GENERALIZED PRINCIPAL COMPONENT ANALYSIS (GPCA): SUBSPACE CLUSTERING BY POLYNOMIAL FACTORIZATION, DIFFERENTIATION AND DIVISION

by

René Vidal, Yi Ma and Shankar Sastry

Memorandum No. UCB/ERL M03/36

15 August 2003

Cover

GENERALIZED PRINCIPAL COMPONENT ANALYSIS (GPCA): SUBSPACE CLUSTERING BY POLYNOMIAL FACTORIZATION, DIFFERENTIATION AND DIVISION

by

René Vidal, Yi Ma and Shankar Sastry

Memorandum No. UCB/ERL M03/36

15 August 2003

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

Generalized Principal Component Analysis (GPCA): Subspace Clustering by Polynomial Factorization, Differentiation and Division*

René Vidal† Yi Ma‡ Shankar Sastry†

†Department of EECS
University of California at Berkeley
301 Cory Hall, Berkeley, CA 94720

‡Computer Engineering Department
University of Illinois at Urbana-Champaign
Urbana, IL 61801

August 15, 2003

Abstract

We consider the so-called *Generalized Principal Component Analysis* (GPCA) problem, i.e., the problem of identifying n linear subspaces of a K-dimensional linear space from a collection of sample points drawn from these subspaces. In the absence of noise, we cast GPCA in an algebraic geometric framework in which the number of subspaces n becomes the degree of a certain polynomial and the subspace parameters become the factors (roots) of such a polynomial. In the presence of noise, we cast GPCA as a constrained nonlinear least squares problem which minimizes the error between the noisy points and their projections subject to all mixture constraints. By converting this constrained problem into an unconstrained one, we obtain an optimal function from which the subspaces can be directly recovered using standard non-linear optimization techniques.

In the case of subspaces of dimension $k = K - 1$, i.e., hyperplanes, we show that the number of hyperplanes n can be obtained from the rank of a certain matrix that depends on the data. Given n, the estimation of the hyperplanes is essentially equivalent to a factorization problem in the space of homogeneous polynomials of degree n in K variables. After proving that such a problem admits a unique solution, we propose two algorithms for estimating the hyperplanes. The *polynomial factorization algorithm* (PFA) obtains a basis for each hyperplane from the roots of a polynomial of degree n in one variable and from the solution of $K - 2$ linear systems in n variables. This shows that the GPCA problem has a closed form solution when $n \leq 4$. The *polynomial differentiation algorithm* (PDA) obtains a basis for each hyperplane by evaluating the derivatives of the polynomial representing the hyperplanes at a collection of points in each one of the hyperplanes. We select those points either by intersecting the hyperplanes with a randomly chosen line, or by else by choosing points in the dataset that minimize a certain distance function.

In the case of subspaces of equal dimension $k_1 = \cdots = k_n = k < K - 1$, we first derive rank constraints on the data from which one can estimate the number of subspaces n and their dimension k. Given n and k, we show that the estimation of the subspaces can be reduced to the estimation of hyperplanes of dimension $k = K' - 1$ which are obtained by first projecting the data onto a K'-dimensional subspace of $\mathbb{R}^K$. Therefore, the estimation of the subspaces can be done using either the polynomial factorization or the polynomial differentiation algorithm for hyperplanes.

In the case of subspaces of arbitrary dimensions, $1 \leq k_1, \ldots, k_n \leq K-1$, we show that the union of all subspaces can be represented by a collection of homogeneous polynomials of degree n is K variables, whose coefficients can be estimated linearly from data. Given such polynomials, we show that one can obtain vectors normal to each one of the subspaces by evaluating the derivatives of such polynomials at a collection of points in each one of the subspaces. The estimation of the dimension and of a basis for (the complement of) each subspace is then equivalent to applying standard PCA to the set of normal vectors. The above algorithm is in essence a generalization of the polynomial differentiation algorithm to subspaces of arbitrary dimensions.

Our theory can be applied to a variety of estimation problems in which the data comes simultaneously from multiple (approximately) linear models. Our experiments on low-dimensional data show that PDA gives about half of the error of the PFA and improves the performance of iterative techniques, such as K-subspace and EM, by about 50% with respect to random initialization. We also present applications of our algorithm on computer vision problems such as vanishing point detection, 2-D and 3-D motion segmentation, and face clustering under varying illumination.

*Research supported by ONR grant N00014-00-1-0621 and UIUC ECE Department startup fund.

1 Introduction

Principal Component Analysis (PCA) [11] refers to the problem of identifying a linear subspace $S \subset \mathbb{R}^K$ of unknown dimension $k < K$ from N sample points $x^j \in S, j = 1, 2, \ldots, N$. This problem shows up in a variety of applications in many fields, e.g., pattern recognition, data compression, image analysis, regression, etc., and can be solved in a remarkably simple way from the singular value decomposition (SVD) of the data matrix $[x^1, x^2, \ldots, x^N] \in \mathbb{R}^{K \times N}$. In the presence of noise, this purely algebraic solution has the geometric interpretation of minimizing the sum of the squared distances from the (noisy) data points x^j to their projections $\tilde{x}^j$ in S.

In addition to this algebraic-geometric interpretation, PCA can also be understood in a probabilistic manner. In Probabilistic PCA [20] (PPCA), the noises are assumed to be independent samples drawn from an unknown distribution, and the problem becomes one of identifying the subspace and the parameters of the distribution in a maximum likelihood sense. When the underlying noise distribution is Gaussian, the algebraic-geometric and probabilistic interpretations coincide [3]. However, when the underlying distribution is non Gaussian the solution to PPCA is no longer linear. For example, in [3] PCA is generalized to arbitrary distributions in the exponential family. The authors use Bregman distances to derive the log-likelihood as a nonlinear function of the natural parameter of the distribution. The log-likelihood is then minimized using standard nonlinear optimization techniques.

Another extension of PCA is nonlinear principal components (NLPCA) or Kernel PCA (KPCA), which is the problem of identifying a *nonlinear* manifold from sample data points. The standard solution to NLPCA [15] is based on first embedding the data into a higher-dimensional *feature* space F and then applying standard PCA to the embedded data. That is, one assumes that there exists an embedding of the data such that the embedded data points lie on a linear subspace of a higher-dimensional space. Since in practice the dimension of F can be large, a more practical solution is obtained from the eigenvalue decomposition of the so-called *kernel* matrix, hence the name KPCA. One of the disadvantages of KPCA is that it is unclear what kernel to use for a given problem, since the choice of the kernel naturally depends on the nonlinear structure of the manifold to be identified. In fact, learning kernels is an active topic of research in the KPCA community.

In this paper, we consider the following (alternative) extension of PCA to the case of mixtures of subspaces, which we call Generalized Principal Component Analysis (GPCA):

Problem 1 (Generalized Principal Component Analysis (GPCA))

Given a set of sample points $X = \{x^j \in \mathbb{R}^K\}_{j=1}^N$ drawn from $n > 1$ different linear subspaces $\{S_i \subseteq \mathbb{R}^K\}_{i=1}^n$ of dimension $k_i = \dim(S_i)$, $0 < k_i < K$, identify each subspace S_i without knowing which points belong to which subspace. By identifying the subspaces we mean the following:

1. Identify the number of subspaces n and their dimensions $\{k_i\}_{i=1}^n$;
2. Identify a basis (or a set of principal components) for each subspace S_i (or equivalently $S_i^\perp$);
3. Group or segment the given N data points into the subspace(s) to which they belong.

Figure 1 illustrates the case of $n = 3$ subspaces of $\mathbb{R}^3$ of dimensions $k_1 = k_2 = k_3 = 2$.

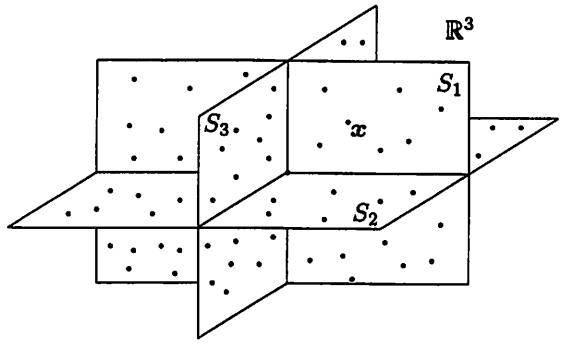


Figure 1: Three ($n = 3$) 2-dimensional subspaces S_1, S_2, S_3 in $\mathbb{R}^3$. The objective of GPCA is to identify all three subspaces from samples $\{x\}$ drawn from these subspaces.

1.1 Previous work on mixtures of principal components

Geometric approaches to mixtures of principal components have been proposed in the computer vision community on the context of 3-D motion segmentation. The main idea is to first segment the data associated with each subspace, and then apply standard PCA to each group. Kanatani [12] (see also [2, 4]) demonstrated that when the pairwise intersection of the subspaces is trivial, which implies that $K \geq nk$, one can use the SVD of all the data to build a similarity matrix from which the segmentation can be easily extracted. In the presence of noise the segmentation of the data becomes a quite challenging problem which can be solved using a time-consuming graph-theoretic approach as demonstrated in [4]. When the intersection of the subspaces in nontrivial, the segmentation of the data is usually done in an ad-hoc fashion using clustering algorithms such as K-means. The only existing geometric solution is for the case of two planes in $\mathbb{R}^3$ and was developed by Shizawa and Mase [17] in the context of 2-D segmentation of transparent motions.[1] To the best of our knowledge, our work is the first one to provide a geometric solution for an arbitrary number n of different subspaces of any dimensions $k_1, \cdots, k_n$ and with arbitrary intersections among them.

Probabilistic approaches to mixtures of principal components [19] assume that sample points within each subspace are drawn from an unknown probability distribution. The membership of the data points to each one of the subspaces is modeled with a multinomial distribution whose parameters are referred to as the mixing proportions. The parameters of this mixture model are estimated in a Maximum Likelihood or Maximum a Posteriori framework as follows: one first estimates the membership of the data given a current estimate of the model parameters, and then estimates the model parameters given a current estimate of the membership of the data. This is usually done in an iterative manner using the Expectation Maximization (EM) algorithm. However, the probabilistic approach to mixtures of principal components suffers from the following disadvantages:

1. It is hard to analyze some theoretical questions such as the existence and uniqueness of a solution to the problem.
2. It relies on a probabilistic model for the data, which is restricted to certain classes of distributions or independence assumptions.
3. The convergence of EM is in general very sensitive to initialization, hence there is no guarantee that it will converge to the optimal solution. To the best of our knowledge, there is no global initialization irrespective of the distribution of the data.
4. There are many cases in which it is very hard to solve the grouping problem correctly, and yet it is possible to obtain a quite precise estimate of the subspaces. In those cases, a direct estimation of the subspaces (without grouping) seems more appropriate than an estimation based on incorrectly segmented data.

One may therefore ask

1. *Is there an algebraic way of initializing statistical approaches to subspace segmentation?*
2. *Is is possible to find algebraic constraints that do not depend on the segmentation of the data?*
3. *If yes, can one use these constraints to estimate all the subspaces directly from all the data?*
4. *Furthermore, since some information about the number of subspaces must also be contained in the data, is there an algebraic way of obtaining an initial estimate for the number of subspaces?*

1.2 Our approach to mixtures of principal components: GPCA

In this paper, we propose a novel algebraic-geometric approach to modeling mixtures of subspaces called *Generalized Principal Component Analysis* (GPCA), which under mild assumptions guarantees a *unique* global solution to clustering of subspaces based on simple *linear algebraic* techniques. The key to our approach is to view the mixture of subspaces as a projective algebraic variety. Estimating the variety from sample data points becomes a particular case of NLPCA for which one can derive the embedding of the data analytically. Then, estimating the individual subspaces is equivalent to estimating the components of the algebraic variety. Unlike previous work, our approach allows *arbitrary intersections* among the subspaces (as long as they are different) and *does not* require previous segmentation of the data in order to estimate the subspaces. Instead, the subspaces are estimated directly by using *segmentation independent constraints* that are satisfied by *all* data points, regardless of the subspace to which they belong.

More specifically, the main aspects behind our approach are the following:

[1] We thank Dr. David Fleet for pointing out this reference.

1. *Algebraic sets and varieties*: We show in Section 2 that the union of n linear subspaces of $\mathbb{R}^K$ corresponds to the (projective) algebraic set defined by one or more homogeneous polynomials of degree n in K variables. Estimating a collection of subspaces is then equivalent to estimating the algebraic variety defined by such a set of polynomials.

2. *Mixtures of $(K-1)$-dimensional subspaces*: We show in Section 3 that the union of n subspaces of dimension $k = K-1$ is defined by a unique homogeneous polynomial $p_n(\boldsymbol{x})$. The degree of $p_n(\boldsymbol{x})$ turns out to be the number of hyperplanes n and each one of the n factors of $p_n(\boldsymbol{x})$ corresponds to each one of the n hyperplanes. Hence the problem of identifying a collection of hyperplanes boils down to estimating and factoring $p_n(\boldsymbol{x})$. Since every sample point $\boldsymbol{x}$ must satisfy $p_n(\boldsymbol{x}) = 0$, one can retrieve $p_n(\boldsymbol{x})$ directly from the given samples without knowing the segmentation of the data. In fact, the number n of subspaces is exactly the lowest degree of $p_n(\boldsymbol{x})$ such that $p_n(\boldsymbol{x}) = 0$ for all sample points. This leads to a simple matrix rank condition which determines the number of hyperplanes n. Given n, the polynomial $p_n(\boldsymbol{x})$ can be determined from the solution of a set of linear equations. Given $p_n(\boldsymbol{x})$, the estimation of the hyperplanes is essentially equivalent to factoring $p_n(\boldsymbol{x})$ into a product of n linear factors. We present two algorithms for solving the factorization problem. The *polynomial factorization algorithm* (PFA) obtains a normal to each hyperplane from the roots of a polynomial of degree n in one variable and from the solution of $K-2$ linear systems in n variables. Thus the problem has a closed form solution if and only if $n \leq 4$. The *polynomial differentiation algorithm* obtains the normals to each hyperplane from the derivatives of $p_n(\boldsymbol{x})$ evaluated at a collection of n points lying on each one of the hyperplanes.

