinity $\mathcal{A}_{j_1,j_2,\ldots,j_{d+2}}$ is defined as

$$\exp\Big(-\frac{1}{2\sigma^2}\mathrm{diam}^2(X_{d+2}) \sum_{\ell=1}^{d+2} \frac{(d+1)!^2 \mathrm{vol}^2(X_{d+2})}{\prod_{\substack{1 \leq m \leq d+2 \\ m \neq \ell}} \|\boldsymbol{x}_{j_m} - \boldsymbol{x}_{j_\ell}\|^2}\Big) \tag{33}$$

if $j_1, j_2, \ldots, j_{d+2}$ are distinct and zero otherwise. A pairwise affinity matrix is then defined as

$$A_{jk} = \sum_{j_2,\ldots,j_{d+1}} \mathcal{A}_{j,j_2,\ldots,j_{d+2}} \mathcal{A}_{k,j_2,\ldots,j_{d+2}}. \tag{34}$$

This requires computing $\mathrm{O}(N^{d+2})$ entries of $\mathcal{A}$ and summing over $\mathrm{O}(N^{d+1})$ elements of $\mathcal{A}$. Therefore, the computational complexity of SCC grows exponentially with the dimension of the subspaces. A practical implementation of SCC uses a fixed number c of $(d+1)$-tuples ($c \ll N^{d+1}$) for each point to build the affinity A. A choice of $c \approx c_0 n^{d+2}$ is suggested in [27], which is much smaller, but still exponential in d. In practice, the method appears to be not too sensitive to the choice of c but more importantly to how the $d+1$ points are chosen. [27] argues that a uniform sampling strategy does not perform well, because many samples could contain subspaces of different dimensions. To avoid this, two stages of sampling are performed. The first stage is used to obtain an initial clustering of the data. In the second stage, the initial clusters are used to guide the sampling and thus obtain a better affinity. Given A, the segmentation is obtained by applying spectral clustering to the normalized Laplacian. One difference of SCC with

respect to the previous methods is that SCC uses a procedure for initializing K-means based on maximizing the variance among all possible combinations of K rows of V.

One advantage of SCC (and also of SSC) over LSA, SLBF and LLMC is that it incorporates many points to define the affinities, while LSA, SLBF and LLMC restrict themselves to K-NNs. This ultimately results in better affinities, especially for subspaces that are not independent. One advantage of SCC over factorization-based methods and GPCA is that it can handle noisy data drawn from both linear and affine subspaces. Another advantage of SCC over GPCA is that does not require the data to be projected onto a low-dimensional subspace. Also, when the data are sampled from a mixture of distributions concentrated around multiple affine subspaces, SCC performs well with overwhelming probability, as shown in [59]. Finally, SCC can be extended to nonlinear manifolds by using kernel methods [60]. However, the main drawbacks of SCC are that it requires sampling of the affinities to reduce the computational complexity and that it requires the subspaces to be of known and equal dimension d. In practice, the algorithm can still be applied to subspaces of different dimensions by choosing $d = d_{\max}$, but the effect of this choice on the definition of spectral curvature remains unknown.

3. APPLICATIONS IN COMPUTER VISION

3.1. Motion segmentation from feature point trajectories

Motion segmentation refers to the problem of separating a video sequence into multiple spatiotemporal regions corresponding to different rigid-body motions. Most existing motion segmentation algorithms proceed by first extracting a set of point trajectories from the video using standard tracking methods. As a consequence, the motion segmentation problem is reduced to clustering these point trajectories according to the different rigid-body motions in the scene.

The mathematical models needed to describe the motion of the point trajectories vary depending on the type of camera projection model. Under the affine model, all the trajectories associated with a single rigid motion live in a 3-dimensional affine subspace. To see this, let $\{x_{fj} \in \mathbb{R}^2\}_{j=1,\ldots,N}^{f=1,\ldots,F}$ denote the 2-D projections of N 3-D points $\{X_j \in \mathbb{R}^3\}_{j=1}^{N}$ on a rigidly moving object onto F frames of a moving camera. The relationship between the tracked feature points and their corresponding 3-D coordinates is

$$x_{fj} = A_f \begin{bmatrix} X_j \\ 1 \end{bmatrix}, \tag{35}$$

where $A_f \in \mathbb{R}^{2\times 4}$ is the affine motion matrix at frame f. If we form a matrix containing all the F tracked feature points corresponding to a point on the object in a column, we get

$$\begin{bmatrix} x_{11} \cdots x_{1N} \\ \vdots \quad \vdots \\ x_{F1} \cdots x_{FN} \end{bmatrix}_{2F\times N} = \begin{bmatrix} A_1 \\ \vdots \\ A_F \end{bmatrix}_{2F\times 4} \begin{bmatrix} X_1 \cdots X_N \\ 1 \cdots 1 \end{bmatrix}_{4\times N} \tag{36}$$

