A New GPCA Algorithm for Clustering Subspaces by Fitting, Differentiating and Dividing Polynomials*

René Vidal
Center for Imaging Science
Johns Hopkins University
rvidal@cis.jhu.edu

Yi Ma
Electrical & Computer Engineering
U. of Illinois at Urbana-Champaign
yima@uiuc.edu

Jacopo Piazzi
Mechanical Engineering
Johns Hopkins University
piazzi@jhu.edu

Abstract

We consider the problem of clustering data lying on multiple subspaces of unknown and possibly different dimensions. We show that one can represent the subspaces with a set of polynomials whose derivatives at a data point give normal vectors to the subspace associated with the data point. Since the polynomials can be estimated linearly from data, subspace clustering is reduced to classifying one point per subspace. We do so by choosing points in the data set that minimize a distance function. A basis for the complement of each subspace is then recovered by applying standard PCA to the set of derivatives (normal vectors) at those points. The final result is a new GPCA algorithm for subspace clustering based on simple linear and polynomial algebra. Our experiments show that our method outperforms existing algebraic algorithms based on polynomial factorization and provides a good initialization to iterative techniques such as K-subspace and EM. We also present applications of GPCA on computer vision problems such as vanishing point detection, face clustering, and news video segmentation.

1 Introduction

Many segmentation problems in computer vision involve the simultaneous estimation of multiple subspaces from sample data points, without knowing which points corresponds to which subspace (see Figure 1). For example, the segmentation of dynamic scenes involves the estimation of multiple 2-D or 3-D motion models from optical flow or point correspondences in two or more views [4, 9, 10].

Subspace clustering is a challenging problem that is usually regarded as "chicken-and-egg." If the segmentation of the data was known, one could easily fit a single subspace to each group of points using standard PCA [**?**]. Conversely, if the subspace bases were known, one could easily find the data points that best fit each subspace. Since in practice neither the subspaces nor the clustering of the data are known, most of the existing approaches randomly choose a basis for each subspace, and then iterate between data clustering and subspace estimation. This can be done using, e.g., K-subspace [3], an extension of K-means to the case of subspaces, subspace growing and subspace selection [6], or Expectation Maximization (EM) for mixtures of PCA's [7]. However, the performance of iterative approaches to subspace clustering is in general very sensitive to initialization. In fact, multiple starts are needed in order to obtain a good solution, which greatly increases the computational cost.

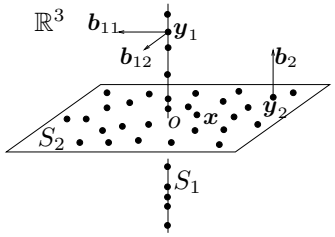

Figure 1: Data samples drawn from a mixture of one plane and one line (through the origin o) in $\mathbb{R}^3$. Arrows are normal vectors.

The initialization problem has motivated the recent development of algebraic and geometric approaches to subspace clustering that do *not* require initialization. When the subspaces are orthogonal, of equal dimensions and intersect only at the origin, one can define a similarity matrix from which the segmentation of the data can be obtained using spectral clustering techniques [1]. Once the clustering of the data has been found, a basis for each subspace is estimated using standard PCA. Unfortunately, this method is sensitive to noise in the data, as pointed out in [5, 12], where various improvements are proposed. When the subspaces are of co-dimension one, i.e. *hyperplanes*, the above method does not apply because the subspaces need not be orthogonal and their intersection is *nontrivial*. A purely algebraic solution to this problem, called Generalized PCA (GPCA), was recently proposed in [8]. It was shown that one can model the hyperplanes with a homogeneous polynomial that can be estimated linearly from data. The clustering problem is shown to be equivalent to factorizing this polynomial into a product of linear factors, where each factor corresponds to one of the hyperplanes. A *polynomial factorization algorithm* (PFA) based on linear algebraic techniques is then used to cluster the hyperplanes. This can be done in closed form when the number of hyperplanes is less than or equal to four. However, the PFA involves computing roots of univariate polynomials, which can be sensitive to noise as the number of hyperplanes (degree of the polynomial) increases.

*We thank Frederik Schaffalitzky for a fruitful discussion on the topic.

Paper contributions. We propose an algebraic geometric approach to subspace clustering based on *polynomial differentiation* rather than *polynomial factorization*. Unlike prior work, we do not restrict the subspaces to be orthogonal, have a trivial intersection, or known and equal dimensions. Instead, we consider the most general case of subspaces of *unknown* and possibly *different* dimensions (e.g., Figure 1).

We model the collection of subspaces as an algebraic set represented by a set of homogeneous polynomials in several variables. We show that one can estimate the set of vectors normal to each subspace by evaluating the derivatives of these polynomials at any point lying on the subspace. Since all the polynomials can be retrieved linearly from the data, this reduces subspace clustering to the problem of classifying one point per subspace. When those points are given (e.g., in semi-supervised learning), this means that in order to learn the mixture of subspaces, it is sufficient to have one positive example per class. When the data is unlabeled (e.g., in unsupervised learning), we propose a simple algorithm that chooses points in the data set that minimize the distance to the algebraic set, hence dealing automatically with noise in the data. A basis for the complement of each subspace is then obtained by applying standard PCA to the set of derivatives (normal vectors) at those points.

Our experiments on low-dimensional data show that our algorithm gives about half the error of the PFA of [8], and improves the performance of iterative techniques, such as K-subspace and EM, by about 50% with respect to random initialization. The best performance is achieved by using our algorithm to initialize K-subspace and EM. We also present computer vision applications on vanishing point detection, face clustering, and news video segmentation.

2 An introductory example

Imagine we are given data in $\mathbb{R}^3$ drawn from the line $S_1 = \{\boldsymbol{x} : x_1 = x_2 = 0\}$ and the plane $S_2 = \{\boldsymbol{x} : x_3 = 0\}$, as shown in Figure 1. We can describe the entire data set as:

$$\begin{aligned} S_1 \cup S_2 &= \{\boldsymbol{x} : (x_1 = x_2 = 0) \vee (x_3 = 0)\} \\ &= \{\boldsymbol{x} : (x_1 x_3 = 0) \wedge (x_2 x_3 = 0)\}. \end{aligned}$$

Therefore, even though each individual subspace is described with polynomials of degree one (linear equations), the mixture of two subspaces is described with the two polynomials of degree two $p_{21}(\boldsymbol{x}) = x_1 x_3$ and $p_{22}(\boldsymbol{x}) = x_2 x_3$. More generally, any two subspaces in $\mathbb{R}^3$ can be represented as the set of points satisfying some polynomials of the form

$$c_1 x_1^2 + c_2 x_1 x_2 + c_3 x_1 x_3 + c_4 x_2^2 + c_5 x_2 x_3 + c_6 x_3^2 = 0.$$

Although these polynomials are nonlinear in each data point $[x_1, x_2, x_3]^T$, they are actually linear in the vector of coefficients $\boldsymbol{c} = [c_1, \ldots, c_6]^T$. Therefore, given enough data points, one can linearly *fit* these *polynomials* to the *data*.