3. *Mixtures of k-dimensional subspaces ($k < (K-1)$)*: We show in Section 4 that even though the union of n subspaces of dimension $k < K-1$ is defined by *more than one* homogeneous polynomial, one can still reduce it to the case of a single polynomial by projecting the data onto a $(k+1)$-dimensional subspace of $\mathbb{R}^K$. However, in order to project the data we need to know the dimension of the subspaces k. In standard PCA, where $n = 1$, one can always estimate k from the rank of the data matrix. In the case of n subspaces, we derive rank constraints from which one can simultaneously estimate n and k, after embedding the data into a higher-dimensional space. Given n and k, one can use the equations of the projected subspaces to first segment the data using GPCA for hyperplanes and then estimate a basis for the original subspaces using standard PCA. Although a single generic projection is sufficient, we also derive a generalization of the polynomial differentiation algorithm that uses multiple projections to estimate each subspace.

4. *Mixtures of subspaces of arbitrary dimensions*: We show in Section 5 that in the case of subspaces of arbitrary dimensions one cannot recover a set of factorable polynomials representing the algebraic variety. Instead, one can only recover a basis for such polynomials whose elements may not be factorable. However, we show that one can still recover a set of vectors normal to the subspaces by evaluating the derivatives of these polynomials at points on the subspaces, regardless of whether they are factorable or not. Given such normal vectors, the estimation of a basis for the the subspaces and their dimensions can be done by applying standard PCA to the set of normal vectors. This algorithm is in essence a generalization of the polynomial differentiation algorithm to subspaces of arbitrary dimensions.

5. *Maximum likelihood estimation*: In Section 6 consider the GPCA problem in the presence of noisy data. We assume a simple probabilistic model in which the data points are corrupted by zero-mean Gaussian noise and cast GPCA as a constrained nonlinear least squares problem which minimizes the error between noisy points and their projections subject to all mixture constraints. By converting this constrained problem into an unconstrained one, we obtain an optimal function from which the subspaces can be directly recovered using standard nonlinear optimization techniques. We show that the optimal objective function is just a *normalized* version of the algebraic error minimized by our analytic solution to GPCA. Although this means that the algebraic solution to GPCA may be sub-optimal in the presence of noise, we can still use it as a global initializer for any of the existing iterative algorithms for clustering mixtures of subspaces. For example, in Section 7 we derive the equations of the K-subspace and EM algorithms for mixtures of subspaces, and show how to use GPCA to initialize them.

Our theory can be applied to a variety of estimation problems in which the data comes simultaneously from multiple (approximately) linear models. In Section 8 we present experiments on low-dimensional data showing that the polynomial differentiation algorithm gives about half of the error of the polynomial factorization algorithm and improves the performance of iterative techniques, such as K-subspace and EM, by about 50% with respect to random initialization. In Section 9 we present applications of GPCA in computer vision problems, such as detection of vanishing points, 2-D and 3-D motion segmentation, and face clustering under varying illumination.

Remark 1 (Higher order SVD) *It is natural to ask if an algebraic solution to the GPCA problem can be obtained by using some generalization of the SVD to higher-order tensors. It turns out that although the SVD has a multilinear counterpart, the so-called higher order singular value decomposition (HOSVD) [5], such a generalization is not unique. Furthermore, while the SVD of a matrix $A = U\Sigma V^T$ produces a diagonal matrix Σ, the HOSVD of a tensor $\mathcal{A}$ produces a tensor $\mathcal{S}$ which is in general not diagonal. Thus, it is not possible to directly apply HOSVD to the mixture of PCAs problem.*

2 Representing mixtures of subspaces as algebraic sets and varieties

We represent each subspace S_i by choosing a basis

$$B_i \doteq [b_{i1}, \ldots, b_{i(K-k_i)}] \in \mathbb{R}^{K \times (K-k_i)} \tag{1}$$

for its orthogonal complement[2] $S_i^\perp$. With this representation, each subspace is described as

$$S_i = \{x \in \mathbb{R}^K : B_i^T x = 0\} = \{x \in \mathbb{R}^K : \bigwedge_{j=1}^{K-k_i} (b_{ij}^T x = 0)\}. \tag{2}$$

Therefore, an arbitrary point x lies on one of the subspaces if and only if

$$(x \in S_1) \vee \cdots \vee (x \in S_n) \equiv \bigvee_{i=1}^{n} (x \in S_i) \equiv \bigvee_{i=1}^{n} \bigwedge_{j=1}^{K-k_i} (b_{ij}^T x = 0) \equiv \bigwedge_{\sigma} \bigvee_{i=1}^{n} (b_{i\sigma(i)}^T x = 0), \tag{3}$$

where the right hand side (RHS) of (3) is obtained by exchanging ands and ors using De Morgan's laws, and σ represents a particular choice of one normal vector $b_{i\sigma(i)}$ from each basis B_i. Notice that each one of the $\prod_{i=1}^{n}(K-k_i)$ equations in the RHS is of the form

$$\bigvee_{i=1}^{n} (b_{i\sigma(i)}^T x = 0) \equiv \left(p_{n\sigma}(x) = \prod_{i=1}^{n} (b_{i\sigma(i)}^T x) = 0 \right), \tag{4}$$

which is simply a homogeneous polynomial of degree n in K variables, i.e., an element of the ring $R_n(K) = R_n[x_1, \ldots, x_K]$, that is *factorable*[3] as a product of n linear expressions in x, i.e., an element of $R_n^F(K) \subset R_n(K)$. Therefore, the collection of subspaces $Z = \cup_{i=1}^{n} S_i$ is an algebraic set that can be represented with a set of up to $m \leq \prod_{i=1}^{n}(K - k_i)$ independent homogeneous polynomials of the form (4).

Example 1 (Representing the $x-y$ plane and the z axis) *Consider the case of $n = 2$ subspaces of $\mathbb{R}^3$ of dimension $\dim(S_1) = 2$ and $\dim(S_2) = 1$ represented as:*

$$S_1 = \{x \in \mathbb{R}^3 : x_3 = 0\} \quad \text{and} \quad S_2 = \{x \in \mathbb{R}^3 : x_1 = 0 \wedge x_2 = 0\}.$$

A point $x = (x_1, x_2, x_3)^T$ belongs to $S_1 \cup S_2$ if an only if

$$(((x_1 = 0) \vee (x_3 = 0)) \wedge ((x_2 = 0) \vee (x_3 = 0))) \equiv ((x_1 x_3 = 0) \wedge (x_2 x_3 = 0)).$$

Therefore, we can represent $Z = S_1 \cup S_2$ as the zero set of the two polynomials

$$p_{21}(x) = x_1 x_3 \quad \text{and} \quad p_{22}(x) = x_2 x_3.$$

Remark 2 *From an algebraic point of view, determining the algebraic set Z is equivalent to determining the* ideal *$I(Z)$ of Z, i.e., the set of polynomials that vanish on Z [9]. In this case, the ideal $I(Z)$ is a homogeneous ideal that can be graded by degree as $I = I_d \oplus \cdots \oplus I_n \oplus I_{n+1} \oplus \cdots$. Then it is clear that I_n is spanned by the set of polynomials of degree n, $\{p_{n\sigma}(x)\}$. Furthermore, if we let I' be the sub-ideal of I generated by the polynomials $\{p_{n\sigma}(x)\}$, then I is exactly the radical ideal[4] of the ideal I', i.e., $I = rad[I']$.*

[2] One could also choose a basis for S_i directly, especially if $k << K$. However, we will show later in the chapter paper that GPCA can always be reduced to the case $K' = \max\{k_i\} + 1$, hence the orthogonal representation is more convenient.

[3] From now on, we will use the word *factorable* as a shorthand for *factorable into a product of linear forms.*

[4] An ideal I is called a radical ideal if f is in I as long as f^s is in I for some integer s.

The problem of identifying each subspace S_i is then equivalent to one of solving for the normal bases $\{B_i\}_{i=1}^n$ from the set of *nonlinear* equations in (4). A standard technique used in algebra to render a nonlinear problem into a linear one is to find an *embedding* that lifts the problem into a higher-dimensional space. To this end, notice that the set of all homogeneous polynomials of degree n in K variables, $R_n(K)$, can be made into a vector space under the usual addition and scalar multiplication. Furthermore, $R_n(K)$ is generated by the set of monomials $\boldsymbol{x}^{\boldsymbol{n}} = x_1^{n_1} x_2^{n_2} \cdots x_K^{n_K}$, with $0 \leq n_j \leq n$, $j = 1, \ldots, K$, and $n_1 + n_2 + \cdots + n_K = n$. It is readily seen that there are a total of

$$M_n(K) = \binom{n+K-1}{K-1} = \binom{n+K-1}{n} \tag{5}$$

different monomials, thus the dimension of $R_n(K)$ as a vector space is $M_n(K)$.[5] Therefore, we can define the following embedding (or lifting) from $\mathbb{R}^K$ into $\mathbb{R}^{M_n}$.

Definition 1 (Veronese map) *Given n and K, the* Veronese map *of degree n, $\nu_n : \mathbb{R}^K \to \mathbb{R}^{M_n}$, is defined as:*

$$\nu_n : [x_1, \ldots, x_K]^T \mapsto [\ldots, \boldsymbol{x}^{\boldsymbol{n}}, \ldots]^T, \tag{6}$$

where $\boldsymbol{x}^{\boldsymbol{n}}$ is a monomial of the form $x_1^{n_1} x_2^{n_2} \cdots x_K^{n_K}$ with $\boldsymbol{n}$ chosen in the degree-lexicographic order.

Remark 3 (Polynomial embedding) *In the context of Kernel methods, the Veronese map is usually referred to as the polynomial embedding and the ambient space $\mathbb{R}^{M_n}$ is called the* feature *space.*

Example 2 (The Veronese map in two variables) *If $\boldsymbol{x} \in \mathbb{R}^2$, the Veronese map of degree n is given by:*

$$\nu_n(x_1, x_2) = [x_1^n, x_1^{n-1} x_2, x_1^{n-2} x_2^2, \ldots, x_2^n]^T. \tag{7}$$

Thanks to the Veronese map, each polynomial in (4) becomes the following linear expression in the vector coefficients $\boldsymbol{c}_n \in \mathbb{R}^{M_n}$

$$p_n(\boldsymbol{x}) = \nu_n(\boldsymbol{x})^T \boldsymbol{c}_n = \sum c_{n_1, \ldots, n_K} x_1^{n_1} \cdots x_K^{n_K} = 0, \tag{8}$$

where $c_{n_1, \ldots, n_K} \in \mathbb{R}$ represents the coefficient of monomial $\boldsymbol{x}^{\boldsymbol{n}}$. Therefore, if we apply (8) to the given collection of N sample points $\boldsymbol{X} = \{\boldsymbol{x}^j\}_{j=1}^N$, we obtain the following system of linear equations on the vector of coefficients $\boldsymbol{c}_n \in \mathbb{R}^{M_n}$

$$L_n(K)\, \boldsymbol{c}_n \doteq \begin{bmatrix} \nu_n(\boldsymbol{x}^1)^T \\ \nu_n(\boldsymbol{x}^2)^T \\ \vdots \\ \nu_n(\boldsymbol{x}^N)^T \end{bmatrix} \boldsymbol{c}_n = 0 \in \mathbb{R}^N, \tag{9}$$

where $L_n(K) \in \mathbb{R}^{N \times M_n}$ is the matrix of *embedded* data points.[6]

Remark 4 (Kernel Matrix) *In the context of Kernel PCA, if the polynomial embedding is used, then $C = L_n^T L_n \in \mathbb{R}^{M_n \times M_n}$ is exactly the* covariance matrix *in feature space and $\mathcal{K} = L_n L_n^T \in \mathbb{R}^{N \times N}$ is the* kernel matrix *associated with the N embedded samples.*

Remark 5 (GPCA and KPCA) *The basic modeling assumption in KPCA is that there exists an embedding of the data into a higher-dimensional feature space $\boldsymbol{F}$ such that the features live in a linear subspace of $\boldsymbol{F}$. However, there is no general methodology for finding the correct embedding for a particular problem. Equation (9) shows analytically that the commonly used polynomial embedding ν_n is the right one to use in KPCA when the data lives in a collection of subspaces, because the embedded data points $\{\nu_n(\boldsymbol{x}^j)\}_{j=1}^N$ live in a $(M_n - m)$-dimensional subspace of $\mathbb{R}^{M_n}$.*

We notice from equation (9) that the vector of coefficients $\boldsymbol{c}_n$ of each one of the polynomials in the ideal $I_n = \text{span}\{p_n(\boldsymbol{x})\}$ must lie in the null space of the embedded data matrix L_n. Therefore, if $m = \dim(I_n) \leq \prod_{i=1}^n (K - k_i)$ is the number of independent polynomials generating I_n and we are given sufficient sample points $\{\boldsymbol{x}^j\}_{j=1}^N$ in *general position*[7] in $Z = \cup_{i=1}^n S_i$, then

$$M_n - \prod_{i=1}^n (K - k_i) \leq \text{rank}(L_n) = M_n - m \leq M_n - 1. \tag{10}$$

[5] From now on, we will use $M_n \doteq M_n(K)$ whenever the dimension K of the ambient space is understood.

[6] From now on, we will use $L_n \doteq L_n(K)$ whenever the dimension K of the ambient space is understood.

[7] In principle, we need to have $N \geq \sum_{i=1}^n k_i$ sample points in $\cup_{i=1}^n S_i$, with at least k_i points in general position within each subspace S_i, i.e., the k_i points must span S_i. However, because we are representing each polynomial $p_n(\boldsymbol{x})$ linearly via the vector of coefficients $\boldsymbol{c}_n$ we need a number of samples such that a basis for I_n can be uniquely recovered from the null space of L_n. Therefore, by a sufficient number of sample points in general position we mean a number of samples such that $\text{rank}(L_n) = M_n - m$, where $m = \dim(I_n)$.