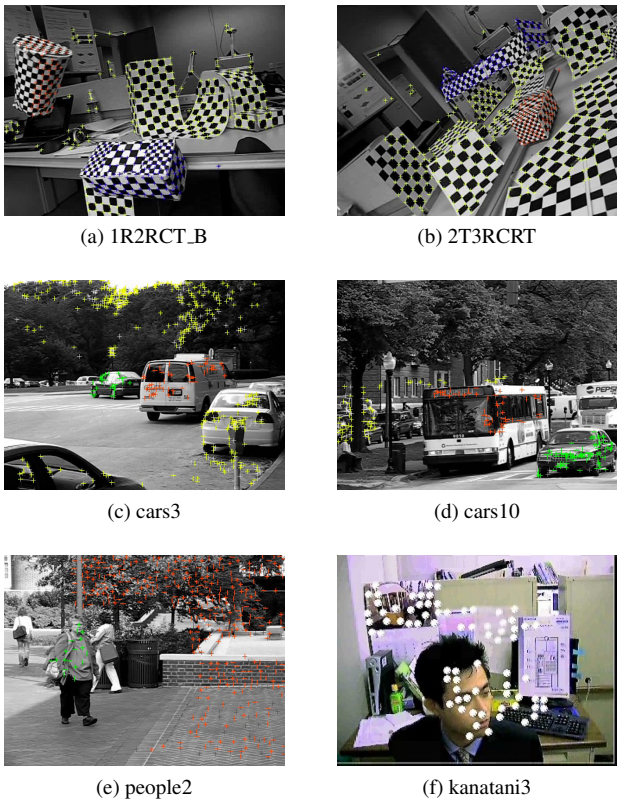
We can briefly write this as $W = MS^\top$, where $M \in \mathbb{R}^{2F\times 4}$ is called the motion matrix and $S \in \mathbb{R}^{N\times 4}$ is called the structure matrix. Since $\text{rank}(M) \leq 4$ and $\text{rank}(S) \leq 4$ we get

$$\text{rank}(W) = \text{rank}(MS^\top) \leq \min(\text{rank}(M), \text{rank}(S)) \leq 4. \tag{37}$$

Moreover, since the last row of $S^\top$ is 1, the feature point trajectories of a single rigid-body motion lie in an affine subspace of $\mathbb{R}^{2F}$ of dimension at most three.

Assume now that we are given N trajectories of n rigidly moving objects. Then, these trajectories will lie in a union of n affine subspaces in $\mathbb{R}^{2F}$. The 3-D motion segmentation problem is the task of clustering these N trajectories into n different groups such that the trajectories in the same group represent a single rigid-body motion. Therefore, the motion segmentation problem reduces to clustering a collection of point trajectories according to multiple affine subspaces.

In what follows, we evaluate a number of subspace clustering algorithms on the Hopkins155 motion segmentation



(a) 1R2RCT_B (b) 2T3RCRT (c) cars3 (d) cars10 (e) people2 (f) kanatani3

Fig. 2: Sample images from some sequences in the database with tracked points superimposed.

database, which is available online at `http://www.vision.jhu.edu/data/hopkins155` [61]. The database consists of 155 sequences of two and three motions which can be divided into three main categories: checkerboard, traffic, and articulated sequences. The checkerboard sequences contain multiple objects moving independently and arbitrarily in 3D space, hence the motion trajectories lie in independent affine subspaces of dimension three. The traffic sequences contain cars moving independently on the ground plane, hence the motion trajectories lie in independent affine subspaces of dimension two. The articulated sequences contain motions of people, cranes, etc., where object parts do not move independently, and so the motion subspaces are dependent. For each sequence, the trajectories are extracted automatically with a tracker and outliers are manually removed. Therefore, the trajectories are corrupted by noise, but do not have missing entries or outliers. Figure 2 shows sample images from videos in the database with the feature points superimposed.

In order to make our results comparable to those in the existing literature, for each method we apply the same preprocessing steps described in their respective papers. Specifically, we project the trajectories onto a subspace of dimension $r \leq 2F$ using either PCA (GPCA, RANSAC, LLMC, LSA, ALC, SCC) or a random projection matrix (SSC) whose entries are drawn from a Bernoulli (SSC-B) or Normal (SSC-N) distribution. Historically, there have been two choices for the dimension of the projection: $r = 5$ and $r = 4n$. These choices are motivated by algebraic methods, which model 3-D affine subspaces as 4-D linear subspaces. Since $d_{\max} = 4$, GPCA chooses $r = d_{\max} + 1 = 5$, while factorization methods use the fact that for independent subspaces $r = \text{rank}(X) = 4n$. In our experiments, we use $r = 5$ for GPCA and RANSAC and $r = 4n$ for GPCA, LLMC, LSA, SCC and SSC. For ALC, r is chosen automatically for each sequence as the minimum r such that $r \geq 8 \log(2F/r)$. We will refer to this one as the *sparsity preserving* (sp) projection. We refer the reader to [62] for more recent work that determines the dimension of the projection automatically. Also, for the algorithms that make use of K-means, either a single restart is used when initialized by another algorithm (LLMC, SCC), or 10 restarts are used when initialized at random (GPCA, LLMC, LSA). SSC uses 20 restarts.

For each algorithm and each sequence, we record the classification error defined as

$$\text{classification error} = \frac{\text{\# of misclassified points}}{\text{total \# of points}} \, \%. \quad (38)$$

Table 1 reports the average and median misclassification errors and Figure 3 shows, the percentage of sequences for which the classification error is below a given percentage of misclassification. More detailed statistics with the classification errors and computation times of each algorithm on each of the 155 sequences can be found at `http://www.vision.jhu.edu/data/hopkins155/`.