Given these collection of polynomials, we are now interested in estimating a basis for each subspace. In our example, let $P_2(\boldsymbol{x}) = [p_{21}(\boldsymbol{x}) \;\; p_{22}(\boldsymbol{x})]$ and consider the derivatives of $P_2(\boldsymbol{x})$ at two points in each of the subspaces $\boldsymbol{y}_1 = [0, 0, 1]^T \in S_1$ and $\boldsymbol{y}_2 = [1, 1, 0]^T \in S_2$:

$$DP_2(\boldsymbol{x}) = \begin{bmatrix} x_3 & 0 \\ 0 & x_3 \\ x_1 & x_2 \end{bmatrix} \Rightarrow DP_2(\boldsymbol{y}_1) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}, DP_2(\boldsymbol{y}_2) = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 1 \end{bmatrix}.$$

Then the columns of $DP_2(\boldsymbol{y}_1)$ span $S_1^\perp$ and the columns of $DP_2(\boldsymbol{y}_2)$ span $S_2^\perp$ (see Figure 1). Thus the dimension of the line is given by $k_1 = 3 - \text{rank}(DP_2(\boldsymbol{y}_1)) = 1$, and the dimension of the plane is $k_2 = 3 - \text{rank}(DP_2(\boldsymbol{y}_2)) = 2$. In conclusion, if we are given one point in each subspace, we can obtain the *subspace bases* and their *dimensions* from the *derivatives of the polynomials* at these points.

The final question is how to find one point per subspace. With perfect data, we may choose a first point as any of the points in the data set. With noisy data, we may first define a distance from any point in $\mathbb{R}^3$ to one of the subspaces, e.g., $d_2(\boldsymbol{x})^2 = p_{21}(\boldsymbol{x})^2 + p_{22}(\boldsymbol{x})^2 = (x_1^2 + x_2^2)x_3^2$, and then choose a point in the data set that minimizes this distance. Say we pick $\boldsymbol{y}_2 \in S_2$ as such point. We can then compute the normal vector $\boldsymbol{b}_2 = [0, 0, 1]^T$ to S_2 from $DP(\boldsymbol{y}_2)$ as above. How do we now pick a point in S_1 but not in S_2? As it turns out, this can be done by *polynomial division*. We can just divide the original polynomials of degree $n = 2$ by $\boldsymbol{b}_2^T \boldsymbol{x}$ to obtain polynomials of degree $n - 1 = 1$

$$p_{11}(\boldsymbol{x}) = \frac{p_{21}(\boldsymbol{x})}{\boldsymbol{b}_2^T \boldsymbol{x}} = x_1 \quad \text{and} \quad p_{12}(\boldsymbol{x}) = \frac{p_{22}(\boldsymbol{x})}{\boldsymbol{b}_2^T \boldsymbol{x}} = x_2.$$

Since these new polynomials vanish on S_1 but not on S_2, we can find a point $\boldsymbol{y}_1$ in S_1 but not in S_2, as a point in the data set that minimizes $d_1^2(\boldsymbol{x}) = p_{11}(\boldsymbol{x})^2 + p_{12}(\boldsymbol{x})^2 = x_1^2 + x_2^2$.

More generally, one can solve the problem of clustering a collection of n subspaces $\{S_i \subset \mathbb{R}^K\}_{i=1}^n$ of *unknown* and possibly *different* dimensions $\{k_i\}_{i=1}^n$ by *polynomial fitting* (Section 3.1), *differentiation* (Section 3.2) and *division* (Section 3.3), as we will show in the next section.

3. Algebraic subspace clustering

Let $\boldsymbol{x}$ be a vector in $\mathbb{R}^K$. A homogeneous polynomial of degree n in $\boldsymbol{x}$ is a polynomial $p_n(\boldsymbol{x})$ such that $p_n(\lambda \boldsymbol{x}) = \lambda^n p_n(\boldsymbol{x})$ for all λ in $\mathbb{R}$. The space of all homogeneous polynomials of degree n in K variables is a vector space of dimension $M_n = \binom{n+K-1}{K-1}$. A particular basis for this space is obtained by considering all the monomials of degree n in K variables, that is $\boldsymbol{x}^I = x_1^{n_1} x_2^{n_2} \cdots x_K^{n_K}$ with $0 \leq n_j \leq n$ for $j = 1, \ldots, K$, and $n_1 + n_2 + \cdots + n_K = n$. Thus, each homogeneous polynomial can be written as a linear combination of a vector of coefficients $\boldsymbol{c}_n \in \mathbb{R}^{M_n}$ as

$$p_n(\boldsymbol{x}) = \boldsymbol{c}_n^T \nu_n(\boldsymbol{x}) = \sum c_{n_1, n_2, \ldots, n_K} x_1^{n_1} x_2^{n_2} \cdots x_K^{n_K}, \quad (1)$$

where $\nu_n : \mathbb{R}^K \to \mathbb{R}^{M_n}$ is the *Veronese map* of degree n [8], also known as the *polynomial embedding* in machine learning, defined as $\nu_n : [x_1, \ldots, x_K]^T \mapsto [\ldots, \boldsymbol{x}^I, \ldots]^T$ with I chosen in the degree-lexicographic order.

3.1. Representing subspaces with polynomials

It is well known that a subspace $S_i \subset \mathbb{R}^K$ of dimension k_i, where $0 < k_i < K$ can be represented with $K - k_i$ linear equations (polynomials of degree *one*) of the form[1]

$$S_i = \{\boldsymbol{x} \in \mathbb{R}^K : B_i^T \boldsymbol{x} = 0\} = \Big\{\boldsymbol{x} \in \mathbb{R}^K : \bigwedge_{j=1}^{K-k_i} (\boldsymbol{b}_{ij}^T \boldsymbol{x} = 0)\Big\},$$

where $B_i \doteq [\boldsymbol{b}_{i1}, \ldots, \boldsymbol{b}_{i(K-k_i)}] \in \mathbb{R}^{K \times (K-k_i)}$ is a basis for the orthogonal complement of S_i, $S_i^\perp$. We now demonstrate that one can also represent a collection of n subspaces $\{S_i \subset \mathbb{R}^K\}_{i=1}^n$ with a set of polynomials, except that the polynomials are of degree n. To see this, we notice that an arbitrary point $\boldsymbol{x} \in \mathbb{R}^K$ belongs to $\cup_{i=1}^n S_i$ if and only if it satisfies $(\boldsymbol{x} \in S_1) \vee \cdots \vee (\boldsymbol{x} \in S_n)$. This is equivalent to

$$\bigvee_{i=1}^{n} (\boldsymbol{x} \in S_i) \Leftrightarrow \bigvee_{i=1}^{n} \bigwedge_{j=1}^{K-k_i} (\boldsymbol{b}_{ij}^T \boldsymbol{x} = 0) \Leftrightarrow \bigwedge_{\sigma} \bigvee_{i=1}^{n} (\boldsymbol{b}_{i\sigma(i)}^T \boldsymbol{x} = 0),$$

where the right hand side is obtained by exchanging ands and ors using De Morgan's laws, and σ represents a particular choice of one normal vector $\boldsymbol{b}_{i\sigma(i)}$ from each basis B_i. Notice that each one of the above equations is of the form

$$\bigvee_{i=1}^{n} (\boldsymbol{b}_{i\sigma(i)}^T \boldsymbol{x} = 0) \Leftrightarrow \Big(\prod_{i=1}^{n} (\boldsymbol{b}_{i\sigma(i)}^T \boldsymbol{x}) = 0\Big) \Leftrightarrow (p_{n\sigma}(\boldsymbol{x}) = 0),$$

which is simply a homogeneous polynomial of degree n in K variables that is factorizable as a product of n linear expressions in $\boldsymbol{x}$. Therefore, we can write each one of those polynomials as a linear combination of a vector of coefficients $\boldsymbol{c}_n \in \mathbb{R}^{M_n}$ as in (1). We have shown the following.