In principle, given sufficient sample points in general configuration in $\cup_{i=1}^{n} S_i$, one should be able to recover a set of generators for the polynomials in I_n by computing the null space of L_n. However, we can not do so because an arbitrary vector in the null space of L_n corresponds to an arbitrary polynomial in $R_n(K)$ that may not be factorable as a product of n linear forms. For example, both $x_1^2 + x_1x_2$ and $x_2^2 - x_1x_2$ are factorable, but their linear combination (sum) $x_1^2 + x_2^2$ is not. One way of avoiding this problem is to find a basis for the null space of L_n whose elements correspond to coefficients of factorable polynomials. This is in general a daunting task, since it is equivalent to solving a set of polynomials of degree n in several variables.[8]

In the following sections, we propose an alternative solution to the above problem. In Section 3, we consider the case of subspaces of dimension $k_1 = \cdots = k_n = k = K - 1$, i.e., hyperplanes, and show that it can be solved by recovering a single (hence factorable) polynomial. In Section 4, we consider the case of subspaces of equal dimension $k_1 = \cdots = k_n = k < K - 1$ and show that it can be reduced to the case of hyperplanes after projecting the data onto a $(k+1)$-dimensional subspace of $\mathbb{R}^K$. In Section 5, we consider the most general case of subspaces of arbitrary dimensions and propose a solution to the GPCA problem that computes a basis for each subspace in spite of the fact that the polynomials estimated from the null space of L_n may not be factorable.

3 Estimating a mixture of hyperplanes of dimension $K - 1$

In this section, we consider a particular case of the GPCA problem in which all the subspaces have equal dimension $k_1 = \cdots = k_n = k = K - 1$. In Section 3.1, we show that the collection of hyperplanes can be represented with a unique (factorable) polynomial $p_n(x)$ whose degree n, the number of hyperplanes, can be recovered from a rank constraint on the embedded data matrix L_n and whose coefficients c_n can be recovered by solving a linear system. In Section 3.2, we propose an algorithm for estimating the hyperplanes based on polynomial factorization that computes a normal to each hyperplane from the roots of a polynomial of degree n in one variable plus the solution of a collection of $K - 2$ linear systems in n variables. In Section 3.3, we propose a second algorithm for estimating the subspaces based on polynomial differentiation and division, which computes a normal to each hyperplane from the derivatives of $p_n(x)$ evaluated at n points each one lying on each one of the hyperplanes.

3.1 Estimating the number of hyperplanes n and the vector of coefficients c_n

We start by noticing that every $(K-1)$-dimensional subspace $S_i \subset \mathbb{R}^K$ can be defined in terms of a nonzero *normal* vector $b_i \in \mathbb{R}^K$ as follows:[9]

$$S_i = \{x \in \mathbb{R}^K : b_i^T x = b_{i1}x_1 + b_{i2}x_2 + \ldots + b_{iK}x_K = 0\}. \tag{11}$$

Therefore, a point $x \in \mathbb{R}^K$ lying on one of the hyperplanes S_i must satisfy the formula:

$$(b_1^T x = 0) \vee (b_2^T x = 0) \vee \cdots \vee (b_n^T x = 0), \tag{12}$$

which is equivalent to the following homogeneous polynomial of degree n in x with real coefficients:

$$\boxed{p_n(x) = \prod_{i=1}^{n} (b_i^T x) = 0.} \tag{13}$$

The problem of identifying each subspace S_i is then equivalent to one of solving for the vectors $\{b_i\}_{i=1}^n$ from the *nonlinear* equation (13). A standard technique used in algebra to render a nonlinear problem into a linear one is to find an embedding that lifts the problem into a higher-dimensional space. As demonstrated in Section 2, we can use the Veronese map of degree, ν_n, to convert equation (13) into the following linear expression in the vector of coefficients $c_n \in \mathbb{R}^{M_n}$:

$$p_n(x) = \nu_n(x)^T c_n = \sum c_{n_1,n_2,\ldots,n_K} x_1^{n_1} x_2^{n_2} \cdots x_K^{n_K} = 0, \tag{14}$$

where $c_{n_1,\ldots,n_K} \in \mathbb{R}$ represents the coefficient of monomial x^n.

[8] To the best of our knowledge, although it has been shown that a polynomial-time algorithm exists, the algorithm is not yet known [18].

[9] Since the subspaces S_i are all different from each other, we assume that the normal vectors $\{b_i\}_{i=1}^n$ are pairwise linearly independent.

Example 3 (Representing two planes in $\mathbb{R}^3$) *If $n = 2$ and $K = 3$, then we have*

$$p_2(x) = (b_{11}x_1 + b_{12}x_2 + b_{13}x_3)(b_{21}x_1 + b_{22}x_2 + b_{23}x_3)$$
$$\nu_2(x) = [x_1^2, x_1x_2, x_1x_3, x_2^2, x_2x_3, x_3^2]^T$$
$$c_2 = [\underbrace{b_{11}b_{21}}_{c_{2,0,0}}, \underbrace{b_{11}b_{22}+b_{12}b_{21}}_{c_{1,1,0}}, \underbrace{b_{11}b_{23}+b_{13}b_{21}}_{c_{1,0,1}}, \underbrace{b_{12}b_{22}}_{c_{0,2,0}}, \underbrace{b_{12}b_{23}+b_{13}b_{22}}_{c_{0,1,1}}, \underbrace{b_{13}b_{23}}_{c_{0,0,2}}]^T.$$

Remark 6 *Notice that each $c_{n_1,\ldots,n_K}$ is a symmetric multilinear function of $(b_1, b_2, \ldots, b_n)$, that is $c_{n_1,\ldots,n_K}$ is linear in each b_i and:*

$$c_{n_1,\ldots,n_K}(b_1, b_2, \ldots, b_n) = c_{n_1,\ldots,n_K}(b_{\sigma(1)}, b_{\sigma(2)}, \ldots, b_{\sigma(n)}) \text{ for all } \sigma \in \mathfrak{S}_n, \tag{15}$$

where $\mathfrak{S}_n$ is the permutation group of n elements.

Remark 7 (Symmetric Tensors) *Any homogeneous polynomial of degree n in K variables is also a symmetric n-th order tensor in K variables, i.e., an element of $Sym^n(\mathbb{R}^K)$. Furthermore, the vector of coefficients c_n of the polynomial $p_n(x)$ can be interpreted as the symmetric tensor product of the coefficients b_i's of each polynomial of degree 1, that is:*

$$c_n \simeq Sym(b_1 \otimes b_2 \otimes \ldots \otimes b_n) = \sum_{\sigma \in \mathfrak{S}_n} b_{\sigma(1)} \otimes b_{\sigma(2)} \otimes \ldots \otimes b_{\sigma(n)}$$

where $\otimes$ represents the tensor or Kronecker product and $\simeq$ represents the homeomorphism between the symmetric tensor $Sym(b_1 \otimes b_2 \otimes \ldots \otimes b_n)$ in $Sym^n(\mathbb{R}^K)$ and its symmetric part written as a vector c_n in $\mathbb{R}^{M_n}$.

As demonstrated in Section 2 (see equation (9)), after applying (14) to the given collection of N sample points $\{x^j\}_{j=1}^N$, we obtain the following system of linear equations on the vector of coefficients c_n

$$L_n\, c_n \doteq \begin{bmatrix} \nu_n(x^1)^T \\ \nu_n(x^2)^T \\ \vdots \\ \nu_n(x^N)^T \end{bmatrix} c_n = 0 \in \mathbb{R}^N. \tag{16}$$

Remark 8 (Brill's equations on the entries of c_n) *Given n, one can solve for c_n from (16) in a linear fashion. However, notice that the entries of c_n cannot be independent from each other, because the polynomial $p_n(x)$ must be factorable as a product of linear forms. The factorability of $p_n(x)$ enforces constraints on the entries of c_n, which are polynomials of degree $(n+1)$ on M_n variables, the so-called Brill's equations [7]. In Example 3, where $n = 2$ and $K = 3$, Brill's equations are $c_{1,0,1}^2 c_{0,2,0} - c_{1,1,0}c_{1,0,1}c_{0,1,1} + c_{1,1,0}^2 c_{0,0,2} + c_{2,0,0}(c_{0,1,1}^2 - 4c_{0,2,0}c_{0,0,2}) = 0$. However, if we are given enough sample points N and there is no noise on the data, then the solution of (16) will automatically satisfy Brill's equations. Understanding how to use Brill's equations to improve the estimation of c_n in the case of noisy data will be a subject of future research.*

We now study under what conditions we can solve for n and c_n from equation (16). To this end, notice that if the number of hyperplanes n was known, we could immediately recover c_n as the eigenvector of $L_n^T L_n$ associated with its smallest eigenvalue. However, since the above linear system (16) depends explicitly on the number of hyperplanes n, we cannot estimate c_n directly without knowing n in advance. It turns out that the estimation of the number of hyperplanes n is very much related to the conditions under which the solution for c_n is unique (up to a scale factor), as stated by the following theorem.

Theorem 1 (Number of hyperplanes) *Assume that a collection of $N \geq M_n - 1$ sample points $\{x^j\}_{j=1}^N$ on n different $(K-1)$-dimensional subspaces of $\mathbb{R}^K$ is given. Let $L_i \in \mathbb{R}^{N \times M_i}$ be the matrix defined in (16), but computed with the Veronese map $\nu_i(x)$ of degree i. If the sample points are in general position and at least $K-1$ points correspond to each hyperplane, then:*

$$\text{rank}(L_i) \begin{cases} > M_i - 1, & i < n, \\ = M_i - 1, & i = n, \\ < M_i - 1, & i > n. \end{cases} \tag{17}$$

Therefore, the number n of hyperplanes is given by:

$$\boxed{n = \min\{i : \text{rank}(L_i) = M_i - 1\}.} \tag{18}$$

Proof. Consider the polynomial $p_n(\boldsymbol{x})$ as a polynomial over the algebraically closed field $\mathbb{C}$ and assume that each hyperplane $\boldsymbol{b}_i^T\boldsymbol{x} = 0$ is different from each other. Then the ideal I generated by $p_n(\boldsymbol{x})$ is a *radical ideal* with $p_n(\boldsymbol{x})$ as its only generator. According to Hilbert's Nullstellensatz (see page 380, [14]), there is a one-to-one correspondence between such an ideal I and the algebraic set (also called algebraic variety in Algebra)

$$Z(I) \doteq \{\boldsymbol{x} : \forall p \in I, p(\boldsymbol{x}) = 0\} \subset \mathbb{C}^K$$

associated with it. Hence its generator $p_n(\boldsymbol{x})$ is uniquely determined by points in this algebraic set. By definition, $p_n(\boldsymbol{x})$ has the lowest degree among all the elements in the ideal I. Hence no polynomial with lower degree would vanish on all points in these subspaces. Furthermore, since all coefficients $\boldsymbol{b}_i$ are real, if $\boldsymbol{x} + \sqrt{-1}\boldsymbol{y} \in \mathbb{C}^K$ is in $Z(I)$, both $\boldsymbol{x} \in \mathbb{R}^K$ and $\boldsymbol{y} \in \mathbb{R}^K$ are in the set of (real) subspaces, because $\boldsymbol{b}_i^T(\boldsymbol{x} + \sqrt{-1}\boldsymbol{y}) = 0 \Leftrightarrow \boldsymbol{b}_i^T\boldsymbol{x} = 0 \wedge \boldsymbol{b}_i^T\boldsymbol{y} = 0$. Hence all points on the (real) subspaces determine the polynomial $p_n(\boldsymbol{x})$ uniquely and vice-versa. Therefore, there is no polynomial of degree $i < n$ that is satisfied by all the data, hence $\text{rank}(L_i) = M_i$ for $i < n$. Conversely, there are multiple polynomials of degree $i > n$, namely any multiple of $p_n(\boldsymbol{x})$, which are satisfied by all the data, hence $\text{rank}(L_i) < M_i - 1$ for $i > n$. Thus the case $i = n$ is the only one in which the linear system (16) has a unique solution (up to a scale factor), namely the vector of coefficients $\boldsymbol{c}_n$ of the polynomial $p_n(\boldsymbol{x})$. ∎

Remark 9 *In the presence of noise, one cannot directly estimate n from (18), because the matrix L_i is always full rank. In practice we declare the rank of L_i to be r if $\sigma_{r+1}/(\sigma_1 + \cdots + \sigma_r) < \epsilon$, where σ_k is the k-th singular value of L_i and $\epsilon > 0$ is a pre-specified threshold. We have found this simple criterion to work well in our experiments.*

Theorem 1 and the linear system in equation (16) allow us to determine the number of hyperplanes n and the vector of coefficients $\boldsymbol{c}_n$, respectively, from sample points $\{\boldsymbol{x}^j\}_{j=1}^N$. The rest of the problem becomes now how to recover the normal vectors $\{\boldsymbol{b}_i\}_{i=1}^n$ from $\boldsymbol{c}_n$. Sections 3.2 and 3.3 present two algorithms for recovering the normal vectors based on polynomial factorization and polynomial differentiation and division, respectively.

3.2 Estimating the hyperplanes: the polynomial factorization algorithm (PFA)

In this section, we give a constructive solution to the GPCA problem in the case of hyperplanes based on polynomial factorization. More specifically, we prove the following theorem:

Theorem 2 (GPCA for mixtures of hyperplanes by polynomial factorization) *The GPCA problem with $k_1 = \cdots = k_n = k = K - 1$ is algebraically equivalent to the factorization of a homogeneous polynomial of degree n in K variables into a product of n polynomials of degree 1. This is in turn algebraically equivalent to solving for the roots of a polynomial of degree n in one variable plus solving $K - 2$ linear systems in n variables. Thus the GPCA problem for $k = K - 1$ has a unique solution which can be obtained in closed form when $n \leq 4$.*

3.2.1 GPCA as a polynomial factorization problem

From equations (13) and (14) we have that:

$$p_n(\boldsymbol{x}) = \sum c_{n_1,n_2,\ldots,n_K} x_1^{n_1} x_2^{n_2} \cdots x_K^{n_K} = \prod_{i=1}^{n} \left(\sum_{j=1}^{K} b_{ij} x_j \right).$$

Therefore, the problem of recovering $\{\boldsymbol{b}_i\}_{i=1}^n$ from $\boldsymbol{c}_n$ is equivalent to the following polynomial factorization problem.