By looking at the results, we can draw the following conclusions about the performance of the algorithms tested.

GPCA. To avoid using multiple polynomials, we use an implementation of GPCA based on hyperplanes in which the data is interpreted as a subspace of dimension $r - 1$ in $\mathbb{R}^r$, where $r = 5$ or $r = 4n$. For two motions, GPCA achieves a classification error of 4.59% for $r = 5$ and 4.10% for $r = 4n$ Notice that GPCA is among the most accurate methods for the traffic and articulated sequences, which are sequences with dependent motion subspaces. However, GPCA has higher errors on the checkerboard sequences, which constitute the majority of the database. This result is expected, because GPCA is best designed for dependent subspaces. Notice also that increasing r from 5 to $4n$ improves the results for checkerboard sequences, but not for the traffic and articulated sequences. This is also expected, because the rank of the data matrix should be high for sequences with full-dimensional and independent motions (checkerboard), and low for sequences with degenerate (traffic) and dependent (articulated) motions. This suggest that using model selection to determine a different value of r for each sequence should improve the results. For three motions, the results are completely different with a segmentation error of 29-37%. This is expected, because the number of coefficients fitted by GPCA grows exponentially with the number of motions, while the number of feature points remains of the same order. Furthermore, GPCA uses a least-squares method for fitting the polynomial, which neglects nonlinear constraints among the coefficients. The number of nonlinear constraints neglected also increases with the number of subspaces.

RANSAC. The results for this purely statistical algorithm are similar to what we found for GPCA. In the case of two motions the results are a bit worse than those of GPCA. In the case of three motions, the results are better than those of GPCA, but still quite far from those of the best performing algorithms. This is expected, because as the number of motions increases, the probability of drawing a set of points from the same group reduces significantly. Another drawback of RANSAC is that its performance varies between two runs on the same data. Our experiments report the average performance over 1,000 trials for each sequence.

LSA. When the dimension for the projection is chosen as $r = 5$, this algorithm performs worse than GPCA. This is because points in different subspaces are closer to each other when $r = 5$, and so a point from a different subspace is more likely to be chosen as a nearest neighbor. GPCA, on the other hand, is not affected by points near the intersection of the subspaces. The situation is completely different when $r = 4n$. In this case, LSA clearly outperforms GPCA and RANSAC, achieving an error of 3.45% for two groups and 9.73% for three groups. These errors could be further reduced by using model selection to determine the dimension of each subspace. Another important thing to observe is that LSA performs bet-

Table 1: Classification errors of several subspace clustering algorithms on the Hopkins 155 motion segmentation database. All algorithms use two parameters (d, r), where d is the dimension of the subspaces and r is the dimension of the projection. Affine subspace clustering algorithms treat subspaces as 3-dimensional affine subspaces, i.e., $d = 3$, while linear subspace clustering algorithms treat subspaces as 4-dimensional linear subspaces, i.e., $d = 4$. The dimensions of the projections are $r = 5$, $r = 4n$, where n is the number of motions, and $r = 2F$, where F is the number of frames. ALC uses a sparsity preserving (sp) dimension for the projection. All algorithms use PCA to perform the projection, except for SSC which uses a random projection with entries drawn from a Bernoulli (SSC-B) or Normal (SSC-N) distribution. The results for GPCA correspond to the spectral clustering-based GPCA algorithm. LLMC-G denotes LLMC initialized by the algebraic GPCA algorithm.

