René Vidal

Subspace Clustering

Applications in motion segmentation and face clustering

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 are 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, and bioinformatics.

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]).

Digital Object Identifier 10.1109/MSP.2010.939739
Date of publication: 17 February 2011

1053-5888/11/$26.00©2011IEEE

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]–[10], iterative methods [11]–[15], statistical methods [16]–[20], and spectral clustering-based methods [7], [21]–[27]. We review these methods, discuss their advantages and disadvantages, and evaluate their performance on the motion segmentation and face-clustering problems.

A NUMBER OF APPROACHES TO SUBSPACE CLUSTERING HAVE BEEN PROPOSED IN THE PAST TWO DECADES.

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

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

where $\boldsymbol{\mu}_i \in \mathbb{R}^D$ is an arbitrary point in subspace S_i that can be chosen as $\boldsymbol{\mu}_i = \boldsymbol{0}$ 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 $\{\boldsymbol{\mu}_i\}_{i=1}^n$, and the segmentation of the points according to the subspaces.

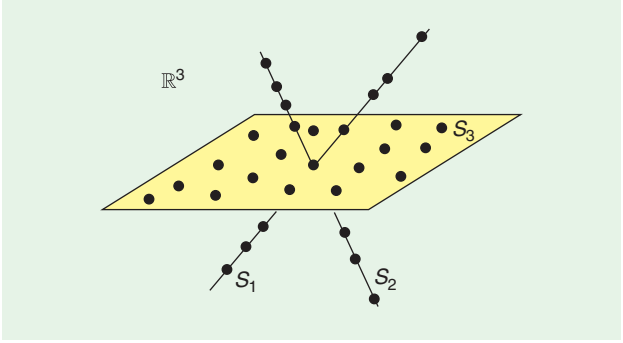
When the number of subspaces is equal to one, this problem reduces to finding a vector $\boldsymbol{\mu} \in \mathbb{R}^D$, a basis $U \in \mathbb{R}^{D \times d}$, a low-dimensional representation $Y = [\boldsymbol{y}_1, \ldots, \boldsymbol{y}_N] \in \mathbb{R}^{d \times N}$, and the dimension d. This problem is known as PCA [28]. (The problem of matrix factorization dates back to the work of Beltrami [29] and Jordan [30]. In the context of stochastic signal processing, PCA is also known as Karhunen-Loeve transform [31]. In the applied statistics literature, PCA is also known as Eckart-Young decomposition [32].) PCA can be solved in a remarkably simple way: $\boldsymbol{\mu} = (1/N)\sum_{j=1}^N \boldsymbol{x}_j$ 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

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

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

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

- First, there is a strong coupling between data segmentation and model estimation. Specifically, if the segmentation of the data is 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.
- Second, the distribution of the data inside the subspaces is generally unknown. If the data within each subspace are distributed around a cluster center and the cluster centers for different subspaces are far apart, the subspace clustering problem reduces to the simpler and well-studied central clustering problem. However, if the distribution of the data points in the subspaces is arbitrary, the subspace clustering problem cannot be solved by central clustering techniques. In addition, the problem becomes more difficult when many points lie close to the intersection of two or more subspaces.
- Third, the position and orientation of the subspaces relative to each other can be arbitrary. As we will show later, when the subspaces are disjoint or independent, the subspace clustering problem can be solved more easily. However, when the subspaces are dependent, the subspace clustering problem becomes much harder. (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.)
- The fourth challenge is that the data can be corrupted by noise, missing entries, and outliers. Although robust estimation techniques for handling such nuisances have been developed for the case of a single subspace, the case of multiple subspaces is not well understood.
- The fifth challenge is model selection. In classical PCA, the only parameter is subspace dimension, which can be found by searching for the subspace of the smallest dimension



[FIG1] A set of sample points in $\mathbb{R}^3$ drawn from a union of three subspaces: two lines and a plane.

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, i.e., 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 dimensions.

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

SUBSPACE CLUSTERING ALGORITHMS

ALGEBRAIC ALGORITHMS

We first review two algebraic algorithms for clustering noise-free data drawn from multiple linear subspaces, i.e., $\boldsymbol{\mu}_i = 0$.

The first algorithm is based on linear algebra, specifically matrix factorization, and is provably correct for independent subspaces. The second one is based on polynomial algebra and is provably correct for both dependent and independent subspaces.

Although these algorithms are designed for linear subspaces, in the case of noiseless data, they can also be applied to affine subspaces by using homogeneous coordinates, thus interpreting an affine subspace of dimension d in $\mathbb{R}^D$ as a linear subspace of dimension $d+1$ in $\mathbb{R}^{D+1}$. (The homogeneous coordinates of $\boldsymbol{x} \in \mathbb{R}^D$ are given by $[\boldsymbol{x}^\top 1]^\top \in \mathbb{R}^{D+1}$.)

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.

MATRIX FACTORIZATION-BASED ALGORITHMS

These algorithms obtain the segmentation of the data from a low-rank factorization of the data matri 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 $r \triangleq \text{rank}(X) = \sum_{i=1}^{n} d_i \leq \min\{D, N\}$ and

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

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 [2] and [33], the matrix Q is such that

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

In the absence of noise, (6) can be used to obtain the segmentation of the data by applying spectral clustering to the eigenvectors of Q [7] (see the "Spectral Clustering-Based Methods" section) or by sorting and thresholding the entries of Q [8], [34]. 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 a subset of the rows of Q. However, as noted in [33] and [35], 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 effect of noise. The entries of Q are first thresholded to obtain an initial oversegmentation of the data. A subspace is then fit to each group G_i, and two groups are merged when the distance between their subspaces is below a threshold. A similar approach is followed by Kanatani et al. [33], [36], except that the geometric Akaike information criterion [37] is used to decide when to merge the two groups.