Problem 2 (Factorization of homogeneous polynomials)

Given a factorable homogeneous polynomial of degree n in K variables $p_n(\boldsymbol{x}) \in R_n^F(K)$, factor it into n *different* polynomials of degree one in K variables $\{(\boldsymbol{b}_i^T\boldsymbol{x}) \in R_1(K)\}_{i=1}^n$.

Remark 10 (Factorization of symmetric tensors) *The polynomial factorization problem can also be interpreted as a tensor factorization problem: Given an n-th order symmetric tensor $\mathcal{V}$ in $Sym^n(\mathbb{R}^K)$, find vectors $v_1, v_2, \ldots, v_n \in \mathbb{R}^K$ such that*

$$\mathcal{V} = Sym(v_1 \otimes v_2 \otimes \ldots \otimes v_n) = \sum_{\sigma \in \mathfrak{S}_n} v_{\sigma(1)} \otimes v_{\sigma(2)} \otimes \ldots \otimes v_{\sigma(n)}$$

Notice that

$$\nu : \mathbb{R}^{K \times n} \to Sym^n(\mathbb{R}^K); \quad (v_1, v_2, \ldots, v_n) \mapsto Sym(v_1 \otimes v_2 \otimes \ldots \otimes v_n)$$

maps a $K \times n$-dimensional space to an M_n-dimensional space. In general M_n is much larger than $(K \times n - n + 1)$.[10]

[10] We here subtract $n - 1$ parameters on the right is because we only have to consider unit vectors.

Therefore, not *all symmetric tensors in the space $Sym^n(\mathbb{R}^K)$ can be factored in the above way.*

Notice that an arbitrary element of $R_n(K)$ is *not* necessarily factorable into n distinct elements of $R_1(K)$, e.g., the polynomial $x_1^2 + x_1x_2 + x_2^2$ is not. However, the existence of a factorization for $p_n(x)$ is guaranteed by its definition as a product of linear functionals. In relation to the uniqueness of the factorization, it is clear that each b_i can be multiplied by an arbitrary scale to obtain the same c_n up to scale. Since we can fix the norm of c_n to be 1 when solving (16), we are actually free to choose the scale of $n-1$ of the b_i's only. The following proposition is a consequence of the well-known Gauss Lemma in Algebra (see page 181, [14]) and guarantees the uniqueness of the factorization of $p_n(x)$ up to $n-1$ scales:

Proposition 1 (Uniqueness of the factorization) *Since $\mathbb{R}$ is a factorial ring, the set of polynomials in K variables $R[x_1, \ldots, x_K]$ is also factorial, that is any polynomial $p \in R[x_1, \ldots, x_K]$ has a unique factorization into irreducible elements. In particular, any element of the set of homogeneous polynomials $R_n(K) \subset R[x_1, \ldots, x_K]$ has a unique factorization.*

3.2.2 Solving the polynomial factorization problem

Knowing the existence and uniqueness of a solution to the polynomial factorization problem (Problem 2), we are now interested in finding an algorithm that recovers the b_i's from c_n. For ease of exposition, we will first present an example with the case $n = 2$ and $K = 3$, because it gives most of the intuition about our general algorithm for arbitrary n and K.

Example 4 (Estimating two planes in $\mathbb{R}^3$) *Consider the case $n = 2$ and $K = 3$ illustrated in Example 3. Then*

$$\begin{aligned} p_2(x) &= (b_1^T x)(b_2^T x) = (b_{11}x_1 + b_{12}x_2 + b_{13}x_3)(b_{21}x_1 + b_{22}x_2 + b_{23}x_3) \\ &= \underbrace{(b_{11}b_{21})}_{c_{2,0,0}}x_1^2 + \underbrace{(b_{11}b_{22}+b_{12}b_{21})}_{c_{1,1,0}}x_1x_2 + \underbrace{(b_{11}b_{23}+b_{13}b_{21})}_{c_{1,0,1}}x_1x_3 + \\ &\quad \underbrace{(b_{12}b_{22})}_{c_{0,2,0}}x_2^2 + \underbrace{(b_{12}b_{23}+b_{13}b_{22})}_{c_{0,1,1}}x_2x_3 + \underbrace{(b_{13}b_{23})}_{c_{0,0,2}}x_3^2. \end{aligned}$$

We notice that the last three terms correspond to a polynomial in x_2 and x_3 only, which is equal to the product of the last two terms of the original factors, i.e.,

$$c_{0,2,0}x_2^2 + c_{0,1,1}x_2x_3 + c_{0,0,2}x_3^2 = (b_{12}x_2 + b_{13}x_3)(b_{22}x_2 + b_{23}x_3).$$

After dividing by x_3^2 and letting $t = x_2/x_3$ we obtain

$$q_2(t) = c_{0,2,0}t^2 + c_{0,1,1}t + c_{0,0,2} = (b_{12}t + b_{13})(b_{22}t + b_{23}).$$

Since $c_2 \in \mathbb{R}^6$ is known, so is the second order polynomial $q_2(t)$. Thus we can obtain $\frac{b_{13}}{b_{12}}$ and $\frac{b_{23}}{b_{22}}$ from the roots t_1 and t_2 of $q_2(t)$. Since b_1 and b_2 are only computable up to scale, we can actually divide c_2 by $c_{0,2,0}$ (if nonzero) and set the last two entries of b_1 and b_2 to

$$b_{12} = 1, \quad b_{13} = -t_1, \quad b_{22} = 1, \quad b_{23} = -t_2.$$

We are left with the computation of the first entry of b_1 and b_2. We notice that the coefficients $c_{1,1,0}$ and $c_{1,0,1}$ are linear functions of the unknowns b_{11} and b_{21}. Thus we can obtain b_{11} and b_{21} from

$$\begin{bmatrix} b_{22} & b_{12} \\ b_{23} & b_{13} \end{bmatrix} \begin{bmatrix} b_{11} \\ b_{21} \end{bmatrix} = \begin{bmatrix} c_{1,1,0} \\ c_{1,0,1} \end{bmatrix} \tag{19}$$

provided that $b_{22}b_{13} - b_{23}b_{12} \neq 0$, i.e., if $t_1 \neq t_2$.

We conclude from the Example 4 that, if $c_{0,2,0} = b_{12}b_{22} \neq 0$ and $b_{22}b_{13} - b_{23}b_{12} \neq 0$, then the factorization of a homogeneous polynomial of degree $n = 2$ in $K = 3$ variables can be done in the following two steps: (1) solve for the last two entries of $\{b_i\}_{i=1}^n$ from the roots of a polynomial $q_n(t)$ associated with the last $n+1 = 3$ coefficients of $p_n(x)$; and (2) solve for the first $K-2$ entries of $\{b_i\}_{i=1}^n$ from $K-2$ linear systems in n variables.

We now generalize these two steps to arbitrary n and K.

1. **Solving for the last two entries of each b_i:** Consider the last $n+1$ coefficients of $p_n(\boldsymbol{x})$:

$$[c_{0,\ldots,0,n,0}\ ,\ c_{0,\ldots,0,n-1,1}\ ,\ \cdots\ ,\ c_{0,\ldots,0,0,n}]^T \in \mathbb{R}^{n+1}, \tag{20}$$

which define the following homogeneous polynomial of degree n in the two variables x_{K-1} and x_K:

$$\sum c_{0,\ldots,0,n_{K-1},n_K} x_{K-1}^{n_{K-1}} x_K^{n_K} = \prod_{i=1}^{n} (b_{iK-1}x_{K-1} + b_{iK}x_K)\,. \tag{21}$$

Letting $t = x_{K-1}/x_K$, we have that:

$$\prod_{i=1}^{n} (b_{iK-1}x_{K-1} + b_{iK}x_K) = 0 \Leftrightarrow \prod_{i=1}^{n} (b_{iK-1}t + b_{iK}) = 0.$$

Hence the n roots of the polynomial

$$q_n(t) = c_{0,\ldots,0,n,0}t^n + c_{0,\ldots,0,n-1,1}t^{n-1} + \cdots + c_{0,\ldots,0,0,n} \tag{22}$$

are exactly $t_i = -b_{iK}/b_{iK-1}$, for all $i = 1,\ldots,n$. Therefore, after dividing $\mathbf{c}_n$ by $c_{0,\ldots,0,n,0}$ (if nonzero), we obtain the last two entries of each $\boldsymbol{b}_i$ as:

$$(b_{iK-1}\ ,\ b_{iK}) = (1\ ,\ -t_i) \qquad i = 1,\ldots,n. \tag{23}$$

If $b_{iK-1} = 0$ for some i, then some of leading coefficients of $q_n(t)$ are zero and we cannot proceed as before, because $q_n(t)$ has less than n roots. More specifically, assume that the first $\ell \leq n$ coefficients of $q_n(t)$ are zero and divide $\mathbf{c}_n$ by the $(\ell+1)$-st coefficient. In this case, we can choose $(b_{iK-1}, b_{iK}) = (0,1)$, for $i = 1,\ldots,\ell$, and obtain $\{(b_{iK-1}, b_{iK})\}_{i=n-\ell+1}^{n}$ from the $n-\ell$ roots of $q_n(t)$ using equation (23). Finally, if all the coefficients of $q_n(t)$ are equal to zero, we set $(b_{iK-1}, b_{iK}) = (0,0)$, for all $i = 1,\ldots,n$.

2. **Solving for the first $K-2$ entries of each b_i:** We have demonstrated how to obtain the last two entries of each $\boldsymbol{b}_i$ from the roots of a polynomial of degree n in one variable. We are now left with the computation of the first $K-2$ entries of each $\boldsymbol{b}_i$. We assume that we have computed b_{ij}, $i = 1,\ldots,n$, $j = J+1,\ldots K$ for some J, starting with the case $J = K-2$, and show how to linearly solve for b_{iJ}, $i = 1,\ldots,n$. As in Example 4, the key is to consider the coefficients of $p_n(\boldsymbol{x})$ associated to monomials of the form $x_J\, x_{J+1}^{n_{J+1}} \cdots x_K^{n_K}$, which are *linear* in x_J. These coefficients are of the form $c_{0,\ldots,0,1,n_{J+1},\ldots,n_K}$ and are linear in b_{iJ}. To see this, we notice that the polynomial $\sum c_{0,\ldots,0,1,n_{J+1},\ldots,n_K} x_{J+1}^{n_{J+1}} \cdots x_K^{n_K}$ is equal to the partial of $p_n(\boldsymbol{x})$ with respect to x_J evaluated at $x_1 = x_2 = \cdots = x_J = 0$. Since

$$\frac{\partial p_n(\boldsymbol{x})}{\partial x_J} = \frac{\partial}{\partial x_J}\left(\prod_{i=1}^{n}(\boldsymbol{b}_i^T\boldsymbol{x})\right) = \sum_{i=1}^{n} b_{iJ}\left(\prod_{\ell=1}^{i-1}(\boldsymbol{b}_i^T\boldsymbol{x})\prod_{\ell=i+1}^{n}(\boldsymbol{b}_i^T\boldsymbol{x})\right), \tag{24}$$

after evaluating at $x_1 = x_2 = \cdots = x_J = 0$ we obtain

$$\sum c_{0,\ldots,0,1,n_{J+1},\ldots,n_K} x_{J+1}^{n_{J+1}} \cdots x_K^{n_K} = \sum_{i=1}^{n} b_{iJ} g_i^J(\boldsymbol{x}), \tag{25}$$

where

$$g_i^J(\boldsymbol{x}) = \prod_{\ell=1}^{i-1}\left(\sum_{j=J+1}^{K} b_{\ell j}x_j\right)\prod_{\ell=i+1}^{n}\left(\sum_{j=J+1}^{K} b_{\ell j}x_j\right) \tag{26}$$

is a homogeneous polynomial of degree $n-1$ in the last $K-J$ variables in $\boldsymbol{x}$. Let $\mathcal{V}_i^J$ be the vector of coefficients of the polynomial $g_i^J(\boldsymbol{x})$. From equation (25) we get

$$\begin{bmatrix} \mathcal{V}_1^J & \mathcal{V}_2^J & \cdots & \mathcal{V}_n^J \end{bmatrix} \begin{bmatrix} b_{1J} \\ b_{2J} \\ \vdots \\ b_{nJ} \end{bmatrix} = \begin{bmatrix} c_{0,\ldots,0,1,n-1,0,\ldots,0} \\ c_{0,\ldots,0,1,n-2,1,\ldots,0} \\ \vdots \\ c_{0,\ldots,0,1,0,0,\ldots,n-1} \end{bmatrix} \tag{27}$$

from which we can linearly solve for the unknowns $\{b_{iJ}\}_{i=1}^{n}$. Notice that the vectors $\{\mathcal{V}_i^J\}_{i=1}^{n}$ are known, because they are functions of the known $\{b_{ij}\}_{i=1}^{n}$, where $j = J+1,\ldots K$.

3.2.3 Uniqueness of the solution given by the factorization algorithm

According to Proposition 1, the polynomial factorization problem admits a unique solution. However, the factorization algorithm that we have just proposed may not give a unique solution. For example, the algorithm fails for $p_2(\boldsymbol{x}) = x_2x_3 \in R_2(3)$. This is because it does not use *all* the entries of c_n in order to obtain the factorization. In fact, it only uses the entries that are linear in the unknowns.