		Two motions								Three motions								All (155)	
		Check. (78)		Traffic (31)		Articul. (11)		All (120)		Check. (26)		Traffic (7)		Articul. (2)		All (35)			
		Mean	Median	Mean	Median	Mean	Median	Mean	Median	Mean	Median	Mean	Median	Mean	Median	Mean	Median	Mean	Median
GPCA	(4,5)	6.09	1.03	1.41	0.00	2.88	0.00	4.59	0.38	31.95	32.93	19.83	19.55	16.85	16.85	28.66	28.26	10.34	2.54
GPCA (4n-1,4n)		4.78	0.51	1.63	0.00	6.18	3.20	4.10	0.44	36.99	36.26	39.68	40.92	29.62	29.62	37.11	37.18	11.55	1.36
RANSAC	(4,5)	6.52	1.75	2.55	0.21	7.25	2.64	5.56	1.18	25.78	26.00	12.83	11.45	21.38	21.38	22.94	22.03	9.76	3.21
LSA	(4,5)	8.84	3.43	2.15	1.00	4.66	1.28	6.73	1.99	30.37	31.98	27.02	34.01	23.11	23.11	29.28	31.63	11.82	4.00
LSA	(4,4n)	2.57	0.27	5.43	1.48	4.10	1.22	3.45	0.59	5.80	1.77	25.07	23.79	7.25	7.25	9.73	2.33	4.94	0.90
LLMC	(4,5)	4.85	0.00	1.96	0.00	6.16	1.37	4.22	0.00	9.06	7.09	6.45	0.00	5.26	5.26	8.33	3.19	5.15	0.00
LLMC	(4,4n)	3.96	0.23	3.53	0.33	6.48	1.30	4.08	0.24	8.48	5.80	6.04	4.09	9.38	9.38	8.04	4.93	4.97	0.87
LLMC-G	(4,5)	4.34	0.00	2.13	0.00	6.16	1.37	3.95	0.00	8.87	7.09	5.62	0.00	5.26	5.26	8.02	3.19	4.87	0.00
LLMC-G	(4,4n)	2.83	0.00	3.61	0.00	5.94	1.30	3.32	0.00	8.20	5.26	6.04	4.60	8.32	8.32	7.78	4.93	4.37	0.53
MSL		4.46	0.00	2.23	0.00	7.23	0.00	4.14	0.00	10.38	4.61	1.80	0.00	2.71	2.71	8.23	1.76	5.03	0.00
ALC	(4,5)	2.56	0.00	2.83	0.30	6.90	0.89	3.03	0.00	6.78	0.92	4.01	1.35	7.25	7.25	6.26	1.02	3.76	0.26
ALC	(4,sp)	1.49	0.27	1.75	1.51	10.70	0.95	2.40	0.43	5.00	0.66	8.86	0.51	21.08	21.08	6.69	0.67	3.37	0.49
SCC	(3, 4)	2.99	0.39	1.20	0.32	7.71	3.67	2.96	0.42	7.72	3.21	0.52	0.28	8.90	8.90	6.34	2.36	3.72	
SCC	(3, 4n)	1.76	0.01	0.46	0.16	4.06	1.69	1.63	0.06	6.00	2.22	1.78	0.42	5.65	5.65	5.14	1.67	2.42	
SCC	(3, 2F)	1.77	0.00	0.63	0.14	4.02	2.13	1.68	0.07	6.23	1.70	1.11	1.40	5.41	5.41	5.16	1.58	2.47	
SCC	(4, 5)	2.31	0.25	0.71	0.26	5.05	1.08	2.15	0.27	5.56	2.03	1.01	0.47	8.97	8.97	4.85	2.01	2.76	
SCC	(4, 4n)	1.30	0.04	1.07	0.44	3.68	0.67	1.46	0.16	5.68	2.96	2.35	2.07	10.94	10.94	5.31	2.40	2.33	
SCC	(4, 2F)	1.31	0.06	1.02	0.26	3.21	0.76	1.41	0.10	6.31	1.97	3.31	3.31	9.58	9.58	5.90	1.99	2.42	
SLBF	(3, 2F)	1.59	0.00	0.20	0.00	0.80	0.00	1.16	0.00	4.57	0.94	**0.38**	0.00	2.66	2.66	3.63	0.64	1.66	
SSC-B	(4,4n)	**0.83**	0.00	0.23	0.00	1.63	0.00	**0.75**	0.00	4.49	0.54	0.61	0.00	1.60	1.60	3.55	0.25	1.45	0.00
SSC-N	(4,4n)	1.12	0.00	**0.02**	0.00	**0.62**	0.00	0.82	0.00	**2.97**	0.27	0.58	0.00	**1.42**	0.00	**2.45**	0.20	**1.24**	0.00

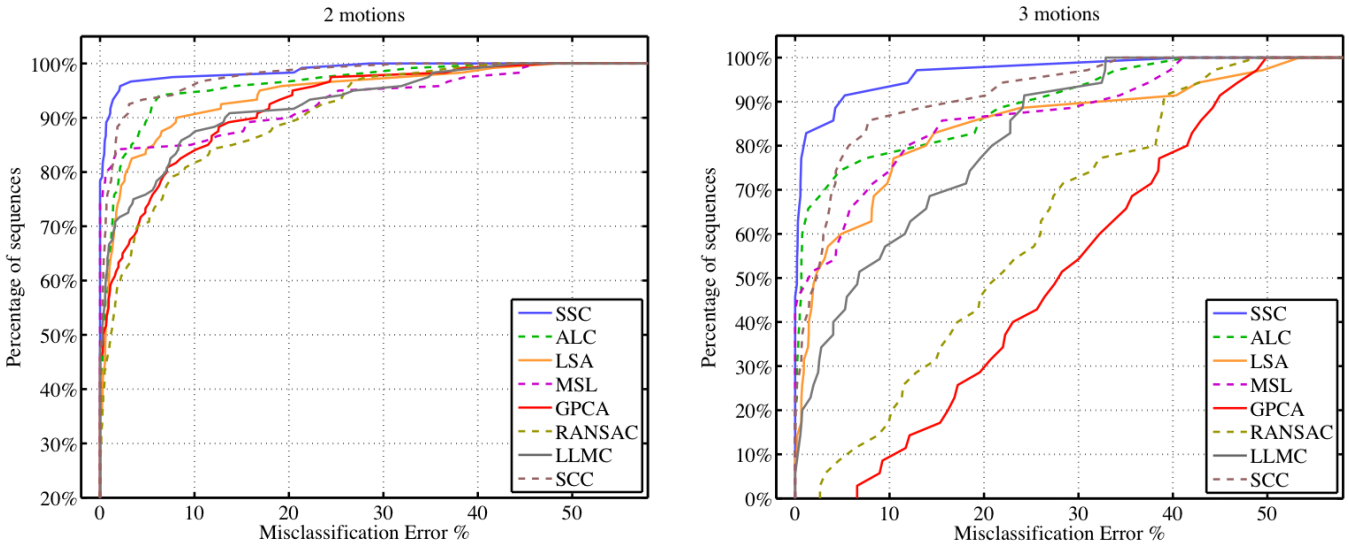


Fig. 3: Percentage of sequences for which the classification error is less than or equal to a given percentage of misclassification. The algorithms tested are GPCA (4,5), RANSAC (4,5), LSA (4,4n), LLMC (4,4n), MSL, ALC (4,sp), SCC (4,4n), SSC-N (4,4n).

ter on the checkerboard sequences, but has larger errors than GPCA on the traffic and articulated sequences. This confirms that LSA has difficulties with dependent subspaces.