Although 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

Generalized PCA (GPCA; see [10] and [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.

The first step of GPCA, 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 $d_{\max} = \max\{d_1, \ldots, d_n\}$. (The value of r is determined using model-selection techniques when the subspace dimensions are unknown.) The rationale behind this step is as follows. Since the maximum dimension of each subspace is $d_{\max}$, a projection onto a generic subspace of $\mathbb{R}^D$ of dimension $d_{\max} + 1$ preserves the number and dimensions of the subspaces with probabilty one. As a by-product, the subspace clustering problem is reduced to clustering subspaces of dimension at most $d_{\max}$ in $\mathbb{R}^{d_{\max}+1}$. As we shall see, this step is very important to reduce the computational complexity of the GPCA algorithm. With an abuse of notation, we will denote the original and projected subspaces as S_i, and the original and projected data matrix as

GENERALIZED PCA IS AN ALGEBRAIC-GEOMETRIC METHOD FOR CLUSTERING DATA LYING IN (NOT NECESSARILY INDEPENDENT) LINEAR SUBSPACES.

$$X = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N] \in \mathbb{R}^{D \times N} \text{ or } \mathbb{R}^{r \times N}. \tag{7}$$

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 $b_i \in \mathbb{R}^3$. The union of the two planes can be represented as a 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

$$c_1 x_1^2 + c_2 x_1 x_2 + c_3 x_1 x_3 + c_4 x_2^2 + c_5 x_2 x_3 + c_6 x_3^2 = 0. \tag{8}$$

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

$$M_n(r) = \binom{n+r-1}{n}$$

independent monomials; hence, $\boldsymbol{c} \in \mathbb{R}^{M_n(r)}$.

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

$$\boldsymbol{c}^\top [\nu_n(\boldsymbol{x}_1), \nu_n(\boldsymbol{x}_2), \cdots, \nu_n(\boldsymbol{x}_N)] \stackrel{\Delta}{=} \boldsymbol{c}^\top V_n = \boldsymbol{0}^\top \tag{9}$$

and the number of polynomials is simply the dimension of the null space of 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

$$n = \min\{i : \text{rank}(V_i) = M_i(r) - 1\}. \tag{10}$$

In the case of data contaminated with small-to-moderate amounts of noise, the polynomial coefficients (9) can be found using least squares—the vectors $\boldsymbol{c}$ are the left singular vectors of 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 V_n and, 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 (10), as shown in [10]. However, determining n and $\{d_i\}_{i=1}^n$ for subspaces of different dimensions from noisy data remains a challenge. 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 data set within a certain tolerance, and to [40] and [42] for algebraic relationships among n, $\{d_i\}_{i=1}^n$ and the number of polynomials, which can 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, $\{\boldsymbol{y}_i \in S_i\}_{i=1}^n$, we could compute the n subspace bases $\{U_i\}_{i=1}^n$ from the gradient of the polynomials at $\{\boldsymbol{y}_i\}_{i=1}^n$ and then obtain the segmentation by assigning each point $\{\boldsymbol{x}_j\}_{j=1}^N$ to its closest subspace. A simple method for choosing the points $\{\boldsymbol{y}_i\}_{i=1}^n$ is to select any data point as $\boldsymbol{y}_1$ to obtain the basis U_1 for the first subspace S_1. After removing the points that belong to S_1 from the data set, we can choose any of the remaining data points as $\boldsymbol{y}_2$ to obtain U_2, hence S_2, and then repeat this process until all the subspaces are found. In the "Spectral Clustering-Based Methods" section, 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 may be used to determine n and $\{d_i\}$, as shown in [40].

The first drawback of GPCA is that its complexity increases exponentially with n and $\{d_i\}$. Specifically, each vector $\boldsymbol{c}$ is of dimension $O(M_n(r))$, while there are only $O(r\sum_{i=1}^{n}(r-d_i))$ 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 $p(\boldsymbol{x})$ 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 $\{d_i\}_{i=1}^{n}$ does not handle noisy data. Fifth, while GPCA can be applied to affine subspaces by using homogeneous coordinates, in our experience, this does not work very well when the data are contaminated with noise.

A VERY SIMPLE WAY OF IMPROVING THE PERFORMANCE OF ALGEBRAIC ALGORITHMS IN THE CASE OF NOISY DATA IS TO USE ITERATIVE REFINEMENT.

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, we can obtain a refined estimate of the subspaces and segmentation. This is the basic idea behind the K-planes [11] algorithm, which generalizes the K-means algorithm [44] from data distributed around multiple cluster centers to data drawn from multiple hyperplanes. The K-subspaces algorithm [12], [13] further generalizes K-planes from multiple hyperplanes to multiple affine subspaces of any dimensions and 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 $\{d_i\}_{i=1}^{n}$ 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

$$\min_{\{\mu_i\},\{U_i\},\{y_i\},\{w_{ij}\}} \quad \sum_{i=1}^{n}\sum_{j=1}^{N} w_{ij}\,\|\,\boldsymbol{x}_j-\boldsymbol{\mu}_i-U_i\boldsymbol{y}_j\,\|^2$$
$$\text{subject to} \quad w_{ij}\in\{0,1\} \text{ and } \sum_{i=1}^{n} w_{ij}=1. \qquad (11)$$

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

$$w_{ij}=\begin{cases}1 & \text{if } i=\arg\min_{k=1,\ldots,n}\|\boldsymbol{x}_j-\boldsymbol{\mu}_k-U_k\boldsymbol{y}_j\|^2\\ 0 & \text{else}\end{cases}. \qquad (12)$$