We will now analyze the conditions under which the proposed algorithm *does* provide a unique solution. From equation (27), we notice that this is the case if and only if the vectors $\mathcal{V}_1^J, \ldots, \mathcal{V}_n^J$ are linearly independent. The following proposition gives a more specific necessary and sufficient condition for the uniqueness in terms of the normal vectors $\{b_i\}_{i=1}^n$:

Proposition 2 (Uniqueness of the solution given by the algorithm) *The vectors $\{\mathcal{V}_i^J\}_{i=1}^n$ are linearly independent if and only if for all $r \neq s$, $1 \leq r, s \leq n$, the vectors $(b_{rJ+1}, b_{rJ+2}, \ldots, b_{rK})$ and $(b_{sJ+1}, b_{sJ+2}, \ldots, b_{sK})$ are pairwise linearly independent. Furthermore, the vectors $\{\mathcal{V}_i^{K-2}\}_{i=1}^n$ are linearly independent if and only if the polynomial $q_n(t)$ has distinct roots and at most one of its leading coefficients is zero.*

Proof. We do the proof by induction on n. Let $h_i^J(\boldsymbol{x}) \doteq \sum_{j=J+1}^K b_{ij}x_j$. By definition, the vectors $\mathcal{V}_i^J$ are linearly independent if

$$h^J \doteq \sum_{i=1}^n \alpha_i h_1^J \cdots h_{i-1}^J h_{i+1}^J \cdots h_n^J = 0 \tag{28}$$

if and only if $\alpha_i = 0, \alpha_i \in \mathbb{R}$, $i = 1, \ldots, n$. If $n = 2$, (28) reduces to:

$$\alpha_1 h_2^J + \alpha_2 h_1^J = 0. \tag{29}$$

Therefore $\mathcal{V}_1^J$ is independent from $\mathcal{V}_2^J$ if and only if h_1^J is independent from h_2^J, which happens if and only if $(b_{1J+1}, b_{1J+2}, \ldots, b_{1K})$ is independent from $(b_{2J+1}, b_{2J+2}, \ldots, b_{2K})$. Thus the proposition is true for $n = 2$. Now assume the that the proposition is true for $n - 1$. After dividing (28) by h_1^J we obtain:

$$\frac{h^J}{h_1^J} = \alpha_1 \frac{h_2^J \cdots h_n^J}{h_1^J} + \sum_{i=2}^n \alpha_i \underbrace{h_2^J \cdots h_{i-1}^J h_{i+1}^J \cdots h_n^J}_{\text{polynomial in } R_{n-1}(K-J)} = 0. \tag{30}$$

If $\alpha_1 = 0$, then the proof reduces to the case $n - 1$, which is true by the induction hypothesis. If $\alpha_1 \neq 0$, then $\frac{h_2^J \cdots h_n^J}{h_1^J}$ must belong to $R_{n-1}(K - J)$, which happens only if h_1^J is proportional to some h_i^J, $i = 2, \ldots, n$, i.e., if $(b_{1J+1}, b_{1J+2}, \ldots, b_{1K})$ is proportional to some $(b_{iJ+1}, b_{iJ+2}, \ldots, b_{iK})$. The fact that the choice of h_1^J as a divisor was arbitrary completes the proof of the first part. As for the second part, by construction the vectors (b_{rK-1}, b_{rK}) and (b_{sK-1}, b_{sK}) are independent if and only if the roots of $q_n(t)$ are distinct and $q_n(t)$ has at most one leading coefficient equal to zero. ∎

3.2.4 Obtaining a unique solution for the degenerate cases

Proposition 2 states that in order for the $K - 2$ linear systems in (27) to have a unique solution, we must make sure that the polynomial $q_n(t)$ is non-degenerate, i.e., $q_n(t)$ has no repeated roots and at most one of its leading coefficients is zero. One possible approach to avoid non-uniqueness is to choose a pair of variables $(x_j, x_{j'})$ for which the corresponding polynomial $q_n(t)$ is non-degenerate. The following proposition guarantees that we can do so if $n = 2$. Unfortunately the result is not true for $n > 2$ as shown by Example 5.

Proposition 3 (Choosing a good pair of variables when $n = 2$)
Given a factorable polynomial $p_2(\boldsymbol{x})$, there exist a pair of variables $(x_j, x_{j'})$ such that the associated polynomial $q_2(t)$ is non-degenerate.

Proof. For the sake of contradiction, assume that for any pair of variables $(x_j, x_{j'})$ the associated polynomial $q_2(t)$ has a repeated root or the first two leading coefficients are zero. Proposition 2 implies that for all $j \neq j'$, $(b_{1j}, b_{1j'})$ is parallel to $(b_{2j}, b_{2j'})$, hence, all the 2×2 minors of the matrix $B = [b_1\ b_2]^T \in \mathbb{R}^{2\times K}$ are equal to zero. This implies that b_1 is parallel to b_2, violating the assumption of different subspaces. ∎

Example 5 (A polynomial with repeated roots) *Consider the following polynomial in $R_3(3)$:*

$$p_3(x) = (x_1 + x_2 + x_3)(x_1 + 2x_2 + 2x_3)(x_1 + 2x_2 + x_3).$$

The associated polynomials in two variables are $4x_2^3 + 10x_2^2x_3 + 8x_2x_3^2 + 2x_3^2$, $x_1^3 + 4x_1^2x_3 + 5x_1x_3^2 + 2x_3^2$ and $x_1^3 + 5x_1^2x_2 + 8x_1x_2^2 + 4x_2^2$, and all of them have repeated roots.

We conclude that, even though the uniqueness of the factorization is guaranteed by Proposition 1, there are some cases for $n > 2$ for which our factorization algorithm (based on solving for the roots a polynomial of degree n in one variable plus $K - 2$ linear systems in n variables) will not be able to provide the *unique* solution. The reason for this is that our algorithm is not using *all* the coefficients in $\mathbf{c}_n$, but only the ones for which the problem is linear.

One possible algorithm to obtain a unique solution for these degenerate cases is to consider the coefficients of $p_n(x)$ which have not been used. Since the equations associated to those coefficients are polynomials of degree $d \geq 2$ in the unknowns $\{b_{iJ}\}_{i=1}^n$, we will not pursue this direction here. Instead, we will try to find a linear transformation on x, hence on the b_{ij}'s, that gives a new vector of coefficients $\mathbf{c}'_n$ whose associated polynomial $q'_n(t)$ is non-degenerate. It is clear that we only need to modify the entries of each b_i associated to the last two variables. Thus, we consider the following linear transformation $T : \mathbb{R}^K \to \mathbb{R}^K$:

$$x = Ty = \begin{bmatrix} 1 & 0 & \cdots & 0 & t & t \\ 0 & 1 & & 0 & t & t \\ \vdots & & \ddots & & & \vdots \\ & & & & 1 & t \\ 0 & 0 & \cdots & & 0 & 1 \end{bmatrix} y. \tag{31}$$

Under this transformation, the polynomial $p_n(x)$ becomes:

$$p'_n(y) = p_n(Ty) = \prod_{i=1}^{n} b_i^T(Ty) = \prod_{i=1}^{n} \Big(\sum_{j=1}^{K-1} b_{ij}y_j + \underbrace{\Big[t \sum_{j=1}^{K-2} b_{ij} + b_{iK-1} \Big]}_{b'_{iK-1}(t)} y_{K-1} + \underbrace{\Big[t \sum_{j=1}^{K-1} b_{ij} + b_{iK} \Big]}_{b'_{iK}(t)} y_K \Big).$$

Therefore, the polynomial associated to y_{K-1} and y_K will have distinct roots for all $t \in \mathbb{R}$, except for the t's which are roots of the following second order polynomial:

$$b'_{rK-1}(t)b'_{sK}(t) = b'_{sK-1}(t)b'_{rK}(t) \tag{32}$$

for some $r \neq s$, $1 \leq r, s \leq n$. Since there are a total of $n(n+1)/2$ such polynomials, each of them having at most 2 roots, we can choose t arbitrarily, except for $n(n+1)$ values.

Once t has been chosen, we need to compute the coefficients $\mathbf{c}'_n$ of the new polynomial $p'_n(y)$. The following proposition establishes the relationship between $\mathbf{c}_n$ and $\mathbf{c}'_n$:

Proposition 4 *Let $\mathbf{c}_n$ and $\mathbf{c}'_n$ be the coefficients of the polynomials $p_n(x) \in R_n(K)$ and $p'_n(y) = p_n(Tx) \in R_n(K)$, respectively, where $T : \mathbb{R}^K \to \mathbb{R}^K$ is a non-singular linear map. Then T induces a linear transformation $\tilde{T} : \mathbb{R}^{M_n} \to \mathbb{R}^{M_n}$, $\mathbf{c}_n \mapsto \mathbf{c}'_n = \tilde{T}\mathbf{c}_n$. Furthermore, the column of $\tilde{T}$ associated to $c_{n_1,n_2,\ldots,n_K}$ is given by the coefficients of the polynomial:*

$$(\ell_1^T y)^{n_1} (\ell_2^T y)^{n_2} \cdots (\ell_K^T y)^{n_K}, \tag{33}$$

where ℓ_j^T is the j-th row of T.

Proof. Let $p_n^1(x), p_n^2(x) \in R_n^F(K)$ and $\alpha, \beta \in \mathbb{R}$. Then the polynomial $\alpha p_n^1(x) + \beta p_n^2(x)$ is transformed by T into $\alpha p_n^1(Ly) + \beta p_n^2(Ly)$. Therefore $\tilde{T}$ is linear. Now in order to find the column of $\tilde{T}$ associated to $c_{n_1,n_2,\ldots,n_K}$, we just need to apply the transformation $\tilde{T}$ to the monomial $x_1^{n_1}x_2^{n_2}\cdots x_K^{n_K} = (e_1^T x)^{n_1}(e_2^T x)^{n_2}\cdots(e_K^T x)^{n_K}$, where $\{e_j\}_{j=1}^K$ is the standard basis for $\mathbb{R}^K$. We obtain $(e_1^T Ty)^{n_1}(e_2^T Ty)^{n_2}\cdots(e_K^T Ty)^{n_K}$, or equivalently $(\ell_1^T y)^{n_1} (\ell_2^T y)^{n_2} \cdots (\ell_K^T y)^{n_K}$. ■

Remark 11 *Due to the upper triangular structure of T in (31), the matrix $\tilde{T}$ will be lower triangular. Furthermore, since each entry of T is a polynomial of degree at most 1 in t, the entries of $\tilde{T}$ will be polynomials of degree at most n in t.*

By construction, the polynomial $q'_n(t)$ associated to the last two variables of $p'_n(y)$ will have no repeated roots. Therefore, we can apply the previously described factorization algorithm to the coefficients c'_n of $p'_n(y)$ to obtain the set of transformed normal vectors $\{b'_i\}_{i=1}^n$. Since by definition of $p'_n(y)$ we have $b_i'^T = b_i^T T$, the original normal vectors are given by $b_i = T^{-T} b'_i$. It turns out that, due to the particular structure of T, we do not actually need to compute T^{-T}. We can obtain $\{b_i\}_{i=1}^n$ directly from $\{b'_i\}_{i=1}^n$ and t as follows:

$$\begin{array}{lcll} b_{ij} & = & b'_{ij}, & i = 1,\ldots,n,\ j = 1,\ldots,K-2 \\ b_{iK-1} & = & b'_{iK-1} - t\sum_{j=1}^{K-2} b_{ij}, & i = 1,\ldots,n \\ b_{iK} & = & b'_{iK} - t\sum_{j=1}^{K-1} b_{ij}, & i = 1,\ldots,n. \end{array} \tag{34}$$

We illustrate the proposed transformation with the following example:

Example 6 *Let $n = 3$ and $K = 3$. Then T and $\tilde{T}$ are given by:*

$$T = \begin{bmatrix} 1 & t & t \\ 0 & 1 & t \\ 0 & 0 & 1 \end{bmatrix} \tag{35}$$

and

$$\tilde{T} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 3t & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 3t & t & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 3t^2 & 2t & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 6t^2 & 2t^2+2t & 2t & 2t & 1 & 0 & 0 & 0 & 0 & 0 \\ 3t^2 & 2t^2 & 2t & t^2 & t & 1 & 0 & 0 & 0 & 0 \\ t^3 & t^2 & 0 & t & 0 & 0 & 1 & 0 & 0 & 0 \\ 3t^3 & t^3+2t^2 & t^2 & 2t^2+t & t & 0 & 3t & 1 & 0 & 0 \\ 3t^3 & 2t^3+t^2 & 2t^2 & t^3+2t^2 & t^2+t & t & 3t^2 & 2t & 1 & 0 \\ t^3 & t^3 & t^2 & t^3 & t^2 & t & t^3 & t^2 & t & 1 \end{bmatrix}. \tag{36}$$

We summarize the results of this section with the polynomial factorization algorithm (PFA) for mixtures of hyperplanes, a GPCA problem with $k_1 = \cdots = k_n = K - 1$.

Algorithm 1 (Polynomial Factorization Algorithm (PFA) for Mixtures of Hyperplanes)

Given sample points $\{x^j\}_{j=1}^N$ lying on a collection of hyperplanes $\{S_i \subset \mathbb{R}^K\}_{i=1}^n$, find the number of hyperplanes n and the normal vector to each hyperplane $\{b_i \in \mathbb{R}^K\}_{i=1}^n$ as follows:

1. Apply the Veronese map of order i, for $i = 1, 2, \ldots$, to the vectors $\{x^j\}_{j=1}^N$ and form the matrix L_i in (16). Stop when $\text{rank}(L_i) = M_i - 1$ and set the number of hyperplanes n to be the current i. Then solve for c_n from $L_n c_n = 0$ and normalize so that $\|c_n\| = 1$.
2. (a) Get the coefficients of the univariate polynomial $q_n(t)$ from the last $n+1$ entries of c_n.
   (b) If the first ℓ, $0 \leq \ell \leq n$, coefficients of $q_n(t)$ are equal to zero, set $(b_{iK-1}, b_{iK}) = (0, 1)$ for $i = 1, \ldots, \ell$. Then use (23) to compute $\{(b_{iK-1}, b_{iK})\}_{i=n-\ell+1}^n$ from the $n - \ell$ roots of $q_n(t)$.
   (c) If all the coefficients of $q_n(t)$ are zero, set $(b_{iK-1}, b_{iK}) = (0, 0)$, for $i = 1, \ldots, n$.
   (d) If (b_{rK-1}, b_{rK}) is parallel to (b_{sK-1}, b_{sK}) for some $r \neq s$, apply the transformation $x = Ty$ in (31) and repeat 2(a), 2(b) and 2(c) for the transformed polynomial $p'_n(y)$ to obtain $\{(b'_{iK-1}, b'_{iK})\}_{i=1}^n$.
3. Given (b_{iK-1}, b_{iK}), $i = 1, \ldots, n$, solve for $\{b_{iJ}\}_{i=1}^n$ from (27) for $J = K-2, \ldots, 1$. If a transformation T was used in 2(d), then compute b_i from b'_i and t using equation (34).