LLMC. The results of this algorithm also represent a clear improvement over GPCA and RANSAC, especially for three motions. The only cases where GPCA outperforms LLMC are for traffic and articulated sequences. This is expected, because LLMC is not designed to handle dependent subspaces. Unlike LSA, LLMC is not significantly affected by the choice of r, with a classification error of 5.15% for $r = 5$ and 4.97% for $r = 4n$. Notice also that the performance of LLMC improves when initialized with GPCA to 4.87% for $r = 5$ and 4.37% for $r = 4n$. However, there are a few sequences for which LLMC performs worse than GPCA even when LLMC is initialized by GPCA. This happens for sequences with dependent motions, which are not well handled by LLMC.

MSL. By looking at the average classification error, we can see that MSL, LSA and LLMC have a similar accuracy. Furthermore, their segmentation results remain consistent when going from two to three motions. However, sometimes the MSL method gets stuck in a local minimum. This is reflected by high classification errors for some sequences, as it can be seen by the long tails in Figure 3.

ALC. This algorithm represents a significant increase in performance with respect to all previous algorithms, especially for the checkerboard sequences, which constitute the majority of the database. However, ALC does not perform very well on the articulated sequences. This is because ALC typically needs the samples from a group to cover the subspace with sufficient density, while many of the articulated scenes have very few feature point trajectories. With regard to the projection dimension, the results indicate that, overall, ALC performs better with an automatic choice of the projection, rather than with a fixed choice of $r = 5$. One drawback of ALC is that it needs to be run about 100 times for different choices of the distortion parameter ε in order to obtain the right number of motions and the best segmentation results.

SCC. This algorithm performs even better than ALC, in almost all motion categories. The only exception is for the articulated sequences with three motions. This is because these sequences contain few trajectories for the sampling strategy to operate correctly. Another advantage of SCC with respect to ALC is that it is not very sensitive to the choice of the parameter c (number of sampled subsets), while ALC needs to be run for several choices of the distortion parameter ε. Notice also that the performance of SCC is not significantly affected by the dimension of the projection $r = 5$, $r = 4n$ or $r = 2F$.

SSC. This algorithm performs extremely well, not only for checkerboard sequences, which have independent and fully-dimensional motion subspaces, but also for traffic and articulated sequences, which are the bottleneck of almost all existing methods, because they contain degenerate and dependent motion subspaces. This is surprising, because the algorithm is provably correct only for independent or disjoint subspaces. Overall, the performance of SSC is not very sensitive to the choice of the projection (Bernoulli versus Normal), though SSC-N gives slightly better results. We have observed also that SSC is not sensitive to the dimension of the projection ($r = 5$ vs. $r = 4n$ vs. $r = 2F$) or the parameter μ.

SLBF. This algorithm performs extremely well for all motion sequences. Its performance is essentially on par with that of SSC. We refer the reader to [22] for additional experiments.

3.2. Face clustering under varying illumination

Given a collection of unlabeled images $\{I_j \in \mathbb{R}^D\}_{j=1}^N$ of n different faces taken under varying illumination, the face clustering problem consists of clustering the images corresponding to the face of the same person. For a Lambertian object, it has been shown that the set of all images taken under all lighting conditions forms a cone in the image space, which can be well approximated by a low-dimensional subspace [3]. Therefore, the face clustering problem reduces to clustering a collection of images according to multiple subspaces.

In what follows, we report experiments from [22], which evaluate the GPCA, ALC, SCC, SLBF and SSC algorithms on the face clustering problem. The experiments are performed on the Yale Faces B database, which is available at `http://cvc.yale.edu/projects/yalefacesB/yalefacesB.html`. This database consists of $10 \times 9 \times 64$ images of 10 faces taken under 9 different poses and 64 different illumination conditions. For computational efficiency, the images are downsampled to 120×160 pixels. Nine subsets of $n = 2, \ldots, 10$ are considered containing the following indices: [5, 8], [1, 5, 8], [1, 5, 8, 10], [1, 4, 5, 8, 10], [1, 2, 4, 5, 8, 10], [1, 2, 4, 5, 7, 8, 10], [1, 2, 4, 5, 7, 8, 9, 10], [1, 2, 3, 4, 5, 7, 8, 9, 10] and [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]. Since in practice the number of pixels D is still large compared with the dimension of the subspaces, PCA is used to project the images onto a subspace of dimension $r = 5$ for GPCA and $r = 20$ for ALC, SCC, SLBF and SSC. In all cases, the dimension of the subspaces is set to $d = 2$.