Given $\{w_{ij}\}$, the cost function in (11) decouples as the sum of n cost functions, one per subspace. Since each cost function is identical to that minimized by standard PCA, the optimal values for $\boldsymbol{\mu}_i$, U_i, and $\boldsymbol{y}_j$ are obtained by applying PCA to each group of points. The K-subspaces algorithm then proceeds by alternating between assigning points to subspaces and reestimating 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 a 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 article to reduce the number of restarts needed. We refer the reader to [22] and [45] for two additional initialization methods. Second, K-subspaces is sensitive to outliers, partly due to the use of the ℓ_2-norm. This issue can be addressed using a robust norm, such as the ℓ_1-norm, as done by the median K-flat algorithm [15]. However, this results in a more complex algorithm, which requires solving a robust PCA problem at each iteration. Alternatively, one can resort to nonlinear minimization techniques, which are only guaranteed to converge to a local minimum. Third, K-subspaces requires n and $\{d_i\}_{i=1}^{n}$ 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 [45] and [46] for a more detailed analysis of some of the aforementioned issues.

STATISTICAL METHODS

The approaches described so far seek to cluster the data according to multiple subspaces 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 data inside the subspaces or about the distribution of noise. Therefore, the estimates they provide are not optimal, e.g., in a maximum likelihood (ML) sense. This issue can be addressed by defining a proper generative model for the data, as described next.

MIXTURE OF PROBABILISTIC PCA

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

$$\boldsymbol{x}=\boldsymbol{\mu}+U\boldsymbol{y}+\boldsymbol{\epsilon}, \qquad (13)$$

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 ML estimates of U and σ can be obtained from the SVD of the data matrix X.

PPCA can be naturally extended to a generative model for a union of subspaces $\cup_{i=1}^n S_i$ by using a mixture of PPCA (MPPCA) model [16]. 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 model

$$p(\boldsymbol{x}) = \sum_{i=1}^{n} \pi_i G(\boldsymbol{x}; \boldsymbol{\mu}_i, U_i U_i^\top + \sigma_i^2 I), \qquad \sum_{i=1}^{n} \pi_i = 1, \tag{14}$$

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 $(\tilde{\boldsymbol{\mu}}_i, \tilde{U}_i, \tilde{\sigma}_i, \tilde{\pi}_i)$ for the model parameters, in the E-step, the probability that $\boldsymbol{x}_j$ belongs to subspace i is estimated as

$$\tilde{p}_{ij} = \frac{G(\boldsymbol{x}_j; \boldsymbol{\mu}_i, \tilde{U}_i \tilde{U}_i^\top + \tilde{\sigma}_i^2 I)\tilde{\pi}_i}{p(\boldsymbol{x}_j)}, \tag{15}$$

and in the M-step, the $\tilde{p}_{ij}$s are used to recompute the subspace parameters using PPCA. Specifically, π_i and $\boldsymbol{\mu}_i$ are updated as

$$\tilde{\pi}_i = \frac{1}{N}\sum_{j=1}^{N} \tilde{p}_{ij} \text{ and } \tilde{\boldsymbol{\mu}}_i = \frac{1}{N\tilde{\pi}_i}\sum_{j=1}^{N} \tilde{p}_{ij}\boldsymbol{x}_j, \tag{16}$$

and σ_i and U_i are updated from the SVD of

$$\tilde{\Sigma}_i = \frac{1}{N\tilde{\pi}_i}\sum_{j=1}^{N} \tilde{p}_{ij}(\boldsymbol{x}_j - \tilde{\boldsymbol{\mu}}_i)(\boldsymbol{x}_j - \tilde{\boldsymbol{\mu}}_i)^\top. \tag{17}$$

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 to put a prior on these parameters, as shown in [51]. A second drawback is that MPPCA is not optimal when the data inside each subspace or the noise is not Gaussian. A third drawback is that MPPCA often converges to a local maximum; hence, a good initialization is critical. The initialization problem can be addressed by using any of the methods described earlier for K-subspaces. For example, the multistage learning (MSL) algorithm [17] uses the factorization method of [8] followed by the agglomerative refinement steps of [33] and [36] for initialization.

AN IMPORTANT DRAWBACK OF MPPCA IS THAT THE NUMBER AND DIMENSIONS OF THE SUBSPACES NEED TO BE KNOWN BEFOREHAND.

AGGLOMERATIVE LOSSY COMPRESSION

The agglomerative lossy compression (ALC) algorithm [18] assumes that the data are drawn from a mixture of degenerate Gaussians. However, unlike MPPCA, ALC does not aim to obtain an 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 independent identically distributed (i.i.d.) samples from a zero-mean D-dimensional Gaussian, i.e., $X \in \mathbb{R}^{D \times N}$, up to a distortion δ can be approximated as $[(N + D)/2] \log_2 \det(I + (D/\delta^2 N)XX^\top)$. Thus, the total number of bits for coding a mixture of Gaussians can be approximated as

$$\sum_{i=1}^{n} \frac{N_i + D}{2} \log_2 \det\left(I + \frac{D}{\delta^2 N_i} X_i X_i^\top\right) - N_i \log_2\left(\frac{N_i}{N}\right), \tag{18}$$

where $X_i \in \mathbb{R}^{D \times N_i}$ 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 (18) 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 [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 δ. When δ is chosen to be very large, all the points could be merged into a single group. Conversely, when δ is very small, each point could end up as a separate group. Since δ is related to the variance of the noise, one can use statistics on the data to determine δ (see [22] and [33] for possible methods). When the number of subspaces is known, one can run ALC for several values of δ, discard the values of δ that give the wrong number of subspaces, and choose the δ 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

Random sample consensus (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 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.