Theorem 1 (Representing subspaces with polynomials) *A collection of n subspaces can be described as the set of points satisfying a set of homogeneous polynomials of the form*

$$p_n(\boldsymbol{x}) = \prod_{i=1}^{n} (\boldsymbol{b}_i^T \boldsymbol{x}) = \boldsymbol{c}_n^T \nu_n(\boldsymbol{x}) = 0, \tag{2}$$

where $\boldsymbol{b}_i \in \mathbb{R}^K$ is a normal vector to the i-th subspace.

Let $\boldsymbol{X} \doteq \{\boldsymbol{x}^j\}_{j=1}^N$ be given a set of points lying on the n subspaces. Since each polynomial $p_n(\boldsymbol{x}) = \boldsymbol{c}_n^T \nu_n(\boldsymbol{x})$ must be satisfied by every data point, we have $\boldsymbol{c}_n^T \nu_n(\boldsymbol{x}^j) = 0$ for all $j = 1, \ldots, N$. Therefore, the vector of coefficients of each one of the polynomials in the set $I_n \doteq \text{span}_\sigma\{p_{n\sigma}\}$ must lie in the null space of the matrix

$$L_n \doteq [\nu_n(\boldsymbol{x}^1), \ldots, \nu_n(\boldsymbol{x}^N)]^T \in \mathbb{R}^{N \times M_n}. \tag{3}$$

Therefore, if we denote $m_n \doteq \dim(I_n)$, then we must have

$$M_n - \prod_{i=1}^{n} (K - k_i) \leq \text{rank}(L_n) = M_n - m_n \leq M_n - 1. \tag{4}$$

This means that, given $M_n - m_n$ sample points from the algebraic set $\cup_{i=1}^n S_i$, with at least k_i points in subspace S_i, we can obtain a basis $\{p_{n\ell}\}_{\ell=1}^{m_n}$ for the polynomials in I_n by computing its coefficients *linearly* as null vectors of L_n.

[1] For affine subspaces (which do not necessarily pass the origin), we first make them subspaces using the homogeneous coordinates.

Remark 1 (Estimating the polynomials from noisy data points) *In the absence of noise, we can estimate the number of polynomials from a generic data set $\boldsymbol{X}$ as $m_n = M_n - \text{rank}(L_n)$ and the polynomials $\{p_{n\ell}\}$ as the m_n singular vectors of L_n associated with its m_n zero singular values. In the presence of moderate noise, this suggests a least squares solution for the coefficients of the polynomials, except that we do not know m_n. We use model selection to determine m_n as*

$$m_n = \arg\min_m \frac{\sigma_{m+1}^2(L_n)}{\sum_{k=1}^m \sigma_k^2(L_n)} + \kappa\, m, \tag{5}$$

with $\sigma_k(L_n)$ the k^{th} singular vector of L_n and κ a parameter.

3.2. Obtaining a basis for each subspace

For the sake of simplicity, let us first consider the case of hyperplanes, i.e. subspaces of equal dimension $k_i = K - 1$. In this case, there is only one vector $\boldsymbol{b}_i \in \mathbb{R}^K$ normal to subspace S_i, for $i = 1, \ldots, n$. Therefore, there is only one polynomial, $p_n(\boldsymbol{x}) = (\boldsymbol{b}_1^T \boldsymbol{x}) \cdots (\boldsymbol{b}_n^T \boldsymbol{x}) = \boldsymbol{c}_n^T \nu_n(\boldsymbol{x})$, representing the n hyperplanes and its coefficients $\boldsymbol{c}_n$ can be computed as the unique vector in the null space of L_n. Given $\boldsymbol{c}_n$, computing the normal vectors $\{\boldsymbol{b}_i\}$ is equivalent to factorizing $p_n(\boldsymbol{x})$ into a product of linear forms as in (2). In [8], we solved this problem using a polynomial factorization algorithm (PFA) that computes the roots of a univariate polynomial of degree n and solves $K - 2$ linear systems.

Unfortunately, one may not be able to apply the PFA in the presence of noisy data, because $\boldsymbol{c}_n$ may correspond to a polynomial that is not factorizable. The same situation occurs in the case of subspaces of unknown and different dimensions, even with perfect data, because here we can only obtain a *basis* for the polynomials in I_n and the elements of this basis may not be factorizable. For example $x_1^2 + x_1 x_2$ and $x_2^2 - x_1 x_2$ are factorizable but their linear combination $x_1^2 + x_2^2$ is not. One way of avoiding this problem is to find another basis for the null space of L_n whose elements correspond to coefficients of factorizable polynomials. This is in general a daunting task, since it is equivalent to solving a set of multivariate homogeneous polynomials of degree n.

In this section, we propose a new algorithm for clustering subspaces which is based on *polynomial differentiation* rather than *polynomial factorization*. In essence, we show that one can recover a basis for the orthogonal complement of each subspace $\{B_i\}_{i=1}^n$ by differentiating all the polynomials $\{p_{n\ell}\}$ obtained from null(L_n) (factorizable or not).

We will first illustrate our polynomial differentiation algorithm in the case of hyperplanes. Consider the derivatives of the polynomial $p_n(\boldsymbol{x})$ representing the hyperplanes:

$$Dp_n(\boldsymbol{x}) = \frac{\partial p_n(\boldsymbol{x})}{\partial \boldsymbol{x}} = \frac{\partial}{\partial \boldsymbol{x}} \prod_{i=1}^{n} (\boldsymbol{b}_i^T \boldsymbol{x}) = \sum_{i=1}^{n} (\boldsymbol{b}_i) \prod_{\ell \neq i} (\boldsymbol{b}_\ell^T \boldsymbol{x}). \tag{6}$$

We notice that if we evaluate Dp_n at a point $\boldsymbol{y}_i \in S_i$, i.e. $\boldsymbol{y}_i$ is such that $\boldsymbol{b}_i^T \boldsymbol{y}_i = 0$, then all summands in (6), except the i^{th}, vanish, because $\prod_{\ell \neq i} (\boldsymbol{b}_\ell^T \boldsymbol{y}_j) = 0$ for $j \neq i$. Therefore, we can immediately obtain the normal vectors as

$$\boldsymbol{b}_i = \frac{Dp_n(\boldsymbol{y}_i)}{\|Dp_n(\boldsymbol{y}_i)\|}, \quad i = 1, \ldots, n. \tag{7}$$

Therefore, in a *semi-supervised learning scenario* in which we are given only *one positive example per class*, the hyperplane clustering problem can be solved *analytically* by simply evaluating the *derivatives* of $p_n(\boldsymbol{x})$ at each one of the points with known labels.