Table 2 shows the average percentage of misclassified faces. As expected, GPCA does not perform well, since it is hard to distinguish faces from only 5 dimensions. Nonetheless, GPCA cannot handle 20 dimensions, especially as the number of groups increases. All other algorithms perform extremely well in this dataset, especially SLBF and ALC.

Table 2: Mean percentage of misclassification on clustering Yale Faces B data set.

n	2	3	4	5	6	7	8	9	10
GPCA	0.0	49.5	0.0	26.6	9.9	25.2	28.5	30.6	19.8
ALC	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
SCC	0.0	0.0	0.0	1.1	2.7	2.1	2.2	5.7	6.6
SLBF	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.2	0.9
SSC	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2.4	4.6

4. CONCLUSIONS AND FUTURE DIRECTIONS

Over the past few decades, significant progress has been made in clustering high-dimensional datasets distributed around a collection of linear and affine subspaces. This article presented a review of such progress, which included a number of existing subspace clustering algorithms together with an experimental evaluation on the motion segmentation problem in computer vision. While earlier algorithms were designed under the assumptions of perfect data and perfect knowledge of the number of subspaces and their dimensions, throughout the years algorithms started to handle noise, outliers, data with missing entries, unknown number of subspaces and unknown dimensions. In the case of noiseless data drawn from linear subspaces, the theoretical correctness of existing algorithms is well studied and some algorithms such as GPCA are able to handle an unknown number of subspaces of unknown dimensions in an arbitrary configuration. However, while GPCA is applicable to affine subspaces, a theoretical analysis of GPCA for affine subspaces in the noiseless case is still due. In the case of noisy data, the theoretical correctness of existing algorithms is largely untouched. To the best of our knowledge, the first works in this direction are [45, 59]. By and large, most existing algorithms assume that the number of subspaces and their dimensions are known. While some algorithms can provide estimates for these quantities, their estimates come with no theoretical guarantees. In our view, the development of theoretically sound algorithms for finding the number of subspaces and their dimension in the presence of noise and outliers is a very important open challenge. On the other hand, it is important to mention that most existing algorithms operate in a batch fashion. In real-time applications, it is important to cluster the data as it is being collected, which motivates the development of online subspace clustering algorithms. The works of [63] and [15] are two examples in this direction. Finally, in our view the grand challenge for the next decade will be to develop clustering algorithms for data drawn from multiple nonlinear manifolds. The works of [64, 65, 66, 67] have already considered the problem of clustering quadratic, bilinear and trilinear surfaces using algebraic algorithms designed for noise free data. The development of methods that are applicable to more general manifolds with corrupted data is still at its infancy.

5. AUTHOR

René Vidal (*rvidal@jhu.edu*) received his B.S. degree in Electrical Engineering (highest honors) from the Pontificia Universidad Católica de Chile in 1997 and his M.S. and Ph.D. degrees in Electrical Engineering and Computer Sciences from the University of California at Berkeley in 2000 and 2003, respectively. He was a research fellow at the National ICT Australia in 2003 and joined The Johns Hopkins University in 2004 as a faculty member in the Department of Biomedical Engineering and the Center for Imaging Science. He was co-editor of the book "Dynamical Vision" and has co-authored more than 100 articles in biomedical image analysis, computer vision, machine learning, hybrid systems, and robotics. He is recipient of the 2009 ONR Young Investigator Award, the 2009 Sloan Research Fellowship, the 2005 NFS CAREER Award and the 2004 Best Paper Award Honorable Mention at the European Conference on Computer Vision. He also received the 2004 Sakrison Memorial Prize for "completing an exceptionally documented piece of research", the 2003 Eli Jury award for "outstanding achievement in the area of Systems, Communications, Control, or Signal Processing", the 2002 Student Continuation Award from NASA Ames, the 1998 Marcos Orrego Puelma Award from the Institute of Engineers of Chile, and the 1997 Award of the School of Engineering of the Pontificia Universidad Católica de Chile to the best graduating student of the school. He is a member of the IEEE and the ACM.

6. REFERENCES

[1] A. Yang, J. Wright, Y. Ma, and S. Sastry, "Unsupervised segmentation of natural images via lossy data compression," *Computer Vision and Image Understanding*, vol. 110, no. 2, pp. 212–225, 2008.

[2] R. Vidal, R. Tron, and R. Hartley, "Multiframe motion segmentation with missing data using PowerFactorization and GPCA," *International Journal of Computer Vision*, vol. 79, no. 1, pp. 85–105, 2008.

[3] J. Ho, M. H. Yang, J. Lim, K.C. Lee, and D. Kriegman, "Clustering appearances of objects under varying illumination conditions.," in *IEEE Conf. on Computer Vision and Pattern Recognition*, 2003.

[4] Wei Hong, John Wright, Kun Huang, and Yi Ma, "Multi-scale hybrid linear models for lossy image representation," *IEEE Trans. on Image Processing*, vol. 15, no. 12, pp. 3655–3671, 2006.

[5] R. Vidal, S. Soatto, Y. Ma, and S. Sastry, "An algebraic geometric approach to the identification of a class of linear hybrid systems," in *Conference on Decision and Control*, 2003, pp. 167–172.