3.3 Estimating the hyperplanes: the polynomial differentiation algorithm (PDA)

The main attraction of the polynomial factorization algorithm (PFA) (Algorithm 1) is that it shows that one can estimate a collection of hyperplanes in a purely algebraic fashion. Furthermore, the PFA does not requires initialization for the normal vectors or the clustering of the data, and the solution can be obtained in closed form for $n \leq 4$ hyperplanes.

However, the PFA presents some disadvantages when applied to noisy data. For instance, there are some degenerate cases for which some of the $K-2$ linear systems have more than one solution, namely whenever the polynomial $q_n(t)$ has repeated roots. Furthermore, even when the true roots of $q_n(t)$ are different, as long as two of them are close to each other the estimated roots may become complex if $\mathbf{c}_n$ is computed from noisy data points. In this case, one cannot proceed with the rest of the algorithm (solving the $K-2$ linear systems), because it assumes that the roots of $q_n(t)$ are real and non-repeated. One may be tempted to choose the real part of such complex solutions when they occur, however this leads to the degenerate case of repeated roots described before. Alternatively, since choosing the last two variables to build the polynomial $q_n(t)$ is arbitrary, one could try choosing a different pair of variables such that the corresponding roots are real and distinct. However, we saw in example 5, that the are polynomials that are factorable, yet every pair of variables has repeated roots. Furthermore, even if there was a pair of variables such that the associated univariate polynomial $q_n(t)$ had non-repeated roots, it is not clear how to choose such a pair without considering all possible pairwise combinations, because the $\mathbf{b}_i$'s are unknown

In this section, we propose a new algorithm for estimating the normal vectors $\{\mathbf{b}_i\}_{i=1}^n$ which is based on *polynomial differentiation* rather than *polynomial factorization*. We show that given $\mathbf{c}_n$ one can recover $\{\mathbf{b}_i\}_{i=1}^n$ by evaluating the derivatives of $p_n(\mathbf{x})$ at points on the hyperplanes. Therefore, the polynomial differentiation algorithm does not the have problems of complex roots or degenerate configuration present in the PFA. More specifically, the polynomial differentiation algorithm (PDA) consists of the following two steps:

1. Compute the number of hyperplanes n and the vector of coefficients $\mathbf{c}_n$ from the linear system $L_n\mathbf{c}_n = 0$, as described in Section 3.1.

2. Compute the normal vectors $\{\mathbf{b}_i\}_{i=1}^n$ as the derivative of $p_n(\mathbf{x})$ evaluated at the n points $\{\mathbf{y}_i \in S_i\}_{i=1}^n$, with each point lying on only one of the hyperplanes.

Therefore, the problem of clustering hyperplanes will be reduced to first finding one point per hyperplane, and then evaluating the derivative of $p_n(\mathbf{x})$, as we describe below.

3.3.1 Obtaining normal vectors by differentiation

Imagine, for the time being, that we were given a set of n points $\{\mathbf{y}_i\}_{i=1}^n$, each one lying on only one of the n hyperplanes, that is $\mathbf{y}_i \in S_i$ for $i = 1, \ldots, n$. This corresponds to a particular supervised learning setting in which we are given only *one example per cluster*. Now let us consider the derivative of $p_n(\mathbf{x})$ evaluated at each $\mathbf{y}_i$. We have:

$$Dp_n(\mathbf{x}) = \frac{\partial p_n(\mathbf{x})}{\partial \mathbf{x}} = \frac{\partial}{\partial \mathbf{x}} \prod_{i=1}^{n} (\mathbf{b}_i^T \mathbf{x}) = \sum_{i=1}^{n} (\mathbf{b}_i) \prod_{\ell \neq i} (\mathbf{b}_\ell^T \mathbf{x}). \tag{37}$$

Because $\prod_{\ell \neq i} (\mathbf{b}_\ell^T \mathbf{y}_j) = 0$ for $j \neq i$, one can obtain each one of the normal vectors as

$$\boxed{\mathbf{b}_i = \frac{Dp_n(\mathbf{y}_i)}{\|Dp_n(\mathbf{y}_i)\|} \quad i = 1, \ldots, n.} \tag{38}$$

Therefore, in the supervised learning setting in which we know one point in each one of the hyperplanes, the clustering problem can be solved *analytically* by simply evaluating the partials of $p_n(\mathbf{x})$ at each one the points with known labels.

Let us now consider the unsupervised learning scenario in which we do not know the membership of any of the data points. We first present an algebraic algorithm for finding one point in each one of the hyperplanes, based on intersecting a random line with each one of the hyperplanes. We then present a simple algorithm that finds one point in each hyperplane from the points in the dataset that minimize a certain distance function.

3.3.2 Obtaining one point per hyperplane: an algebraic solution

Consider a random line $\mathcal{L} \doteq \{t\mathbf{v} + \mathbf{x}_0,\ t \in \mathbb{R}\}$ with direction $\mathbf{v}$ and base point $\mathbf{x}_0$. We can always obtain one point in each hyperplane by intersecting $\mathcal{L}$ with the union of all the hyperplanes, except when the chosen line is parallel to one of the hyperplanes, which corresponds to a zero-measure set of lines. Since at the intersection points we must have $p_n(t\mathbf{v} + \mathbf{x}_0) = 0$, the n points $\{\mathbf{y}_i\}_{i=1}^n$ can be obtained as

$$\boxed{\mathbf{y}_i = t_i\mathbf{v} + \mathbf{x}_0 \quad i = 1, \ldots, n,} \tag{39}$$

where $\{t_i\}_{i=1}^n$ are the roots of the univariate polynomial of degree n

$$q_n(t) = p_n(tv + x_0) = \prod_{i=1}^{n} (tb_i^T v + b_i^T x_0). \tag{40}$$

The problem is now how to choose v and x_0. In the absence of noise, one can choose a line at random, because the set of lines that intersect a collection of n hyperplanes into n distinct points is an open set. However, there is a zero measure set of lines for which the roots of $q_n(t)$ are not real and distinct. For example, if $x_0 = 0$ or if x_0 is parallel to v, then the number of roots is either one or infinity. These two cases can be obviously avoided. Another degenerate configuration happens when the direction v is parallel to one of the hyperplanes. In this case the polynomial $q_n(t)$ has less than n roots, because at least one of them is at infinity. Since v is parallel to one of the hyperplanes if and only if $b_i^T v = 0$ for some $i = 1, \ldots, n$, this degenerate case can be avoided by choosing v such that $p_n(v) \neq 0$. Therefore, in the absence of noise, we randomly choose x_0 and v on the unit sphere and if the above conditions are met, we proceed with the computation of t_i, y_i and b_i, else we randomly choose a new line. Of course, in the presence of noise, different choices of $\mathcal{L}$ would give rise to different normal vectors. In order to make the process more robust, we choose multiple lines $\{\mathcal{L}_\ell\}_{\ell=1}^m$ and compute the set of normal vectors $\{b_{i\ell}\}$ corresponding to each line. For each set of normal vectors we reconstruct their corresponding collection of hyperplanes $\{S_{i\ell}\}$, and then project each data point in X onto the closest hyperplane. We then choose the set of subspaces that gives the smallest reconstruction error. In our experiments, choosing $m = 3$ random lines was enough to obtain a small reconstruction error.

Remark 12 (Connection with PFA) *Notice that the first step of the PFA (solving for the roots of a univariate polynomial) is a special case of the above algorithm in which the line $\mathcal{L}$ is chosen as $x_0 = [0, \ldots, 0, 0, 1]^T$ and $v = [0, \ldots, 0, 1, 0]^T$. Therefore, we can summarize together the PFA discussed in the preceding section and the PDA described in this section in the following algorithm.*

Algorithm 2 (PFA and Algebraic PDA for Mixtures of Hyperplanes)

solve $L_n c_n = 0$;

set $p_n(x) = c_n^T \nu_n(x)$;

compute the n roots $t_1, \ldots, t_n$ of the univariate polynomial $q_n(t) = p_n(tv + x_0)$ with:

- PFA: $x_0 = [0, \ldots, 0, 0, 1]^T$ and $v = [0, \ldots, 0, 1, 0]^T$;
- PDA: x_0 and v chosen randomly;

obtain the hyperplane normal vectors b_i:

- PFA: solve $K - 2$ linear systems of equations to find the normal vectors b_i;
- PDA: differentiate $p_n(x)$ to obtain $b_i = \frac{Dp_n(y_i)}{\|Dp_n(y_i)\|}$ at $y_i = x_0 + vt_i$.

3.3.3 Obtaining one point per hyperplane: a recursive solution

As we will see in Section 5, the technique of intersecting a line with each one of the hyperplanes does not generalize to subspaces of arbitrary dimensions. We therefore propose an alternative algorithm for computing one point per hyperplane. The idea is that we can always choose a point y_n lying on one of the hyperplanes by checking that $p_n(y_n) = 0$. Since we are given a set of data points $X = \{x^j\}_{j=1}^N$ lying on the hyperplanes, in principle we can choose y_n to be any of the data points. However, in the presence of noise and outliers a random choice of y_n may be far from the true hyperplanes. Another possibility is to choose y_n as the point in X that minimizes $|p_n(x)|$. However, the above choice has the following problems in the presence of noise:

1. The value $|p_n(x)|$ is merely an *algebraic* error, i.e., it does not really represent the *geometric* distance from x to the closest subspace. Furthermore, notice that finding the geometric distance to each subspace is in principle hard, because we do not know the normal vectors $\{b_i\}_{i=1}^n$.

2. Points x lying close to the intersection of two or more subspaces are more likely to be chosen, because two or more factors in $p_n(x) = (b_1^T x) \cdots (b_n^T x)$ are approximately zero, which yields a smaller value for $|p_n(x)|$. Furthermore, since $Dp_n(x) = 0$ for x in the intersection of two or more subspaces, one should avoid choosing points close to the intersections, because they will give very noisy estimates of the normal vectors. In fact, we can see from (37) that for arbitrary x the vector $Dp_n(x)$ is a linear combination of the normal vectors $\{b_i\}_{i=1}^n$. Thus if x is close to two subspaces the derivative will be a linear combination of both normals.

It turns out that one can avoid both of these problems thanks to the following lemma.

Lemma 1 *Let $\tilde{\boldsymbol{x}} \in S_i$ be the projection of a point $\boldsymbol{x} \in \mathbb{R}^K$ onto its closest hyperplane S_i. Then the Euclidean distance from $\boldsymbol{x}$ to S_i is given by*

$$\|\boldsymbol{x} - \tilde{\boldsymbol{x}}\| = n \frac{|p_n(\boldsymbol{x})|}{\|Dp_n(\boldsymbol{x})\|} + O(\|\boldsymbol{x} - \tilde{\boldsymbol{x}}\|^2). \tag{41}$$

Proof. Replace $m = 1$ in the proof of Lemma 2. ∎

The importance of Lemma 1 is that it allows us to compute a first order approximation of the distance from each point in $\boldsymbol{X}$ to its closest hyperplane without having to first compute the normal vectors. In fact the geometric distance (41) depends only on the polynomial $p_n(\boldsymbol{x})$ and is obtained by normalizing the algebraic error $|p_n(\boldsymbol{x})|$ by the norm of the derivative $\|Dp_n(\boldsymbol{x})\|$. Therefore, we can use this geometric distance to choose a point in the data set close to one of the subspaces as:

$$\boldsymbol{y}_n = \underset{\boldsymbol{x} \in \boldsymbol{X} : Dp_n(\boldsymbol{x}) \neq 0}{\arg\min} \frac{|p_n(\boldsymbol{x})|}{\|Dp_n(\boldsymbol{x})\|}, \tag{42}$$

and then compute the normal vector at $\boldsymbol{y}_n$ as $\boldsymbol{b}_n = Dp_n(\boldsymbol{y}_n)/\|Dp_n(\boldsymbol{y}_n)\|$. Notice that points $\boldsymbol{x}$ close to the intersection of two or more hyperplanes are immediately avoided, because $Dp_n(\boldsymbol{x}) \approx 0$.

In order to find a point $\boldsymbol{y}_{n-1}$ in one of the other $(n-1)$ hyperplanes, we could just remove the points on the subspace $S_n = \{\boldsymbol{x} : \boldsymbol{b}_n^T \boldsymbol{x} = 0\}$ from $\boldsymbol{X}$ and compute $\boldsymbol{y}_{n-1}$ similarly to (42), but minimizing over $\boldsymbol{X} \setminus S_n$. However, in the presence of noise we would have to choose a threshold in order to determine which points correspond to S_n, and the algorithm would depend on the choice of such a threshold. Alternatively, we notice that a point $\boldsymbol{x}$ lying on one of the other $(n-1)$ hyperplanes should satisfy

$$p_{n-1}(\boldsymbol{x}) \doteq p_n(\boldsymbol{x})/(\boldsymbol{b}_n^T \boldsymbol{x}) = (\boldsymbol{b}_1^T \boldsymbol{x}) \cdots (\boldsymbol{b}_{n-1}^T \boldsymbol{x}) = 0. \tag{43}$$

Therefore, similarly to (42), we can choose a point on (close to) $\cup_{i=1}^{n-1} S_i$ as the point in the data set that minimizes $|p_{n-1}(\boldsymbol{x})|/\|Dp_{n-1}(\boldsymbol{x})\|$. By applying the same reasoning to the remaining hyperplanes, we obtain the following recursive *polynomial differentiation algorithm* (PDA-rec) for finding one point per hyperplane and computing the normal vectors.