As it turns out, the same principle applies to subspaces of arbitrary dimensions. In this case, we can compute m_n (not necessarily factorizable) polynomials representing the union of the n subspaces. By construction, even though each vector of coefficients $\boldsymbol{c}_n \in \text{null}(L_n)$ may not correspond to a factorizable polynomial, we can still write $\boldsymbol{c}_n$ as a linear combination of vectors $\boldsymbol{c}_{n\ell}$ which correspond to factorizable polynomials, i.e. $\boldsymbol{c}_n = \sum \alpha_\ell \boldsymbol{c}_{n\ell}$. Then the derivative at a point $\boldsymbol{y}_i \in S_i$ is

$$\frac{\partial}{\partial \boldsymbol{x}} \boldsymbol{c}_n^T \nu_n(\boldsymbol{x})\Big|_{\boldsymbol{x}=\boldsymbol{y}_i} = \frac{\partial}{\partial \boldsymbol{x}} \sum_\ell \alpha_\ell \boldsymbol{c}_{n\ell}^T \nu_n(\boldsymbol{x})\Big|_{\boldsymbol{x}=\boldsymbol{y}_i} = \sum_\ell \alpha_\ell \boldsymbol{b}_{i\ell}, \quad (8)$$

where $\boldsymbol{b}_{i\ell} \in S_i^\perp$ is a normal vector to subspace S_i. Therefore, although each polynomial in I_n may not be factorizable, its derivative at $\boldsymbol{y}_i$ still gives a vector normal to S_i. This fact should come at no surprise. The zero set of each polynomial $p_{n\ell}$ is just a surface in $\mathbb{R}^K$, therefore its derivative at point $\boldsymbol{y}_i \in S_i$, $Dp_{n\ell}(\boldsymbol{y}_i)$, gives a vector normal to the surface. Since a mixture of subspaces is locally flat, i.e. in a neighborhood of $\boldsymbol{y}_i$ the surface is merely the subspace S_i, then the derivative at $\boldsymbol{y}_i$ lives in the orthogonal complement $S_i^\perp$ of S_i. Indeed, as stated by the following theorem, the span of all the derivatives equals $S_i^\perp$, hence we can also obtain the dimension of each subspace $k_i = \dim(S_i)$.

Theorem 2 (Subspace bases by polynomial differentiation) *If the set $\boldsymbol{X}$ is such that $\dim(\text{null}(L_n)) = \dim(I_n) = m_n$, and one generic point $\boldsymbol{y}_i$ is given for each subspace S_i, then we have*

$$S_i^\perp = \text{span}\Big\{ \frac{\partial}{\partial \boldsymbol{x}} \boldsymbol{c}_n^T \nu_n(\boldsymbol{x})\Big|_{\boldsymbol{x}=\boldsymbol{y}_i},\ \forall \boldsymbol{c}_n \in \text{null}(L_n) \Big\}. \quad (9)$$

Therefore, the dimensions of the subspaces are given by

$$k_i = K - \text{rank}\big(DP_n(\boldsymbol{y}_i)\big) \quad i = 1, \ldots, n, \quad (10)$$

where $DP_n(\boldsymbol{x}) \doteq [Dp_{11}(\boldsymbol{x}), \ldots, Dp_{1m_n}(\boldsymbol{x})] \in \mathbb{R}^{K \times m_n}$.

As a consequence of Theorem 2, we already have the sketch of an algorithm for clustering subspaces of arbitrary dimensions in a semi-supervised learning scenario in which we are given *one example per class* $\{\boldsymbol{y}_i \in S_i\}_{i=1}^n$:

1. Compute a basis for the null space null(L_n) using, for example, SVD.
2. Evaluate the derivative of the (possibly nonfactorizable) polynomial $\boldsymbol{c}_n^T \nu_n(\boldsymbol{x})$ at $\boldsymbol{y}_i$ for each $\boldsymbol{c}_n$ in the basis of null(L_n) to obtain a set of normal vectors in $S_i^\perp$.
3. Compute a basis for $S_i^\perp$ by applying PCA to the normal vectors obtained in step 2. PCA automatically gives the dimension of each subspace $k_i = \dim(S_i)$.
4. Cluster the data points by assigning point $\boldsymbol{x}^j$ to subspace i if
$$i = \arg \min_{\ell=1,\ldots,n} \|B_\ell^T \boldsymbol{x}^j\|. \quad (11)$$

Remark 2 (Estimating the bases from noisy data points) *Since we are using PCA, with a moderate level of noise we can still obtain a basis for each subspace and cluster the data as above, because the coefficients of the polynomials and their derivatives depend continuously on the data. In fact, one can compute the derivatives of $p_n(\boldsymbol{x}) = \boldsymbol{c}_n^T \nu_n(\boldsymbol{x})$ algebraically. For instance, one may compute $Dp_n(\boldsymbol{x}) = \boldsymbol{c}^T D\nu_n(\boldsymbol{x}) = \boldsymbol{c}^T E_n \nu_{n-1}(\boldsymbol{x})$, with $E_n \in \mathbb{R}^{M_n \times M_{n-1}}$ a constant matrix containing the exponents of the Veronese map $\nu_n(\boldsymbol{x})$. Notice that we can also obtain the dimension of each subspace by looking at the singular values of the matrix of derivatives, similarly to (5).*

3.3. Obtaining one point per subspace

Theorem 2 demonstrates that one can obtain a basis for each $S_i^\perp$ directly from the derivatives of the polynomials representing the union of the subspaces. However, in order to proceed we need to have one point per subspace, i.e. we need to know the vectors $\{\boldsymbol{y}_i\}_{i=1}^n$. We now consider the *unsupervised learning scenario* in which we do not know the label for any of the data points.

The idea is that we can always choose a point $\boldsymbol{y}_n$ lying on one of the subspaces, say S_n, by checking that $P_n(\boldsymbol{y}_n) = 0$. Since we are given a set of data points $\boldsymbol{X} = \{\boldsymbol{x}^j\}_{j=1}^n$ lying on the subspaces, in principle we can choose $\boldsymbol{y}_n$ to be any of the data points. However, in the presence of noise and outliers, a random choice of $\boldsymbol{y}_n$ may be far from the true subspaces. One may be tempted to choose a point in the data set $\boldsymbol{X}$ that minimizes $\|P_n(\boldsymbol{x})\|$, as we did in Section 2. However, such a choice has the following problems:

1. The value $\|P_n(\boldsymbol{x})\|$ is merely an *algebraic* error, i.e. it does not really represent the *geometric* distance from $\boldsymbol{x}$ to its closest subspace. In principle, finding the geometric distance from $\boldsymbol{x}$ to its closest subspace is a hard problem, because we do not know the bases $\{B_i\}_{i=1}^n$.
2. Points $\boldsymbol{x}$ lying close to the intersection of two or more subspaces are more likely to be chosen, because two or more factors in $p_n(\boldsymbol{x}) = (\boldsymbol{b}_1^T \boldsymbol{x}) \cdots (\boldsymbol{b}_n^T \boldsymbol{x})$ are approximately zero, which yields a smaller value for $|p_n(\boldsymbol{x})|$. Since $Dp_n(\boldsymbol{x}) = 0$ for $\boldsymbol{x}$ in the intersection of two or more subspaces, one should avoid choosing such points, because they typically give very noisy estimates of the normal vectors. In fact, we can see from (6) and (8) that for an arbitrary $\boldsymbol{x}$ the vector $Dp_n(\boldsymbol{x})$ is a linear combination of the normal vectors $\{\boldsymbol{b}_i\}_{i=1}^n$. Thus if $\boldsymbol{x}$ is close to two subspaces, $Dp_n(\boldsymbol{x})$ will be a linear combination of both normals.

As it turns out, one can avoid both of these problems thanks to the following lemma.

Lemma 1 *Let $\tilde{\boldsymbol{x}}$ be the projection of $\boldsymbol{x} \in \mathbb{R}^K$ onto its closest subspace. The Euclidean distance from $\boldsymbol{x}$ to $\tilde{\boldsymbol{x}}$ is*

$$\|\boldsymbol{x} - \tilde{\boldsymbol{x}}\| = \sqrt{P_n(\boldsymbol{x}) \big(DP_n(\boldsymbol{x})^T DP_n(\boldsymbol{x})\big)^\dagger P_n(\boldsymbol{x})^T} + O\big(\|\boldsymbol{x} - \tilde{\boldsymbol{x}}\|^2\big),$$

where $P_n(\boldsymbol{x}) = [p_{n1}(\boldsymbol{x}), \ldots, p_{nm_n}(\boldsymbol{x})] \in \mathbb{R}^{1 \times m_n}$, $DP_n(\boldsymbol{x}) = [Dp_{n1}(\boldsymbol{x}), \ldots, Dp_{nm_n}(\boldsymbol{x})] \in \mathbb{R}^{K \times m_n}$, and $A^\dagger$ is the Moore-Penrose inverse of A.