RANDOM SAMPLE CONSENSUS IS A STATISTICAL METHOD FOR FITTING A MODEL TO A CLOUD OF POINTS CORRUPTED WITH OUTLIERS IN A STATISTICALLY ROBUST WAY.

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 by fitting 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 to the first subspace.
2) Remove the inliers from the current data set and repeat Step 1 to find the second subspace and so on 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 the 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. As shown in [55], this issue can be addressed by introducing a nonuniform prior in the sampling strategy so that points in the same subspace are more likely to be chosen than points in different subspaces. 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].

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 (j,k)th 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 a different group. 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 $\{U_j\}_{j=1}^N$ are the eigenvectors of L, then $n \ll N$ 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 ones, and the normalized Laplacian, $L_{\text{sym}} = I - \text{diag}(A\mathbf{1})^{-1/2} A \text{diag}(A\mathbf{1})^{-1/2}$. 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 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 review some of the methods for building a pairwise affinity for 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 modifying the affinity or 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

As noticed in [2] and [57], the GPCA algorithm can also be used to define an affinity between two points. Specifically, recall that an estimate $\hat{S}_j$ of the subspace passing through the point $\boldsymbol{x}_j$ can

be obtained from the derivatives of the polynomials $p(\boldsymbol{x})$ at $\boldsymbol{x}_j$. Let θ_{jk}^m be the mth principal angle between $\hat{S}_j$ and $\hat{S}_k$, for $j,k = 1,\ldots,N$. One can use these angles to define an affinity as

$$A_{jk} = \prod_{m=1}^{\min(d_j,d_k)} \cos^2(\theta_{jk}^m). \tag{19}$$

Notice that this affinity is applicable only to linear subspaces, because it only captures the similarity between the subspace bases. To see this, notice that when two affine subspaces are parallel to each other, all their principal angles are equal to zero; hence, A_{jk} is equal to one not only for points j and k in the same subspace, but also for points j and k in two different subspaces. Therefore, in the case of data drawn from affine subspaces, A_{jk} needs to be modified to also incorporate an appropriate distance between points j and k. We will discuss ways to do this in the next paragraph. Given a pairwise affinity, GPCA finds the segmentation of the data by applying spectral clustering to the normalized Laplacian.

LOCAL SUBSPACE AFFINITY AND SPECTRAL LOCAL BEST-FIT FLATS

The local subspace affinity (LSA) [21] and spectral local best-fit flats (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 of 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 and finds K-NN using the angle between the two data points or as a metric. PCA is then used to fit the local subspace $\hat{S}_j$. The subspace dimension d_j is then determined using model-selection techniques. SLBF determines both the number of neighbors K_j and the subspace $\hat{S}_j$ for each 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

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

where θ_{jk}^m is the mth principal angle between the estimated subspaces $\hat{S}_j$ and $\hat{S}_k$. As in the case of the GPCA-based affinity in (19), the affinity in (20) is applicable only to linear subspaces. SLBF addresses this issue by using the affinity

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

where σ_j measures how well point j and its K_j-NNs are fit by $\hat{S}_j$, $\hat{d}_{jk} = \sqrt{\text{dist}(\boldsymbol{x}_j,\hat{S}_k)\text{dist}(\boldsymbol{x}_k,\hat{S}_j)}$, and $\text{dist}(\boldsymbol{x},S)$ is the Euclidean distance from point $\boldsymbol{x}$ to subspace S. Notice that this affinity uses the distance from points to subspaces; thus, it is applicable to both linear and affine subspaces. Given a pairwise affinity, LSA and SLBF find the segmentation of the data by applying spectral clustering to the normalized Laplacian.

The LSA and SLBF algorithms have two main advantages when compared with 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 $O(nd_{\max})$ data points, while GPCA needs $O(M_n(d_{\max}+1))$. 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 $r = \text{rank}(X) \leq \sum d_i$. This is because if the dimension of the projection is too small [less than $\text{rank}(X)$], the projected subspaces become dependent. While in theory, LSA can handle both independent and dependent subspaces, the projection increases the dimension of the intersection of two subspaces; hence, many of the data points could be projected close to the intersection. As a consequence, LSA does not perform as well with dependent subspaces, as the experiments will show. Another major difference between LSA and GPCA is that LSA fits a subspace locally around each projected point, while GPCA uses the gradient of a polynomial that is globally fit to the projected data.

LOCALLY LINEAR MANIFOLD CLUSTERING

The locally linear manifold clustering (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

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

where $\|X\|_F^2 = \sum X_{ij}^2$ is the Frobenius norm of X, 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 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 of L; hence, 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].

SSC RELIES ON THE FACT THAT A POINT IN A LINEAR OR AFFINE SUBSPACE OF DIMENSION *D* CAN ALWAYS BE WRITTEN AS A LINEAR OF AFFINE COMBINATION OF *D* OR *D* + 1 DATA IN THE SAME SUBSPACE.

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 nonlinear 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 has problems with points near the intersections, because it is not always the case that a point and its K-NNs are in the same subspace. Also, properly choosing the number of NNs is a challenge. These issues could be resolved by choosing the neighborhood automatically, as done by SLBF. Finally, even though, in theory, LLMC can handle both dependent and independent subspaces, in practice, it does not perform as well with dependent subspaces for the same reasons as for LSA.