[6] L. Parsons, E. Haque, and H. Liu, "Subspace clustering for high dimensional data: a review," *ACM SIGKDD Explorations Newsletter*, 2004.

[7] T.E. Boult and L.G. Brown, "Factorization-based segmentation of motions," in *IEEE Workshop on Motion Understanding*, 1991, pp. 179–186.

[8] J. Costeira and T. Kanade, "A multibody factorization method for independently moving objects.," *Int. Journal of Computer Vision*, vol. 29, no. 3, 1998.

[9] C. W. Gear, "Multibody grouping from motion images," *Int. Journal of Computer Vision*, vol. 29, no. 2, pp. 133–150, 1998.

[10] R. Vidal, Y. Ma, and S. Sastry, "Generalized Principal Component Analysis (GPCA)," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 12, pp. 1–15, 2005.

[11] P. S. Bradley and O. L. Mangasarian, "k-plane clustering," *J. of Global Optimization*, vol. 16, no. 1, pp. 23–32, 2000.

[12] P. Tseng, "Nearest q-flat to m points," *Journal of Optimization Theory and Applications*, vol. 105, no. 1, pp. 249–252, 2000.

[13] P. Agarwal and N. Mustafa, "k-means projective clustering," in *ACM SIGMOD-SIGACT-SIGART Symposium on Principles of database systems*, 2004.

[14] L. Lu and R. Vidal, "Combined central and subspace clustering on computer vision applications," in *International Conference on Machine Learning*, 2006, pp. 593–600.

[15] T. Zhang, A. Szlam, and G. Lerman, "Median k-flats for hybrid linear modeling with many outliers," in *Workshop on Subspace Methods*, 2009.

[16] M. Tipping and C. Bishop, "Mixtures of probabilistic principal component analyzers," *Neural Computation*, vol. 11, no. 2, pp. 443–482, 1999.

[17] Y. Sugaya and K. Kanatani, "Geometric structure of degeneracy for multi-body motion segmentation," in *Workshop on Statistical Methods in Video Processing*, 2004.

[18] Y. Ma, H. Derksen, W. Hong, and J. Wright, "Segmentation of multivariate mixed data via lossy coding and compression," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 9, pp. 1546–1562, 2007.

[19] S. Rao, R. Tron, Y. Ma, and R. Vidal, "Motion segmentation via robust subspace separation in the presence of outlying, incomplete, or corrupted trajectories," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2008.

[20] A. Y. Yang, S. Rao, and Y. Ma, "Robust statistical estimation and segmentation of multiple subspaces," in *Workshop on 25 years of RANSAC*, 2006.

[21] J. Yan and M. Pollefeys, "A general framework for motion segmentation: Independent, articulated, rigid, non-rigid, degenerate and non-degenerate," in *European Conf. on Computer Vision*, 2006, pp. 94–106.

[22] T. Zhang, A. Szlam, Y. Wang, and G. Lerman, "Hybrid linear modeling via local best-fit flats," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2010, pp. 1927–1934.

[23] A. Goh and R. Vidal, "Segmenting motions of different types by unsupervised manifold clustering," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2007.

[24] E. Elhamifar and R. Vidal, "Sparse subspace clustering," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

[25] E. Elhamifar and R. Vidal, "Clustering disjoint subspaces via sparse representation," in *IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2010.

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

[27] G. Chen and G. Lerman, "Spectral curvature clustering (SCC)," *International Journal of Computer Vision*, vol. 81, no. 3, pp. 317–330, 2009.

[28] I. Jolliffe, *Principal Component Analysis*, Springer-Verlag, New York, 1986.

[29] E. Beltrami, "Sulle funzioni bilineari," *Giornale di Mathematiche di Battaglini*, vol. 11, pp. 98–106, 1873.

[30] M.C. Jordan, "Mémoire sur les formes bilinéaires," *Journal de Mathématiques Pures et Appliqués*, vol. 19, pp. 35–54, 1874.

[31] H. Stark and J.W. Woods, *Probability and Random Processes with Applications to Signal Processing*, Prentice Hall, 3rd edition, 2001.

[32] C. Eckart and G. Young, "The approximation of one matrix by another of lower rank," *Psychometrika*, vol. 1, pp. 211–218, 1936.

[33] K. Kanatani, "Motion segmentation by subspace separation and model selection," in *IEEE Int. Conf. on Computer Vision*, 2001, vol. 2, pp. 586–591.

[34] N. Ichimura, "Motion segmentation based on factorization method and discriminant criterion," in *IEEE Int. Conf. on Computer Vision*, 1999, pp. 600–605.

[35] Y. Wu, Z. Zhang, T.S. Huang, and J.Y. Lin, "Multibody grouping via orthogonal subspace decomposition," in *IEEE Conf. on Computer Vision and Pattern Recognition*, 2001, vol. 2, pp. 252–257.

[36] K. Kanatani and C. Matsunaga, "Estimating the number of independent motions for multibody motion segmentation," in *European Conf. on Computer Vision*, 2002, pp. 25–31.

[37] K. Kanatani, "Geometric information criterion for model selection," *International Journal of Computer Vision*, pp. 171–189, 1998.