Thanks to Lemma 1, we can immediately choose a point $\boldsymbol{y}_n$ lying on (close to) one of the subspaces and not in (far from) the other subspaces as

$$\boldsymbol{y}_n = \underset{\boldsymbol{x} \in \boldsymbol{X} : DP_n(\boldsymbol{x}) \neq 0}{\arg\min} P_n(\boldsymbol{x}) \big(DP_n(\boldsymbol{x})^T DP_n(\boldsymbol{x}) \big)^\dagger P_n(\boldsymbol{x})^T, \quad (12)$$

and then compute the basis $B_n \in \mathbb{R}^{K \times (K-k_n)}$ for $S_n^\perp$ by applying PCA to $DP_n(\boldsymbol{y}_n)$.

In order to find a point $\boldsymbol{y}_{n-1}$ lying on (close to) one of the remaining $(n-1)$ subspaces but not in (far from) S_n, we could in principle choose $\boldsymbol{y}_{n-1}$ as in (12) after removing the points in S_n from the data set $\boldsymbol{X}$. With noisy data, however, this depends on a threshold and is not very robust. Alternatively, we can find a new set of polynomials $\{p_{(n-1)\ell}(\boldsymbol{x})\}$ defining the algebraic set $\cup_{i=1}^{n-1} S_i$. In the case of hyperplanes, there is only one such polynomial, namely

$$p_{n-1}(\boldsymbol{x}) \doteq (\boldsymbol{b}_1 \boldsymbol{x}) \cdots (\boldsymbol{b}_{n-1}^T \boldsymbol{x}) = \frac{p_n(\boldsymbol{x})}{\boldsymbol{b}_n^T \boldsymbol{x}} = \boldsymbol{c}_{n-1}^T \nu_{n-1}(\boldsymbol{x}).$$

Therefore, we can obtain $p_{n-1}(\boldsymbol{x})$ by *polynomial division*. Notice that dividing $p_n(\boldsymbol{x})$ by $\boldsymbol{b}_n^T \boldsymbol{x}$ is a *linear problem* of the form $R_n(\boldsymbol{b}_n)\boldsymbol{c}_{n-1} = \boldsymbol{c}_n$, where $R_n(\boldsymbol{b}_n) \in \mathbb{R}^{M_n \times M_{n-1}}$. This is because solving for the coefficients of $p_{n-1}(\boldsymbol{x})$ is equivalent to solving the equations $(\boldsymbol{b}_n^T \boldsymbol{x})(\boldsymbol{c}_{n-1}^T \nu_n(\boldsymbol{x})) = \boldsymbol{c}_n^T \nu_n(\boldsymbol{x})$, where $\boldsymbol{b}_n$ and $\boldsymbol{c}_n$ are already known.

Example 1 *If $n = 2$ and $\boldsymbol{b}_2 = [b_1, b_2, b_3]^T$, then the matrix $R_2(\boldsymbol{b}_2)$ is given by*

$$R_2(\boldsymbol{b}_2) = \begin{bmatrix} b_1 & b_2 & b_3 & 0 & 0 & 0 \\ 0 & b_1 & 0 & b_2 & b_3 & 0 \\ 0 & 0 & b_1 & 0 & b_2 & b_3 \end{bmatrix}^T \in \mathbb{R}^{6 \times 3}.$$

In the case of subspaces of arbitrary dimensions, in principle we cannot simply divide the polynomials P_n by $\boldsymbol{b}_n^T \boldsymbol{x}$ for any column $\boldsymbol{b}_n$ of B_n, because the polynomials $\{p_{n\ell}(\boldsymbol{x})\}$ may not be factorizable. Furthermore, they do not necessarily have the common factor $\boldsymbol{b}_n^T \boldsymbol{x}$. The following theorem resolves this difficulty by showing how to compute the polynomials associated with the remaining subspaces $\cup_{i=1}^{n-1} S_i$.

Theorem 3 (Obtaining points by polynomial division) *If the set $\boldsymbol{X}$ is such that $\dim(\text{null}(L_n)) = \dim(I_n)$, the set of homogeneous polynomials of degree $(n-1)$ associated with the algebraic set $\cup_{i=1}^{n-1} S_i$ is given by $\{\boldsymbol{c}_{n-1}^T v_{n-1}(\boldsymbol{x})\}$ where the vectors of coefficients $\boldsymbol{c}_{n-1} \in \mathbb{R}^{M_{n-1}}$ must satisfy*

$$L_n R_n(\boldsymbol{b}_n)\boldsymbol{c}_{n-1} = 0, \quad \textit{for all} \quad \boldsymbol{b}_n \in S_n^\perp. \quad (13)$$

Thanks to Theorem 3, we can obtain a collection of polynomials $\{p_{(n-1)\ell}(\boldsymbol{x})\}_{\ell=1}^{m_{n-1}}$ representing $\cup_{i=1}^{n-1} S_i$ from the null space of $L_n R_n(\boldsymbol{b}_n) \in \mathbb{R}^{N \times M_{n-1}}$. We can then repeat the same procedure to find a basis for the remaining subspaces. We thus obtain the following *polynomial differentiation algorithm* (PDA) for finding a basis for a collection of subspaces of unknown and possibly different dimensions from sample data points $\{\boldsymbol{x}^j\}_{j=1}^N$.

Polynomial Differentiation Algorithm (PDA)

set $L_n = [\nu_n(\boldsymbol{x}^1), \ldots, \nu_n(\boldsymbol{x}^N)]^T \in \mathbb{R}^{N \times M_n}$;
for $i = n : 1$ **do**
  solve $L_i \boldsymbol{c} = 0$ to obtain a basis $\{\boldsymbol{c}_{i\ell}\}_{\ell=1}^{m_i}$ of null(L_i), where the number of polynomials m_i is obtained as in (5);
  set $P_i(\boldsymbol{x}) = [p_{i1}(\boldsymbol{x}), \ldots, p_{im_i}(\boldsymbol{x})] \in \mathbb{R}^{1 \times m_i}$, where $p_{i\ell}(\boldsymbol{x}) = \boldsymbol{c}_{i\ell}^T \nu_i(\boldsymbol{x})$ for $\ell = 1, \ldots, m_i$;
  do
    $\boldsymbol{y}_i = \underset{\boldsymbol{x} \in \boldsymbol{X} : DP_i(\boldsymbol{x}) \neq 0}{\arg\min} P_i(\boldsymbol{x}) \big(DP_i(\boldsymbol{x})^T DP_i(\boldsymbol{x}) \big)^\dagger P_i(\boldsymbol{x})^T$,
    $B_i = PCA\big(DP_i(\boldsymbol{y}_i)\big)$,
    $L_{i-1} = L_i \Big[R_i^T(\boldsymbol{b}_{i1}) \cdots R_i^T(\boldsymbol{b}_{i,K-k_i}) \Big]^T$, with $\boldsymbol{b}_{ij}$ columns of B_i;
  end do
end for
for $j = 1 : N$ **do**
  assign point $\boldsymbol{x}^j$ to subspace S_i if $i = \arg\min_\ell \|B_\ell^T \boldsymbol{x}^j\|$;
end for