SPARSE SUBSPACE CLUSTERING

Sparse subspace clustering (SSC) [24], [25] is also based on the idea of writing a data point as a linear or affine combination of neighboring data points. However, while LSA, SLBF, and LLMC use the angular or Euclidean distance between two points to choose the K-NNs, SSC uses the principle of sparsity to choose any of the remaining data points ($N - 1 \gg K$) as a possible neighbor. Specifically, SSC relies on the fact that a point in a linear or affine subspace of dimension d can always be written as a linear or affine combination of d or $d + 1$ data points in the same subspace. Therefore, if we write a data point $\boldsymbol{x}_j \in S_i$ as a linear or affine combination of all other $N - 1$ data points $\{\boldsymbol{x}_k\}_{k\neq j}$ drawn from $\cup_{i=1}^n S_i$ with $d_i = \dim(S_i)$, then a sparse linear or affine combination can be obtained by choosing d_i or $d_i + 1$ nonzero coefficients corresponding to points from S_i. This sparse linear or affine combination $\boldsymbol{x}_j = \sum_{k\neq j} w_{jk}\boldsymbol{x}_k$ can be found by minimizing the number of nonzero coefficients w_{jk}, subject to $\sum w_{jk} = 1$ in the case of affine subspaces. Since this problem is combinatorial, the SSC algorithm solves the following simpler ℓ_1 optimization problem instead

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

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, a *sparse representation* is obtained, where each point is written as a linear or affine combination of a few 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 or affine combination of other points. Instead, a penalty in the ℓ_2-norm of the error is added to the ℓ_1 norm. Specifically, the sparse coefficients are found by solving the problem

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

where $\lambda > 0$ is a parameter. Obviously, different solutions for $\{w_{jk}\}$ will be obtained for different choices of the parameter λ. 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 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

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

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

Given a sparse representation for each data point, the pairwise 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 itself as a linear combination of a point that is very far from all of 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 $O(N)$ 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. Moreover, the dimension of the individual subspaces does not need to be known beforehand and can be estimated from the number of nonzero coefficients. More importantly, the SSC algorithm is provably correct for independent [24] and disjoint [25] 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 SSC over GPCA is that it 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 dependent subspaces.

LOW-RANK REPRESENTATION

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

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

This problem can be rewritten in matrix form as

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

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

$$\min_{W,E} \| W \|_1 + \lambda \| E \|_F^2 \quad \text{s.t. } X = XW^\top + E, \text{diag}(W) = \mathbf{0} \text{ (and } W\mathbf{1} = \mathbf{1}). \quad (30)$$

The LRR algorithm aims to minimize rank(W) instead of $\| W \|_1$. Since this rank-minimization problem is nondeterministic polynomial (NP) time 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

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

It can be shown that when the data are 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 from (6) that this matrix is such that $Q_{jk} = 0$ when points j and k are in different subspaces, hence it 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

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

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 Frobenius norms are replaced by the nuclear and $\ell_{2,1}$ 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 LRR is that it provides a theoretical justification for the Costeira and Kanade algorithm. A second advantage is that, similarly to SSC, the optimization problem is convex. One drawback of LLR is that it is provably correct only in the case of noiseless data drawn from independent subspaces. Another drawback is that the optimization problem involves $O(N^2)$ variables.

SPECTRAL CURVATURE CLUSTERING

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 [27], [58], and [59] 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 two points.

Specifically, let $X_{d+2} = \{\boldsymbol{x}_{j_\ell}\}_{j=1}^{d+2}$ be $d+2$ randomly chosen data points. One possible affinity is the volume of the $(d+1)$-simplex formed by these points, 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 spectral curvature clustering (SCC) algorithm [27] is based on the concept of polar curvature, which is also zero when the points are in the same subspace. The multiway affinity $\mathcal{A}_{j_1,j_2,\ldots,j_{d+2}}$ is defined as

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

if $j_1, j_2, \ldots, j_{d+2}$ are distinct and zero otherwise, where diam(X_{d+2}) is the diameter of X_{d+2}. Notice that this affinity is

invariant to scaling of the data points while the volume is not. A pairwise affinity matrix is then defined as

$$A_{jk} = \sum_{j_2,\ldots,j_{d+1}\in\{1,\ldots,N\}} \mathcal{A}_{j,j_2,\ldots,j_{d+2}}\mathcal{A}_{k,j_2,\ldots,j_{d+2}}. \quad (34)$$

This requires computing $O(N^{d+2})$ entries of $\mathcal{A}$ and summing over $O(N^{d+1})$ 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. Reference [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 uses points from the entire data set to define the affinity between two points, while LSA, SLBF, and LLMC restrict themselves to K-NNs. This ultimately results in better affinities because it is less likely that they are built using points from different subspaces. 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 it 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 [60]. Finally, SCC can be extended to nonlinear manifolds by using kernel methods [61]. 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.

ONE ADVANTAGE OF SCC OVER FACTORIZATION-BASED METHODS AND GPCA IS THAT IT CAN HANDLE NOISY DATA DRAWN FROM BOTH LINEAR AND AFFINE SUBSPACES.

APPLICATIONS IN COMPUTER VISION

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 three-dimensional (3-D) affine subspace. To see this, let $\{x_{fj} \in \mathbb{R}^2\}_{j=1,\ldots,N}^{f=1,\ldots,F}$ denote the two-dimensional (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

$$x_{fj} = A_f \begin{bmatrix} X_j \\ 1 \end{bmatrix}, \quad (35)$$

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

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

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 $\text{rank}(M) \le 4$ and $\text{rank}(S) \le 4$ we get

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

Moreover, since the last row of $S^\top$ is one, 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 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 database, which is available online at http://www.vision.jhu.edu/data/hopkins155 [57]. 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 3-D space, hence the motion trajectories are

expected to lie in independent affine subspaces of dimension three. The traffic sequences contain cars moving independently on the ground plane, hence the motion trajectories are expected to 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 expected to be 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.