[38] L. Zelnik-Manor and M. Irani, "On single-sequence and multi-sequence factorizations," *Int. Journal of Computer Vision*, vol. 67, no. 3, pp. 313–326, 2006.

[39] Y. Ma, A. Yang, H. Derksen, and R. Fossum, "Estimation of subspace arrangements with applications in modeling and segmenting mixed data," *SIAM Review*, 2008.

[40] H. Derksen, "Hilbert series of subspace arrangements," *Journal of Pure and Applied Algebra*, vol. 209, no. 1, pp. 91–98, 2007.

[41] N. Ozay, M Sznaier, C. Lagoa, and O. Camps, "Gpca with denoising: A moments-based convex approach," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2010.

[42] A. Yang, S. Rao, A. Wagner, Y. Ma, and R. Fossum, "Hilbert functions and applications to the estimation of subspace arrangements," in *IEEE International Conference on Computer Vision*, 2005.

[43] K. Huang, Y. Ma, and R. Vidal, "Minimum effective dimension for mixtures of subspaces: A robust GPCA algorithm and its applications," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2004, vol. II, pp. 631–638.

[44] R. Duda, P. Hart, and D. Stork, *Pattern Classification*, Wiley, New York, 2nd edition, 2000.

[45] A. Aldroubi and K. Zaringhalam, "Nonlinear least squares in $\mathbb{R}^N$," *Acta Applicandae Mathematicae*, vol. 107, no. 1-3, pp. 325–337, 2009.

[46] A. Aldroubi, C. Cabrelli, and U. Molter, "Optimal nonlinear models for sparsity and sampling," *Journal of Fourier Analysis and Applications*, vol. 14, no. 5-6, pp. 793–812, 2008.

[47] M. Tipping and C. Bishop, "Probabilistic principal component analysis," *Journal of the Royal Statistical Society*, vol. 61, no. 3, pp. 611–622, 1999.

[48] A. Dempster, N. Laird, and D. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *Journal of the Royal Statistical Society B*, vol. 39, pp. 1–38, 1977.

[49] C. Archambeau, N. Delannay, and M. Verleysen, "Mixtures of robust probabilistic principal component analyzers," *Neurocomputing*, vol. 71, no. 7–9, pp. 1274–1282, 2008.

[50] A. Gruber and Y. Weiss, "Multibody factorization with uncertainty and missing data using the EM algorithm," in *IEEE Conf. on Computer Vision and Pattern Recognition*, 2004, vol. I, pp. 707–714.

[51] J. Paisley and L. Carin, "Nonparametric factor analysis with beta process priors,," in *International Conference on Machine Learning*, 2009.

[52] A. Leonardis, H. Bischof, and J. Maver, "Multiple eigenspaces," *Pattern Recognition*, vol. 35, no. 11, pp. 2613–2627, 2002.

[53] Z. Fan, J. Zhou, and Y. Wu, "Multibody grouping by inference of multiple subspaces from high-dimensional data using oriented-frames," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 28, no. 1, pp. 91–105, 2006.

[54] M. A. Fischler and R. C. Bolles, "RANSAC random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 26, pp. 381–395, 1981.

[55] J. Yan and M. Pollefeys, "Articulated motion segmentation using RANSAC with priors," in *Workshop on Dynamical Vision*, 2005.

[56] U. von Luxburg, "A tutorial on spectral clustering," *Statistics and Computing*, vol. 17, 2007.

[57] S. Agarwal, J. Lim, L. Zelnik-Manor, P. Perona, D. Kriegman, and S. Belongie, "Beyond pairwise clustering," in *IEEE Conference on Computer Vision and Pattern Recognition*, June 2005, vol. 2, pp. 838–845.

[58] V. Govindu, "A tensor decomposition for geometric grouping and segmentation," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2005, pp. 1150–1157.

[59] G. Chen and G. Lerman, "Foundations of a multi-way spectral clustering framework for hybrid linear modeling," *Foundations of Computational Mathematics*, vol. 9, no. 5, 2009.

[60] G. Chen, S. Atev, and G. Lerman, "Kernel spectral curvature clustering (KSCC)," in *Workshop on Dynamical Vision*, 2009.

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

[62] F. Lauer and C. Schnörr, "Spectral clustering of linear subspaces for motion segmentation," in *IEEE International Conference on Computer Vision*, 2009.

[63] R. Vidal, "Online clustering of moving hyperplanes," in *Neural Information Processing Systems, NIPS*, 2006.

[64] R. Vidal, Y. Ma, S. Soatto, and S. Sastry, "Two-view multibody structure from motion," *International Journal of Computer Vision*, vol. 68, no. 1, pp. 7–25, 2006.

[65] R. Vidal and Y. Ma, "A unified algebraic approach to 2-D and 3-D motion segmentation," *Journal of Mathematical Imaging and Vision*, vol. 25, no. 3, pp. 403–421, 2006.

[66] R. Vidal and R. Hartley, "Three-view multibody structure from motion," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 2, pp. 214–227, 2008.

[67] S. Rao, A. Yang, S. Sastry, and Y. Ma, "Robust algebraic segmentation of mixed rigid-body and planar motions from two views," *International Journal of Computer Vision*, vol. 88, no. 3, pp. 425–446, 2010.