Remark 3 (Avoiding polynomial division) *Notice that one may avoid computing P_i for $i < n$ by using a heuristic distance function to choose the points $\{\boldsymbol{y}_i\}_{i=1}^n$ as follows. Since a point in $\cup_{\ell=i}^n S_\ell$ must satisfy $\|B_i^T \boldsymbol{x}\| \cdots \|B_n^T \boldsymbol{x}\| = 0$, we can choose a point $\boldsymbol{y}_{i-1}$ on $\cup_{\ell=1}^{i-1} S_\ell$ as:*

$$\boldsymbol{y}_{i-1} = \underset{\boldsymbol{x} \in \boldsymbol{X} : DP_n(\boldsymbol{x}) \neq 0}{\arg\min} \frac{\sqrt{P_n(\boldsymbol{x}) (DP_n(\boldsymbol{x})^T DP_n(\boldsymbol{x}))^\dagger P_n(\boldsymbol{x})^T} + \delta}{\|B_i^T \boldsymbol{x}\| \cdots \|B_n^T \boldsymbol{x}\| + \delta},$$

where $\delta > 0$ is chosen to avoid cases in which both the numerator and the denominator are zero (e.g., with perfect data).

4. Further analysis of the algorithm

4.1. Projection and minimum representation

In practice, when the dimension of the ambient space K is large, the complexity of the above polynomial-based algorithms becomes prohibitive.[2] If the dimension of the individual subspaces k_i is relatively small, it seems rather redundant to use $\mathbb{R}^K$ to represent such low-dimensional linear structure. In such cases, one may reduce the dimensionality by projecting the data onto a lower-dimensional (sub)space.

We will distinguish between two different kinds of "projections." The first kind is for the case in which the span of all the subspaces is a proper subspace of the ambient space, i.e. span$(\cup_{i=1}^n S_i) \subset \mathbb{R}^K$. In this case, one may simply run the classic PCA algorithm to eliminate the redundant dimensions. The second kind is for the case in which the largest dimension of the subspaces, denoted by k_{max}, is strictly less than $K-1$. When $k_{\max}$ is known,[3] one may choose a $(k_{max}+1)$-dimensional (affine) subspace $\mathcal{P}$ such that, by projecting $\mathbb{R}^K$ onto this subspace:

$$\pi_{\mathcal{P}} : \boldsymbol{x} \in \mathbb{R}^K \quad \mapsto \quad \boldsymbol{x}' = \pi_{\mathcal{P}}(\boldsymbol{x}) \in \mathcal{P},$$

[2] This is because M_n is of the order n^K.

[3] For example, in 3-D motion segmentation from affine cameras, it is known that the subspaces have dimension at most four [5, 10].

the dimension of each original subspace S_i is preserved,[4] and there is a one-to-one correspondence between S_i and its projection – no reduction in the number of subspaces n.[5] Although all $(k_{max}+1)$-dimensional affine subspaces of $\mathbb{R}^K$ that have the above properties form an open and dense set, it remains unclear what a "good" choice is for such a subspace when there is noise in the data. One possibility is to randomly select a few projections and choose the one that results in the smallest fitting error. Another possibility is to apply classic PCA to project onto a $(k_{max}+1)$-dimensional affine subspace. In either case, the resulting $\mathcal{P}$ gives a "minimum" representation that preserves the multilinear structure of the original data.[6]

Therefore, given a sample data set $\boldsymbol{X}$ embedded in a high-dimensional space, we can simultaneously identify and represent its multilinear structure in a space with the lowest possible dimension through the following steps:

$$\boldsymbol{X} \xrightarrow{PCA(\pi_{\mathcal{P}})} \boldsymbol{X}' \xrightarrow{PDA} \cup_{i=1}^{n} \pi_{\mathcal{P}}(S_i) \xrightarrow{\pi_{\mathcal{P}}^{-1}} \cup_{i=1}^{n} S_i.$$

4.2. Robustness and outlier rejection

In practice, there could be points in $\boldsymbol{X}$ that are far away from any of the subspaces. Such sample points are called outliers. By detecting and rejecting those samples we can typically ensure a much better estimate for the subspaces. Many methods can be deployed to detect and reject the outliers. For instance, the function

$$d^2(\boldsymbol{x}) = P_n(\boldsymbol{x})\big(DP_n(\boldsymbol{x})^T DP_n(\boldsymbol{x})\big)^{\dagger} P_n(\boldsymbol{x})^T$$

approximates the squared distance of a point $\boldsymbol{x}$ to the subspaces. From the d^2-histogram of the sample set $\boldsymbol{X}$, we may exclude from $\boldsymbol{X}$ all points that have unusually large d^2 values and use only the remaining sample points to *rerun* the PDA algorithm. For instance, if we assume that the sample points are drawn around each subspace from independent Gaussian distributions with a small variance σ^2, $\frac{d^2}{\sigma^2}$ is approximately a χ^2-distribution with $\sum_i (K-k_i)$ degrees of freedom. We can apply standard χ^2-test to reject samples which deviate significantly from this distribution.

4.3. Combining PDA with spectral clustering

Although subspace clustering can be viewed as a special clustering problem, many of the classical clustering algorithms such as spectral clustering [11] cannot be directly applied. This is because, in order for spectral clustering algorithms to work well, one would need to define a distance between any pair of points in the subspaces that depends only on the geometry of the subspaces but not on the locations of the points inside the subspaces. The Euclidean distance between sample points in the sample set $\boldsymbol{X}$ does not have this property. However, Theorem 2 enables us to associate every point $\boldsymbol{x}_i$ in S_i with a basis B_i for $S_i^{\perp}$. A distance function between points $\boldsymbol{x}_i$ in S_i and $\boldsymbol{x}_j$ in S_j can be defined between the two bases:

$$\mathcal{D}_{ij} \doteq \langle B_i, B_j \rangle, \tag{14}$$

where $\langle \cdot, \cdot \rangle$ denotes the largest subspace angle between the two subspaces. Notice that this distance does not depend on the particular location of the point in each subspace. Based on this distance function, one can define an $N \times N$ similarity matrix, e.g., $\mathcal{S}_{ij} = \exp(-\mathcal{D}_{ij}^2)$, for the N samples in $\boldsymbol{X}$. This allows one to apply classical spectral clustering algorithms to the subspace clustering problem.

5. Experiments on synthetic data

We first evaluate the performance of PDA by comparing it with PFA, K-subspace and EM on synthetically generated data. The experimental setup consists of choosing $n = 2, 3, 4$ collections of $N = 200n$ points on randomly chosen planes in $\mathbb{R}^3$. Zero-mean Gaussian noise with s.t.d. from 0% to 5% is added to the sample points. We run 1000 trials for each noise level. For each trial the error between the true (unit) normals $\{\boldsymbol{b}_i\}_{i=1}^n$ and the estimates $\{\hat{\boldsymbol{b}}_i\}_{i=1}^n$ is computed as

$$\text{error} = \frac{1}{n}\sum_{i=1}^{n} \text{acos}\left(\boldsymbol{b}_i^T \hat{\boldsymbol{b}}_i\right) \text{(degrees)}. \tag{15}$$

Error vs. noise. Figure 2 (top) plots the mean error as a function of noise for $n = 4$. Similar results were obtained for $n = 2, 3$, though with smaller errors. Notice that the estimates of PDA with $\delta = 0.02$ have an error of about 50% compared to PFA. This is because PDA deals automatically with noisy data and outliers by choosing the points $\{\boldsymbol{y}_i\}_{i=1}^n$ in an optimal fashion. The choice of δ was not important (results were similar for $\delta \in [0.001, 0.1]$), as long as it is a small number. Notice also that both K-subspace and EM have a nonzero error in the noiseless case, showing that they frequently converge to a local minima when a single randomly chosen initialization is used. When initialized with PDA, both K-subspace and EM reduce the error to approximately 35-50% with respect to random initialization.