MOTION SEGMENTATION REFERS TO THE PROBLEM OF SEPARATING A VIDEO SEQUENCE INTO MULTIPLE SPATIOTEMPORAL REGIONS CORRESPONDING TO DIFFERENT RIGID-BODY MOTIONS.

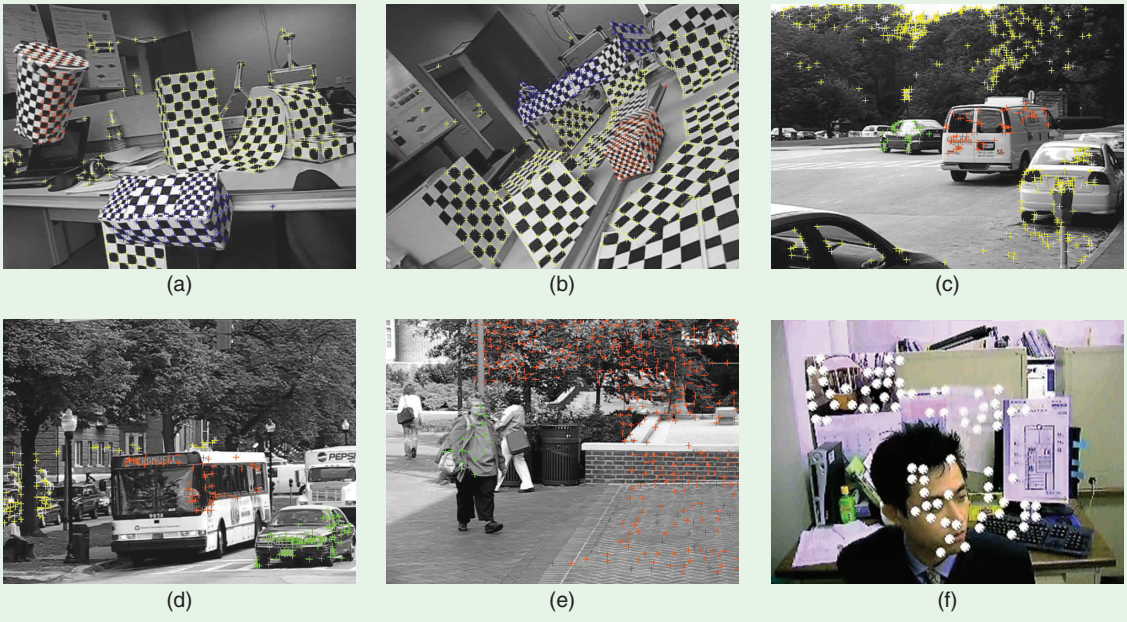
To make our results comparable to those in the existing literature, for each method we apply the same preprocessing steps described in their respective articles. Specifically, we project the trajectories onto a subspace of dimension $r \leq 2F$ using either PCA (GPCA, RANSAC, LLMC, LSA, ALC, and SCC) 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 four-dimensional (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 choice 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 ten 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

$$\text{Classification error} = \frac{\text{number of misclassified points}}{\text{total number of points}} \times 100\%. \quad (38)$$

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.



[FIG2] Sample images from some sequences in the database with tracked points superimposed: (a) 1R2RCT_B, (b) 2T3RCRT, (c) cars3, (d) cars10, (e) people2, and (f) kanatani3.

[TABLE 1] CLASSIFICATION ERRORS OF SEVERAL SUBSPACE CLUSTERING ALGORITHMS ON THE HOPKINS 155 MOTION SEGMENTATION DATABASE.

	TWO MOTIONS								THREE MOTIONS									
	CHECK. (78)		TRAFFIC (31)		ARTICUL. (11)		ALL (120)		CHECK. (26)		TRAFFIC (7)		ARTICUL. (2)		ALL (35)		ALL (155)	
	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

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-D affine subspaces, i.e., $d = 3$, while linear subspace clustering algorithms treat subspaces as four-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 an sp dimension for the projection. All algorithms use PCA to perform the projection, except for SSC that uses a random projection with entries drawn from SSC-B or 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.

GPCA

To avoid using multiple polynomials, we use an implementation of GPCA based on hyperplanes in which the data are 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 a 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 suggests 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 by more than 1,000 trials for each sequence.

LSA

When the dimension of the projection is chosen as $r = 5$, this algorithm performs worse than GPCA. This is because the 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 an NN. 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 better 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.

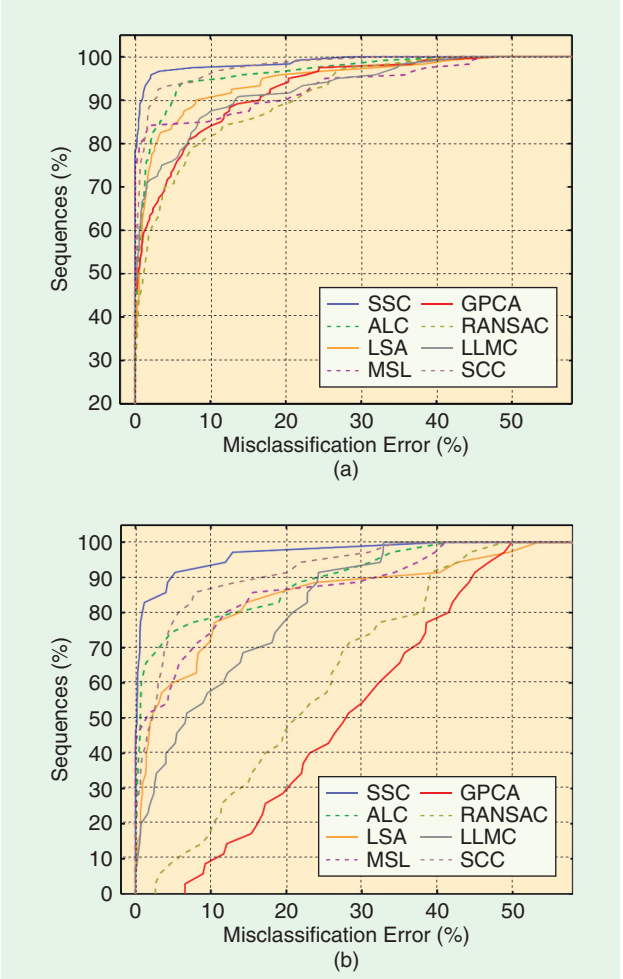
However, sometimes the MSL method gets stuck in a local minimum. This is reflected by high classification errors for some sequences, as can be seen by the long tails in Figure 3.