Algorithm 3 (Polynomial Differentiation Algorithm (PDA) for Mixtures of Hyperplanes)

solve $L_n \boldsymbol{c}_n = 0$;
set $p_n(\boldsymbol{x}) = \boldsymbol{c}_n^T \nu_n(\boldsymbol{x})$;
for $i = n : 1$,

$$\boldsymbol{y}_i = \underset{\boldsymbol{x} \in \boldsymbol{X} : Dp_i(\boldsymbol{x}) \neq 0}{\arg\min} \frac{|p_i(\boldsymbol{x})|}{\|Dp_i(\boldsymbol{x})\|}, \tag{44}$$

$$\boldsymbol{b}_i = \frac{Dp_i(\boldsymbol{y}_i)}{\|Dp_i(\boldsymbol{y}_i)\|}, \tag{45}$$

$$p_{i-1}(\boldsymbol{x}) = \frac{p_i(\boldsymbol{x})}{\boldsymbol{b}_i^T \boldsymbol{x}}, \tag{46}$$

end;
assign point $\boldsymbol{x}^j$ to subspace S_i if $i = \arg\min_{\ell=1,\ldots,n} |\boldsymbol{b}_\ell^T \boldsymbol{x}^j|$.

Remark 13 (Polynomial division) *Notice that the last step of the PDA is to divide $p_i(\boldsymbol{x})$ by $\boldsymbol{b}_i^T \boldsymbol{x}$ to obtain $p_{i-1}(\boldsymbol{x})$. Given the vector of coefficients of $p_i(\boldsymbol{x})$, $\boldsymbol{c}_i \in \mathbb{R}^{M_i}$, and the normal vector $\boldsymbol{b}_i \in \mathbb{R}^K$, solving for the vector of coefficients of $p_{i-1}(\boldsymbol{x})$, $\boldsymbol{c}_{i-1} \in \mathbb{R}^{M_{i-1}}$, is simply a* linear *problem of the form $\mathcal{D}_i(\boldsymbol{b}_i)\boldsymbol{c}_{i-1} = \boldsymbol{c}_i$, with $\mathcal{D}_i(\boldsymbol{b}_i) \in \mathbb{R}^{M_i \times M_{i-1}}$.*

Remark 14 *Notice that one can avoid computing $p_i(\boldsymbol{x})$ in each step of the PDA and choose $\boldsymbol{y}_{i-1}$ by a heuristic distance function (therefore not optimal). Since a point in $\cup_{\ell=i}^{n} S_i$ must satisfy $(\boldsymbol{b}_i^T \boldsymbol{x}) \cdots (\boldsymbol{b}_n^T \boldsymbol{x}) = 0$, we can choose a point $\boldsymbol{y}_{i-1}$ in $\cup_{\ell=1}^{i-1} S_i$ by as*

$$\boldsymbol{y}_{i-1} = \underset{\boldsymbol{x} \in \boldsymbol{X} : Dp_n(\boldsymbol{x}) \neq 0}{\arg\min} \frac{\frac{|p_n(\boldsymbol{x})|}{\|Dp_n(\boldsymbol{x})\|} + \delta}{|(\boldsymbol{b}_i^T \boldsymbol{x}) \cdots (\boldsymbol{b}_n^T \boldsymbol{x})| + \delta}, \tag{47}$$

where we add a small positive number $\delta > 0$ to both the numerator and denominator in order to avoid the case in which both of them are zero (e.g. with perfect data).

Remark 15 (Merging PDA-alg and PDA-rec) *Notice that, in the presence of noise, PDA-rec finds points that are close to but not necessarily in the hyperplanes. To resolve this problem, we may add an additional computation from the first step of PDA-alg to each iteration of PDA-rec as follows: First choose $\mathbf{y}_i$ and obtain $\mathbf{b}_i$ from $D\mathbf{p}_i(\mathbf{y}_i)$ according to PDA-rec; then set $\mathbf{x}_0 = \mathbf{y}_i$ and $\mathbf{v} = \mathbf{b}_i$ and solve for the roots of $q_n(t) = p_n(t\mathbf{v} + \mathbf{x}_0)$. Choose the root $t^* = \min(|t_i|)$ and obtain a new point lying on one of the hyperplanes as $\mathbf{y}_i \leftarrow t^*\mathbf{v} + \mathbf{x}_0$.*

4 Estimating a mixture of subspaces of equal dimension $k < K$

We showed in Section 2 that estimating a collection of subspaces of arbitrary dimensions is equivalent to estimating and factoring a collection of homogeneous polynomials from sample data points. However, we also showed that in general one can only recover a basis for those factorable polynomials, and that each element in the basis may not be factorable.

In Section 3 we considered the particular case of data lying on hyperplanes, and showed that in this case there is a single polynomial representing the data, which is automatically factorable.

In this section, we extend the results of Section 3 to the case of subspaces of equal dimension $0 < k_1 = \cdots = k_n = k < K$. In Section 4.1 we show that if the dimension of the subspaces k is known, then it is possible to recover a factorable polynomial by first projecting the data onto a generic $(k+1)$-dimensional subspace of $\mathbb{R}^K$. Since in practice the dimension of the subspaces could be unknown, in Section 4.2 we derive rank constraints on the data matrix that allow us to simultaneously estimate the number of subspaces n and their dimension k. Given n and k, in Section 4.3 we present two algorithms for recovering the subspaces. The first one uses a single projection followed by either PFA or PDA to segment the data, and then obtains a basis for each one of the original subspaces by applying PCA to the segmented data. The second one uses multiple projections followed by a generalization of PDA that deals with multiple polynomials.

4.1 Projecting samples onto a $(k+1)$-dimensional subspace

In this section, we show that the segmentation of a sample set X drawn from n k-dimensional subspaces of a space of dimension $K > k$ is preserved after projecting the sample set X onto a *generic* subspace Q of dimension $k+1$ ($\leq K$). An example is shown in Figure 2, where two lines L_1 and L_2 in $\mathbb{R}^3$ are projected onto a plane Q not orthogonal to the plane containing the lines.

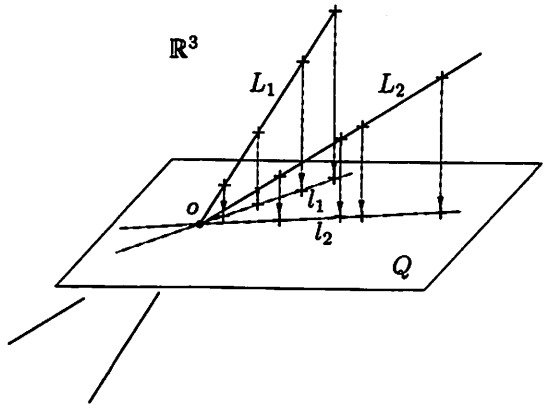


Figure 2: Two 1-dimensional subspaces L_1, L_2 in $\mathbb{R}^3$ projected onto a 2-dimensional plane Q. Clearly, the membership of each sample (labeled as "+"on the lines) is preserved through the projection.

More generally, let us denote the projection onto a $(k+1)$-dimensional subspace Q as

$$\pi_Q : \mathbb{R}^K \to Q \qquad \mathbf{x} \mapsto \mathbf{x}', \tag{48}$$

and the projection of S_i as $S_i' \doteq \pi_Q(S_i)$. Also let $X' \doteq \pi_Q(X)$ be the set of projected data points lying on the collection of projected subspaces $Z' \doteq \pi_Q(Z) = \cup_{i=1}^n S_i'$.

From a geometric point of view, we notice that if the subspace Q is in *general position*[11], then $\dim(S_i')$ remains

[11] As defined by the transversality conditions in footnotes 12 and 13.

to be k – no reduction in the dimension of each subspace[12], and there is a one-to-one correspondence between S_i' and S_i – no reduction in the number of subspaces[13] n. In other words, if the subspace Q is in general position, then segmenting the original collection of subspaces $Z = \cup_{i=1}^{n} S_i$ is equivalent to segmenting the collection of projected subspaces $Z' = \cup_{i=1}^{n} S_i'$.

The effect of such a projection can also be explained from an algebraic viewpoint. As we discussed in Sections 2, when $0 < k < K-1$ determining the ideal $I(Z)$ requires the identification of all its generators. However, after projecting the data onto the subspace Q the ideal $I(Z')$ becomes a principal ideal which is generated by a unique homogeneous polynomial $p_n'(\boldsymbol{x}')$, as demonstrated in Section 3. Therefore, when k is known, identifying Z is equivalent to identifying $p_n'(\boldsymbol{x}')$, and the GPCA problem for subspaces of equal dimension k, where $0 < k < K$, can always be reduced to the case of hyperplanes. Furthermore, since $p_n'(\boldsymbol{x}') \in I(Z')$ is factorable and $\boldsymbol{x}' = \pi_Q(\boldsymbol{x})$, then $p_n(\boldsymbol{x}) = p_n'(\boldsymbol{x}') \in I(Z)$ is also factorable. Therefore, each projection onto a $(k+1)$-dimensional subspace Q produces a factorable polynomial $p_n(\boldsymbol{x}) \in I(Z)$. Thus, we can obtain a basis of factorable polynomials of degree n in $I(Z)$ by choosing a large enough collection of projections $\{\pi_Q\}$.

The projection of samples onto a $(k+1)$-dimensional space also reveals an interesting *duality* between the two cases $\dim(S_i) = k$ and $\dim(S_i) = K - k$. If S_i is a 1-dimensional subspace of $\mathbb{R}^K$, i.e., a line through the origin, then for every sample point $\boldsymbol{x} \in S_i$ we can choose $K-1$ vectors $\{\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_{K-1}\}$ which together with $\boldsymbol{x}$ form an orthogonal basis of $\mathbb{R}^K$. Then each point $\boldsymbol{y}_j$ lies in the subspace $S_i^\perp$ orthogonal to S_i, which we simply denote as the *co-subspace* of S_i. Thus, the problem of segmenting samples from n 1-dimensional subspaces $Z = \cup_{i=1}^{n} S_i$ is equivalent one of segmenting a corresponding set of *co-samples* from n $(K-1)$-dimensional co-subspaces $\cup_{i=1}^{n} S_i^\perp$. At first sight, this construction of duality does not apply to subspaces S_i of dimension $k > 1$, because it is impossible to compute co-samples $\boldsymbol{y} \in S^\perp$ associated to a sample $\boldsymbol{x} \in S_i$ without knowing S_i. However, if we apply one of the GPCA algorithms in Section 3 (PFA or PDA) to the projected data $\boldsymbol{X}'$, we obtain a collection of vectors $\{\boldsymbol{b}_i' \in \mathbb{R}^{k+1}\}_{i=1}^{n}$ normal to the subspaces $\{S_i' \subset Q\}_{i=1}^{n}$ in Q, respectively. If we now embed each vector $\boldsymbol{b}_i'$ back into the space $\mathbb{R}^K$ through the inclusion $\iota_Q : Q \to \mathbb{R}^K$ and call $\boldsymbol{b}_i = \iota(\boldsymbol{b}_i')$, then we have $\boldsymbol{b}_i \perp S_i'$ and $\boldsymbol{b}_i \perp Q^\perp$, thus $\boldsymbol{b}_i \perp S_i$ is a vector orthogonal to the original subspace S_i. The overall process can be summarized by the following diagram:

$$\{\boldsymbol{x} \in \cup S_i\} \xrightarrow{\pi_Q} \{\boldsymbol{x}' \in \cup S_i'\} \underset{PDA}{\overset{PFA}{\Longrightarrow}} \{\boldsymbol{b}' \in \cup S_i'^{\perp}\} \xrightarrow{\iota_P} \{\boldsymbol{b} \in \cup S_i^{\perp}\}. \tag{49}$$

Through this process, a different choice for the subspace Q will give rise to a different set of vectors $\{\boldsymbol{b}\}$ in the co-subspaces $\cup S_i^\perp$. The more projection subspaces Q we use, the more co-samples we draw from these co-subspaces. Notice that if we do not segment the data right after we obtain the normal vectors $\{\boldsymbol{b}_i'\}_{i=1}^{n}$ from each Q, we will not know the co-subspace $S_i^\perp$ with which each normal vector $\boldsymbol{b}$ is associated. Therefore, we will be facing exactly the *dual* problem to the original GPCA problem: Segmentation of samples $\{\boldsymbol{x}\}$ drawn from the subspaces $\cup S_i$ versus segmentation of (induced) co-samples $\{\boldsymbol{b}\}$ drawn from the co-subspaces $\cup S_i^\perp$. Therefore, the two cases $\dim(S_i) = k$ and $\dim(S_i) = K - k$ are indeed *dual* to each other, hence computationally equivalent.

4.2 Estimating the number of subspaces n and their dimension k

In order to be able to project samples onto a $(k+1)$-dimensional space, we need to know the dimension of the original subspaces k. If we were estimating a single subspace, as in standard PCA we could obtain k directly as the rank of the data matrix [11]. However, since we are studying the case of n subspaces, whenever k is unknown we need to know how to compute it from data. In this section, we study under what conditions the problem of recovering n and k from data is well-posed, and derive rank conditions on the data from which one can estimate n and k.

First of all, we notice that a simultaneous recovery of n and k may be ambiguous if we are not clear about what we are asking for. For example, in the extreme cases, one may interpret the sample set $\boldsymbol{X}$ as N 1-dimensional subspaces, with each subspace spanned by each one of the sample points $\boldsymbol{x} \in \boldsymbol{X}$, or one may view the whole $\boldsymbol{X}$ as belonging to one K-dimensional subspace, i.e., $\mathbb{R}^K$ itself. Besides these two trivial interpretations, ambiguity may also arise in cases such as that of Figure 3, in which a collection of lines can also be interpreted as a collection of planes.