Error vs. number of subspaces. Figure 2 (bottom) plots the estimation error of PDA as a function of the number of subspaces n, for different levels of noise. As expected, the error increases as a function of n.

Computing time. Table 1 shows the mean computing time and the mean number of iterations for a MATLAB implementation of each one of the algorithms over 1000 trials. Among the algebraic algorithms, the fastest one is PFA which directly factors $p_n(\boldsymbol{x})$ given $\boldsymbol{c}_n$. The extra cost of PDA relative to PFA is to compute the derivatives $Dp_n(\boldsymbol{x})$ for all $\boldsymbol{x} \in \boldsymbol{X}$ and to divide the polynomials. Overall, PDA gives half of the error of PFA in about twice as much time. Notice also that PDA reduces the number of iterations of

[4] This requires that $\mathcal{P}$ be transversal to each $S_i^{\perp}$, i.e. span$\{\mathcal{P}, S_i^{\perp}\} = \mathbb{R}^K$ for every $i = 1, \ldots, n$. Since n is finite, this transversality condition can be easily satisfied. Furthermore, the set of positions for $\mathcal{P}$ which violate the transversality condition is only a zero-measure closed set [2].

[5] This requires that all S_i' be transversal to each other in $\mathcal{P}$, which is guaranteed if we require $\mathcal{P}$ to be transversal to $S_i^{\perp} \cap S_j^{\perp}$ for $i, j = 1, .., n$. All $\mathcal{P}$'s which violate this condition form again only a zero-measure set.

[6] The effect of the projection $\pi_{\mathcal{P}}$ can also be explained from an algebraic viewpoint. In the case of hyperplanes i.e. $k = K - 1$, the ideal I associated with the algebraic set $\cup_{i=1}^n S_i$ becomes a principle ideal generated by a unique homogeneous polynomial with the lowest possible degree.

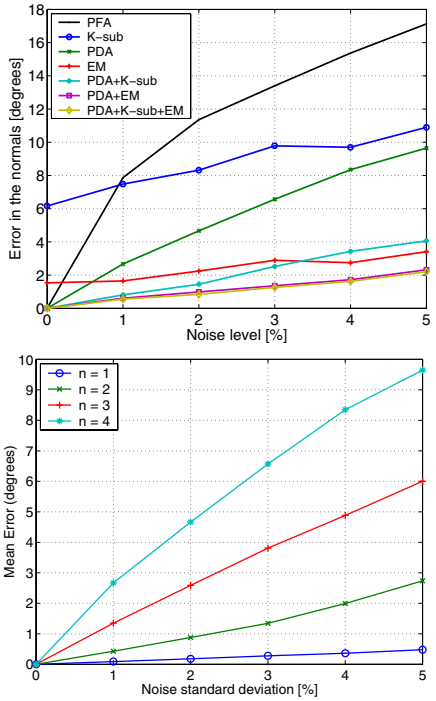

Figure 2: Error vs. noise for data lying on two-dimensional subspaces of $\mathbb{R}^3$. Top: PFA, PDA ($\delta = 0.02$), K-subspace and EM randomly initialized, K-subspace and EM initialized with PDA, and EM initialized with K-subspace initialized with PDA for $n = 4$ subspaces. Bottom: PDA for $n = 1, \ldots, 4$ subspaces.

K-subspace and EM to approximately 1/3 and 1/2, respectively. The computing times for K-subspace and EM are also reduced including the extra time spent on initialization with PDA or PDA+K-subspace.

Table 1: Mean computing time and mean number of iterations for each one of the algorithms.

Algorithms	$L_n\mathbf{c} = 0$	PFA	PDA	K-sub
Time (sec.)	0.0854	0.1025	0.1818	0.4637
# Iterations	none	none	none	19.7

Algorithms	PDA +K-sub	EM	PDA+EM	PDA+K-sub+EM
Time (sec.)	0.2525	1.0408	0.6636	0.7528
# Iterations	7.1	30.8	17.1	15.0

6. Applications in computer vision

In this section, we apply GPCA to problems in computer vision such as vanishing point detection, face clustering, and news video segmentation. We refer the reader to [9], [4] and [10] for applications in 2-D and 3-D motion segmentation from two, three and multiple views, respectively.

Detection of vanishing points. Given a collection of parallel lines in 3-D, it is well know that their perspective projections intersect at the so-called *vanishing point*, which is located either in the image or at infinity. Let $\{\boldsymbol{\ell}_j \in \mathbb{P}^2\}_{j=1}^N$ be the projection of n sets of parallel lines intersecting at the vanishing points $\{\boldsymbol{v}_i \in \mathbb{P}^2\}_{i=1}^n$. Since for each line j there exists a point i such that $\boldsymbol{v}_i^T \boldsymbol{\ell}_j = 0$, the problem of estimating the n vanishing points from the set of N lines without knowing which subsets of lines intersect in the same point, is equivalent to estimating a collection of n planes in $\mathbb{R}^3$ with normal vectors $\{\boldsymbol{v}_i \in \mathbb{P}^2\}_{i=1}^n$ from sample data points $\{\boldsymbol{\ell}_j \in \mathbb{P}^2\}_{j=1}^N$. Figure 3 shows an example from the Corel Database with $n = 3$ sets of $N = 30$ parallel lines, which were manually extracted. For each one of the three set of lines we computed their intersecting point (assuming known segmentation) and regarded those intersection points as ground truth data. We then applied PDA to the set of lines assuming unknown segmentation and used the resulting vanishing points to initialize K-subspace. Figure 3 (top-right) shows the segmentation of the lines obtained by PDA. Only one line is misclassified, the top horizontal line, because it approximately passes through two vanishing points. Figure 3 (bottom) shows the vanishing points estimated by PDA and PDA+K-subspace, and compares them with the ground truth. The error in the estimation of the vanishing points with respect to the ground truth are $1.7°$, $11.1°$ and $1.5°$ for PDA and $0.4°$, $2.0°$, and $0.0°$ for PDA+K-sub.

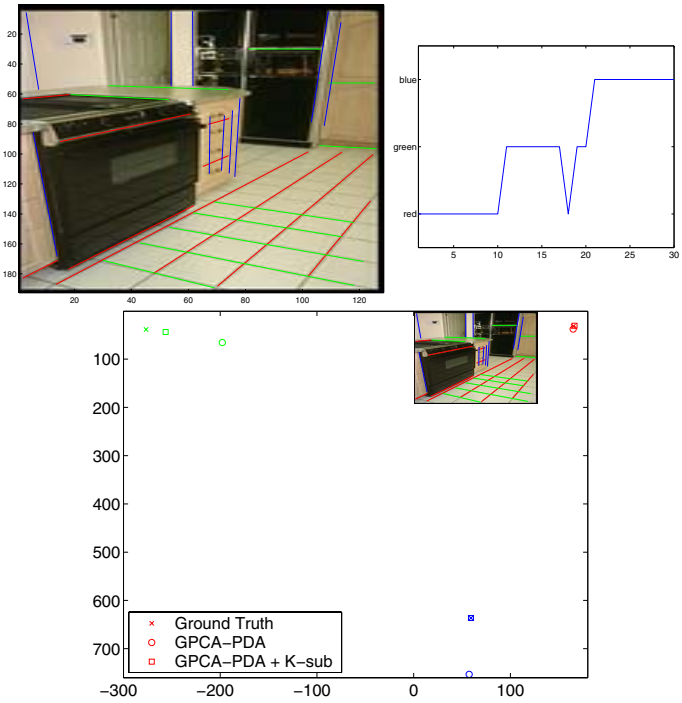

Figure 3: Top-left: Image #364081 from the Corel database with 3 sets of 10 parallel lines superimposed. Top-right: Segmentation of the 30 lines given by PDA. Bottom: Vanishing points estimated by PDA and PDA+K-subspace and the true vanishing points.