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 δ 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



[FIG3] 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), and SSC-N (4,4n). (a) Two motions. (b) Three motions.

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 δ. 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

[TABLE 2] MEAN PERCENTAGE OF MISCLASSIFICATION ON CLUSTERING YALE FACE 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
SCC	0.0	0.0	0.0	1.1	2.7	2.1	2.2	5.7	6.6
SSC	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2.4	4.6
SLBF	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.2	0.9
ALC	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

SSC is not very sensitive to the choice of the projection (Bernoulli versus normal), though SSC-N gives slightly better results. We have also observed that SSC is not sensitive to the dimension of the projection ($r = 5$ versus $r = 4n$ versus $r = 2F$) or the parameter λ.

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.

FACE CLUSTERING UNDER VARYING ILLUMINATION

Given a set of images $\{I_j \in \mathbb{R}^D\}_{j=1}^N$ of n different faces taken from the same viewpoint under varying illumination conditions, the face clustering problem consists of clustering the images according to the identity of the 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 set of images according to multiple subspaces.

Table 2 shows the experiments from [22], which evaluate the performance of 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 ten faces taken under nine different viewpoints and 64 different illumination conditions. Nine subsets containing the images of the frontal views of the following $n = 2, \ldots, 10$ individuals are considered: {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}. For computational efficiency, the images are downsampled to 120×160 pixels. Since this number 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 $d = 2$.

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

GPCA

This algorithm does not perform very well. This is attributed to the fact that it is very hard to distinguish faces from only five dimensions. While one could have chosen to project the faces to a space of larger dimension, GPCA cannot handle a large number of variables, especially as the number of groups increases.

SCC

This algorithm performs better than GPCA, achieving a perfect classification for $n \leq 4$. However, as n increases from 5 to 10, the classification error ranges from 1.1% to 6.6%.

SSC

This algorithm performs very well, achieving perfect classification for $n \leq 8$ and classification errors of 2.4% and 4.6% for $n = 9$ and $n = 10$, respectively.

SLBF

This algorithm performs very well, slightly better than SSC. It achieves perfect classification for $n \leq 8$ and errors of 1.2% and 0.9% for $n = 9$ and $n = 10$, respectively.

ALC

This algorithm performs extremely well, achieving 100% accuracy in all cases. However, this requires using the algorithm from [22] to set the parameter δ. When multiple values of δ are chosen, the error goes up to 50% for $n = 2$ and stays at 0% in other cases, as reported in [22].

Although these experiments show very promising results, we believe there is still plenty of room for improvement. For example, the face clustering problem is more challenging from nonfrontal faces, thus it would be natural to evaluate the algorithms for nonfrontal faces and see if their performance deteriorates. Also, many of the images in the Yale faces B database contain not only faces but also background, which can facilitate the clustering of the images using the background intensities. Thus, it would be natural to evaluate the algorithms on the cropped images and see if their performance deteriorates. Finally, one could also explore several choices for the subspace dimensions d and for the dimension of the projection D.

CONCLUSIONS AND FUTURE DIRECTIONS

Over the past few decades, significant progress has been made in clustering high-dimensional data sets 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 and face clustering problems in computer vision.

While earlier algorithms were designed under the assumptions of perfect data and 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.

Some algorithms are provably correct for independent subspaces, others are provably correct for disjoint subspaces, and others are able to handle an unknown number of subspaces of unknown dimensions in an arbitrary configuration. However, a theoretical analysis of the applicability of many methods to affine subspaces in the noiseless case is still due.

IN OUR VIEW, THE GRAND CHALLENGE FOR THE NEXT DECADE WILL BE TO DEVELOP CLUSTERING ALGORITHMS FOR DATA DRAWN FROM MULTIPLE NONLINEAR MANIFOLDS.

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] and [60]. 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 [15] and [63] 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]–[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.

ACKNOWLEDGMENTS

This work has been supported by the National Science Foundation (NSF) IIS-0447739 and Office of Naval Research (ONR) N000140510836. The author thanks Prof. Yi Ma, Prof. Richard Hartley, Dr. Alvina Goh, Dr. Shankar Rao, Mr. Ehsan Elhamifar, and Mr. Roberto Tron for their numerous contributions to this work. The author also thanks Prof. Gilad Lerman for organizing a wonderful workshop on multimanifold data analysis at the Institute for Mathematics and its Applications of the University of Minnesota and his students Mr. Guangliang Chen and Mr. Teng Zhang for answering numerous questions about their work. The author also thanks an anonymous reviewer for his/her very insightful comments, which have significantly improved this manuscript.

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 Johns Hopkins University in 2004 as a faculty member in the Department of Biomedical Engineering and Center for Imaging Science. He was coeditor of the book *Dynamical Vision* and has coauthored more than 100 articles in biomedical image analysis, computer vision, machine learning, hybrid systems, and robotics. He is a 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 National Aeronautics and Space Administration, 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 Association for Computing Machinery.