Face clustering under varying illumination. Given a collection of unlabeled images $\{I_j \in \mathbb{R}^K\}_{j=1}^N$ of n different faces taken under varying illumination, we would like to cluster the images corresponding to the same person. For a Lambertian object, it has been shown that the set of all images taken under all lighting conditions forms a cone in the image space, which can be well approximated by a low-

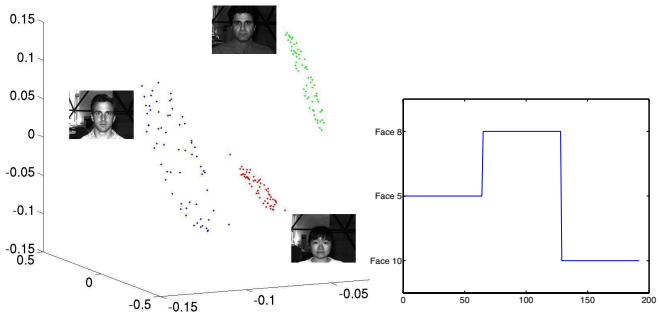

Figure 4: Clustering a subset of the Yale Face Database B consisting of 64 frontal views under varying lighting conditions for subjects 2, 5 and 8. Left: Image data projected onto the three principal components. Right: Clustering of the images given by PDA.

dimensional subspace [3]. Therefore, we can cluster the collection of images by estimating a basis for each one of those subspaces, because the images of different faces will live in different subspaces. Since in practice the number of pixels K is large compared with the dimension of the subspaces, we first apply PCA to project the images onto $\mathbb{R}^{K'}$ with $K' << K$. More specifically, we compute the SVD of the data $[I_1\ I_2 \cdots I_N]_{K\times N} = USV^T$ and consider a matrix $X \in \mathbb{R}^{K'\times N}$ consisting of the first K' columns of V. We obtain a new set of data points in $\mathbb{R}^{K'}$ from each one of the rows of X. We use homogeneous coordinates $\{\boldsymbol{x}_j \in \mathbb{R}^{K'+1}\}_{j=1}^N$ so that each subspace goes through the origin. The new data set also lives in a collection of subspaces, because it is the projection of the original set of subspaces onto a lower-dimensional linear space. We consider a subset of the Yale Face Database B consisting of $N = 64n$ frontal views of $n = 3$ faces (subjects 5, 8 and 10) under 64 varying lighting conditions. For computational efficiency, we downsampled each image to $K = 30 \times 40$ pixels. Then we projected the data onto the first $K' = 3$ principal components, as shown in Figure 4 (left). We applied PDA to the data in homogeneous coordinates and fitted three subspaces of dimension 3, 2, and 2. PDA obtained perfect a segmentation as shown in Figure 4 (right).

Segmentation of news video sequences. Consider a news video sequence in which the camera is switching among a small number of scenes. For instance, the host could be interviewing a guest and the camera may be switching between the host, the guest and both of them, as shown in Figure 5. Given the frames $\{I_j \in \mathbb{R}^K\}_{j=1}^N$, we would like to cluster them according to the different scenes. We assume that all the frames corresponding to the same scene live in a low-dimensional subspace of $\mathbb{R}^K$ and that different scenes correspond to different subspaces. As in the case of face clustering, we may segment the video sequence into different scenes by applying PDA to the image data projected onto the first few principal components. Figure 5 shows the segmentation results for two video sequences. In both cases, a perfect segmentation is obtained.

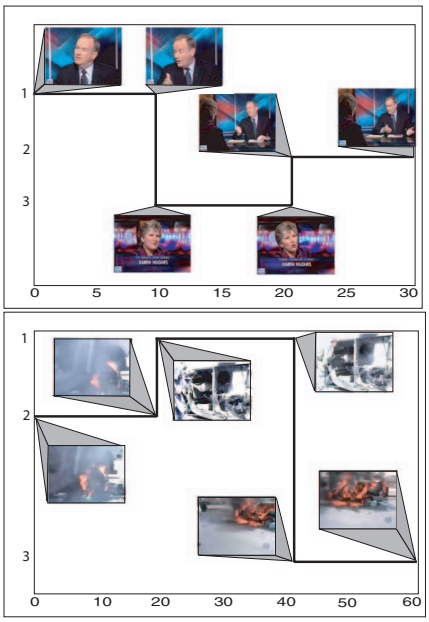

Figure 5: Clustering frames of a news video into groups of scenes. Top: 30 frames clustered into 3 groups: host, guest and both. Bottom: 60 frames clustered into 3 groups: rear of a car with a burning wheel, a burnt car with people and a burning car.

7 Conclusions

We have presented a new algebraic algorithm for subspace clustering. Our algorithm is based on estimating a collection of polynomials from data and then evaluating their derivatives at points on the subspaces. We showed that our algorithm gives about half of the error w.r.t extant algebraic algorithms based on polynomial factorization, and significantly improves the performance of iterative techniques such as K-subspace and EM. We also demonstrated the performance of the new algorithm on the estimation of vanishing points, face clustering, and news video segmentation.

References

[1] J. Costeira and T. Kanade. A multibody factorization method for independently moving objects. *IJCV*, 29(3):159–179, 1998.

[2] M. Hirsch. *Differential Topology*. Springer, 1976.

[3] J. Ho, M.-H. Yang, J. Lim, K.-C. Lee, and D. Kriegman. Clustering apperances of objects under varying illumination conditions. In *CVPR*, pages 11-18, 2003.

[4] R. Hartley and R. Vidal. The multibody trifocal tensor: Motion segmentation from 3 perspective views. In *CVPR*, 2004.

[5] K. Kanatani. Motion segmentation by subspace separation and model selection. In *ICCV* , volume 2, pages 586–591, 2001.

[6] A. Leonardis, H. Bischof, and J. Maver. Multiple eigenspaces. *Pattern Recognition*, 35(11):2613–2627, 2002.

[7] M. Tipping and C. Bishop. Mixtures of probabilistic principal component analyzers. *Neural Computation*, 11(2), 1999.

[8] R. Vidal, Y. Ma, and S. Sastry. Generalized principal component analysis (GPCA). In *CVPR*, volume 1, pages 621–628, 2003.

[9] R. Vidal and Y. Ma. A unified algebraic approach to 2-D and 3-D motion segmentation. In *ECCV*, 2004.

[10] R. Vidal and R. Hartley. Motion segmentation with missing data using PowerFactorization and GPCA. In *CVPR*, 2004.

[11] Y. Weiss. Segmentation using eigenvectors: a unifying view. In *IEEE International Conference on Computer Vision*, pages 975–982, 1999.

[12] Y. Wu, Z. Zhang, T.S. Huang and J.Y. Lin. Multibody grouping via orthogonal subspace decomposition. In *CVPR*, volume 2, pages 252–257, 2001.