REFERENCES

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

[2] R. Vidal, R. Tron, and R. Hartley, "Multiframe motion segmentation with missing data using power factorization and GPCA," *Int. J. Comput. Vis.*, 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 *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2003, pp. 11–18.

[4] W. Hong, J. Wright, K. Huang, and Y. Ma, "Multi-scale hybrid linear models for lossy image representation," *IEEE Trans. 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 *Proc. Conf. 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 Explor. Newslett.*, vol. 6, no. 1, pp. 90–105, 2004.

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

[8] J. Costeira and T. Kanade, "A multibody factorization method for independently moving objects," *Int. J. Comput. Vis.*, vol. 29, no. 3, pp. 159–179, 1998.

[9] C. W. Gear, "Multibody grouping from motion images," *Int. J. Comput. Vis.*, vol. 29, no. 2, pp. 133–150, 1998.

[10] R. Vidal, Y. Ma, and S. Sastry, "Generalized principal component analysis (GPCA)," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 27, no. 12, pp. 1–15, 2005.

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

[12] P. Tseng, "Nearest q-flat to m points," *J. Optim. Theory Applicat.*, vol. 105, no. 1, pp. 249–252, 2000.

[13] P. Agarwal and N. Mustafa, "k-means projective clustering," in *Proc. ACM SIGMOD-SIGACT-SIGART Symp. Principles of Database Systems*, 2004, pp. 155–165.

[14] L. Lu and R. Vidal, "Combined central and subspace clustering on computer vision applications," in *Proc. Int. Conf. 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 *Proc. Workshop Subspace Methods*, 2009, pp. 234–241.

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

[17] Y. Sugaya and K. Kanatani, "Geometric structure of degeneracy for multi-body motion segmentation," in *Proc. Workshop Statistical Methods in Video Processing*, 2004, pp. 13–25.

[18] Y. Ma, H. Derksen, W. Hong, and J. Wright, "Segmentation of multivariate mixed data via lossy coding and compression," *IEEE Trans. Pattern Anal. Machine Intell.*, 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 *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2008.

[20] A. Y. Yang, S. Rao, and Y. Ma, "Robust statistical estimation and segmentation of multiple subspaces," in *Proc. Workshop 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 *Proc. European Conf. 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 *Proc. IEEE Conf. 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 *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2007.

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

[25] E. Elhamifar and R. Vidal, "Clustering disjoint subspaces via sparse representation," in *Proc. IEEE Int. Conf. Acoustics, Speech, and Signal Processing*, 2010, pp. 1926–1929.

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

[27] G. Chen and G. Lerman, "Spectral curvature clustering (SCC)," *Int. J. Comput. Vis.*, vol. 81, no. 3, pp. 317–330, 2009.

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

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

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

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

[32] C. Eckart and G. Young, "The approximation of one matrix by another of lower rank," *Psychometrika*, vol. 1, no. 3, pp. 211–218, 1936.

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

[34] N. Ichimura, "Motion segmentation based on factorization method and discriminant criterion," in *Proc. IEEE Int. Conf. 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 *Proc. IEEE Conf. 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 *Proc. European Conf. Computer Vision*, 2002, pp. 25–31.

[37] K. Kanatani, "Geometric information criterion for model selection," *Int. J. Comput. Vis.*, vol. 26, no. 3, pp. 171–189, 1998.

[38] L. Zelnik-Manor and M. Irani, "On single-sequence and multi-sequence factorizations," *Int. J. Comput. Vis.*, 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 Rev.*, vol. 50, no. 3, pp. 413–458, 2008.

[40] H. Derksen, "Hilbert series of subspace arrangements," *J. Pure Appl. 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 *Proc. IEEE Conf. 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 *Proc. IEEE Int. Conf. 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 *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2004, vol. 2, pp. 631–638.

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

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

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

[47] M. Tipping and C. Bishop, "Probabilistic principal component analysis," *J. Royal Stat. Soc.*, 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," *J. Royal Stat. Soc. B*, vol. 39, no. 1, 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 *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2004, vol. 1, pp. 707–714.

[51] J. Paisley and L. Carin, "Nonparametric factor analysis with beta process priors," in *Proc. Int. Conf. Machine Learning*, 2009, pp. 777–780.

[52] A. Leonardis, H. Bischof, and J. Maver, "Multiple eigenspaces," *Pattern Recognit.*, 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. Pattern Anal. Machine Intell.*, 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," *Commun. ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[55] J. Yan and M. Pollefeys, "Articulated motion segmentation using RANSAC with priors," in *Proc. Workshop Dynamical Vision*, 2005, pp. 75–85.

[56] U. von Luxburg, "A tutorial on spectral clustering," *Stat. Comput.*, vol. 17, no. 4, pp. 395–416, 2007.

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

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

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

[60] G. Chen and G. Lerman, "Foundations of a multi-way spectral clustering framework for hybrid linear modeling," *Foundat. Comput. Math.*, vol. 9, no. 5, pp. 517–558, 2009.

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

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

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

[64] R. Vidal, Y. Ma, S. Soatto, and S. Sastry, "Two-view multibody structure from motion," *Int. J. Comput. Vis.*, 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," *J. Math. Imag. Vis.*, vol. 25, no. 3, pp. 403–421, 2006.

[66] R. Vidal and R. Hartley, "Three-view multibody structure from motion," *IEEE Trans. Pattern Anal. Machine Intell.*, 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," *Int. J. Comput. Vis.*, vol. 88, no. 3, pp. 425–446, 2010. [SP]