A formal way of resolving such ambiguous interpretations in the absence of noise is by looking at the algebraic structure of the GPCA problem. We notice that the sample points are drawn from a collection of subspaces $\{S_i\}_{i=1}^{n}$, which can always be interpreted as an *algebraic set* $Z = \cup_{i=1}^{n} S_i$ generated by irreducible subsets S_i's (irreducible algebraic sets are also called *varieties*). The decomposition of Z into $\{S_i\}_{i=1}^{n}$ is always unique [9]. Therefore, the $k = \dim(S_i)$ and the number of subspaces n are always uniquely defined in a purely algebraic fashion. In this sense,

[12]This requires that Q be transversal to each $S_i^\perp$, i.e., $\text{span}\{Q, S_i^\perp\} = \mathbb{R}^K$ for $i = 1, 2, \ldots, n$. Since n is finite, this transversality condition can be easily satisfied. Furthermore, the set of positions for Q which violate the transversality condition is only a zero-measure closed set [10].

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

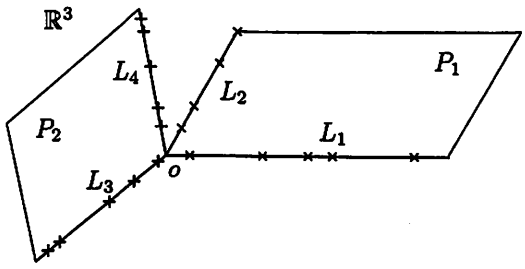


Figure 3: A set of samples that can be interpreted as coming either from four 1-dimensional subspaces L_1, L_2, L_3, L_4 in $\mathbb{R}^3$, or from two 2-dimensional subspaces P_1, P_2 in $\mathbb{R}^3$.

for the case shown in Figure 3, the first interpretation (4 lines) would be the right one and the second one (2 planes) would be incorrect since, e.g., $L_1 \cup L_2$ is not an irreducible algebraic set.

Having established that the problem of simultaneously estimating n and k is well-posed, we are left with deriving an actual formula to compute them. We consider the following three cases.

4.2.1 Case 1: k known

Imagine for a moment that k was known, and that we wanted to compute n only. Since k is known, we can first project the data onto a $(k+1)$-dimensional space and then form the matrix $L_i(k+1)$ in (9) by applying the Veronese map of degree $i = 1, 2, \ldots$ to the projected data. From our analysis in Sections 2 and 4.1, there is a unique polynomial of degree n generating $I(Z')$ whose coefficients are in the null space of $L_n(k+1)$. Thus $\text{rank}(L_n(k+1)) = M_n(k+1) - 1$. Furthermore, there cannot be a polynomial of lower degree that is satisfied by all the data, hence $\text{rank}(L_i(k+1)) = M_i(k+1)$ for $i < n$. Similarly, there are infinitely many polynomials of degree more than n that are satisfied by all the data, namely any multiple of $p_n(x)$. Therefore, $\text{rank}(L_i(k+1)) < M_i(k+1) - 1$ for $i > n$. Consequently, if k is known and a generic set of $N \geq M_n - 1$ sample points are given, we can compute n by first projecting the data onto a $(k+1)$-dimensional space and then setting

$$\boxed{n = \min\{i : \text{rank}(L_i(k+1)) = M_i(k+1) - 1\}.} \tag{50}$$

4.2.2 Case 2: n known

Consider now the opposite case in which n is known, but k is unknown. Let $L_n(\ell+1)$ be defined as in (9), but computed from the data projected onto an $(\ell+1)$-dimensional subspace. When $\ell < k$, we have a collection of $(\ell+1)$-dimensional subspaces in a $(\ell+1)$-dimensional space, which implies that $L_n(\ell+1)$ is full rank. If $\ell = k$, then from (54) we have that $\text{rank}(L_n(\ell+1)) = M_n(\ell+1) - 1$. When $\ell > k$, then equation (9) has more than one solution, thus $\text{rank}(L_n(\ell+1)) < M_n(\ell+1) - 1$. Therefore, if n is known, we can compute k as

$$\boxed{k = \min\{\ell : \text{rank}(L_n(\ell+1)) = M_n(\ell+1) - 1\}.} \tag{51}$$

4.2.3 Case 3: n and k unknown

We are left with the case in which both n and k are unknown. Let $L_i(\ell+1)$ be defined as in (9), but computed by applying the Veronese map of degree i to the data projected onto an $(\ell+1)$-dimensional subspace. As before, if $\ell < k$ then $L_i(\ell+1)$ is full rank for all i. When $\ell = k$, $L_i(\ell+1)$ is full rank for $i < n$, drops rank by one if $i = n$ and drops rank by more than one if $i > n$. Thus one can set k to be the smallest integer ℓ for which there exist an i such that $L_i(\ell+1)$ drops rank, that is

$$\boxed{k = \min\{\ell : \exists i \geq 1 \text{ such that } \text{rank}(L_i(\ell+1)) < M_i(\ell+1)\}.} \tag{52}$$

Given k one can compute n as in equation (54).

More formally, we have shown the following.

Theorem 3 (Number of subspaces n and their dimension k) *Assume that a collection of $N \geq M_n(k+1) - 1$ sample points $\{x^j\}_{j=1}^N$ on n different k-dimensional subspaces of $\mathbb{R}^K$ is given. Let $L_i(\ell+1) \in \mathbb{R}^{N \times M_i(k+1)}$ be the matrix defined in (9), but computed with the Veronese map ν_i of degree i applied to the data projected onto a generic*

$(\ell+1)$-dimensional subspace of $\mathbb{R}^K$. If the sample points are in general position and at least k points correspond to each subspace, then the dimension of the subspaces k can be obtained as:

$$\boxed{k = \min\{\ell : \exists i \geq 1 \text{ such that } \text{rank}(L_i(\ell+1)) < M_i(\ell+1)\},} \tag{53}$$

and the number n of subspaces is given by:

$$\boxed{n = \min\{i : \text{rank}(L_i(k+1)) = M_i(k+1) - 1\}.} \tag{54}$$

Proof. Since the theorem deals with the case of data projected onto $\mathbb{R}^{K'}$, with $K' = k+1$, and the projected data points live on hyperplanes, equation (54) is a direct consequence of Theorem 1. The rest of the theorem follows from the analysis given in this section. ∎

Corollary 1 *The vector of coefficients $c_n \in \mathbb{R}^{M_n(k+1)}$ of the homogeneous polynomial $p_n(x)$ can be uniquely determined (up to a scale factor) as the kernel of the matrix $L_n(k+1) \in \mathbb{R}^{N \times M_n(k+1)}$ from at least $M_n(k+1) - 1$ points on the subspaces, with at least k points on each subspace.*

Remark 16 *The above statement indirectly claims that in order to* linearly *estimate the polynomial $p_n(x)$, one needs as many sample points as the dimension of the feature space. It is therefore unclear whether one could apply the* kernel trick *to reduce the dimensionality of the problem.*

Remark 17 *Although we have derived rank conditions on the data from which n and k can be estimated, in practice this requires to search for up to possibly $(K-1)$ values for k and $\lceil N/(K-1) \rceil$ values for n. The problem becomes even harder in the presence of noise, since one needs to threshold the singular values of $L_i(\ell+1)$ to determine its rank (see Remark 9). In our experience, the rank conditions work well when either k or n are known. It remains open to find a good search strategy for n and k when both of them are unknown.*

4.3 Estimating the subspaces: the polynomial differentiation algorithm (PDA)

As we discussed in Section 4.1, when $k < K-1$, there are both geometric and algebraic reasons that suggest that we should first project the sample set X onto a $(k+1)$-dimensional subspace, say Q, of $\mathbb{R}^K$. In many practical applications the dimension of the subspaces k is small compared to the dimensionality of the data K. Therefore, we can choose Q so that the variance of the projected data is maximized, which is equivalent to choosing Q as the subspace spanned by the first $k+1$ principal components of the data. Given the projected data, we can apply one of the GPCA algorithms given in Section 3 (PFA or PDA) to obtain a normal vector to each one of the projected subspaces restricted to Q. In the absence of noise, we can use the (projected) normals to segment the projected data points, which automatically induces a segmentation of the original data into different groups. Given the segmentation of the data, we can estimate a basis for the original subspaces in $\mathbb{R}^K$ by applying PCA to each group. This leads to the following algorithm for estimating subspaces of equal dimension based on a single projection computed using PCA.

Algorithm 4 (PCA-GPCA Algorithm for Mixtures of Subspaces of Equal Dimension $k < K-1$)

1. Obtain the number of subspaces n and their dimension k as in Theorem 3.
2. Apply PCA with $k+1$ principal components to the original data points $[x^1, \ldots, x^N] \in \mathbb{R}^{K \times N}$ to obtain the projected data points $[x'^1, \ldots, x'^N] \in \mathbb{R}^{(k+1) \times N}$.
3. Apply GPCA for hyperplanes (PFA or PDA) to the projected data $[x'_1, \ldots, x'_N] \in \mathbb{R}^{k+1 \times N}$ to obtain a collection of normal vectors $\{b'_i \in \mathbb{R}^{k+1}\}_{i=1}^n$.
4. Cluster the original data by assigning point x^j to subspace S_i if
$$i = \arg \min_{\ell=1,\ldots,n} |b'^T_\ell x'^j|. \tag{55}$$
5. Obtain a basis for S_i by applying PCA to the points in S_i, for $i = 1, \ldots, n$.

However, it is important to notice that Algorithm 4 can fail to give the correct segmentation. For example, consider the case of data in $\mathbb{R}^3$ lying on $n = 3$ lines along the x, y and z axis. If the data is such that the covariance matrix is the identity, then in the first step of the algorithm we could obtain the $x - y$ plane as the two principal components. Hence the segmentation of the data would not be preserved, because one of the lines (the z-axis) is projected onto the origin. Even though cases like this are rare,[14] in the presence of noise one could choose a projection Q that is close to a degenerate configuration, thus affecting the quality of the segmentation. Furthermore, the algorithm also has the disadvantage of having to cluster the data before estimating a basis for each subspace, which further increases the dependency of its performance on the choice of Q.

In the rest of the section, we propose an alternative solution that uses multiple randomly chosen projections. The algorithm is a generalization of the polynomial differentiation algorithm (PDA) that uses multiple randomly chosen projections to determine a basis for each subspace.

Let $\{Q^\ell\}_{\ell=1}^m$ be a collection of m randomly chosen $(k+1)$-dimensional subspaces of $\mathbb{R}^K$. For each ℓ, let $p'_{n\ell}(x')$ be the polynomial representing the collection of projected subspaces $\{S'_i = \pi_{Q^\ell}(S_i)\}_{i=1}^n$, and let $p_{n\ell}(x) = p'_{n\ell}(\pi_{Q^\ell}(x))$ be its corresponding polynomial in $I(Z)$. Then, from the analysis of Section 3.3 we have that if $y_i \in S_i$ then

$$\left.\frac{\partial p_{n\ell}(x)}{\partial x}\right|_{x=y_i} \in S_i^\perp \text{ for all } \ell = 1, \ldots, m. \tag{56}$$

In other words, if we are given a collection of n points $\{y_i \in S_i\}$, with each point lying on only one of the subspaces, then we can estimate a collection of vectors normal to each one of the subspaces from the partial derivatives of the m polynomials $\{p_{n\ell}(x)\}_{\ell=1}^m$. The question is now how to obtain a collection of n points $\{y_i \in S_i\}_{i=1}^n$, each one lying on only one of the subspaces. As in the case of hyperplanes, we can choose points x in the data set X that minimize a certain distance from x to its closest subspace. The following lemma tells us how to compute such a distance.

Lemma 2 *The Euclidean distance from point x to its closest subspace is given by*

$$\|x - \tilde{x}\| = n\sqrt{P_n(x)\big(DP_n(x)^T DP_n(x)\big)^\dagger P_n(x)^T} + O\big(\|x - \tilde{x}\|^2\big), \tag{57}$$

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

Proof. The projection $\tilde{x}$ of a point x onto the zero set of the polynomials $\{p_{n\ell}\}_{\ell=1}^m$ can be obtained as the solution of the following constrained optimization problem

$$\begin{array}{ll} \min & \|\tilde{x} - x\|^2 \\ \text{subject to} & p_{n\ell}(\tilde{x}) = 0 \quad \ell = 1, \ldots, m. \end{array} \tag{58}$$

By using Lagrange multipliers $\lambda \in \mathbb{R}^m$, we can convert this problem into the unconstrained optimization problem

$$\min_{\tilde{x},\lambda} \|\tilde{x} - x\|^2 + P_n(\tilde{x})\lambda. \tag{59}$$

From the first order conditions with respect to $\tilde{x}$ we have

$$2(\tilde{x} - x) + DP_n(\tilde{x})\lambda. \tag{60}$$

After multiplying on the left by $(DP_n(\tilde{x}))^T$ and $(\tilde{x} - x)^T$, respectively, we obtain

$$\lambda = 2\big(DP_n(\tilde{x})^T DP_n(\tilde{x})\big)^\dagger DP_n(\tilde{x})^T x \tag{61}$$

$$\|\tilde{x} - x\|^2 = \frac{1}{2} x^T DP_n(\tilde{x})\lambda, \tag{62}$$

where we have used the fact that $(DP_n(\tilde{x}))^T\tilde{x} = nP_n(\tilde{x}) = 0$. After replacing (61) on (62) we obtain that the squared distance from x to its closest subspace can be expressed as

$$\|\tilde{x} - x\|^2 = x^T DP_n(\tilde{x})\big(DP_n(\tilde{x})^T DP_n(\tilde{x})\big)^\dagger DP_n(\tilde{x})^T x. \tag{63}$$

After expanding in Taylor series about $\tilde{x} = x$, and noticing that $DPn(x)^T x = nP_n(x)^T$ we obtain

$$\|\tilde{x} - x\|^2 \approx n^2 P_n(x)\big(DP_n(x)^T DP_n(x)\big)^\dagger P_n(x)^T, \tag{64}$$

[14] Recall from Section 2 that the set of subspaces Q that fail to preserve the segmentation is a zero-measure set.

which completes the proof. ∎

Thanks to Lemma 2, we can choose a point $\boldsymbol{y}_n$ in the dataset $\boldsymbol{X}$ that lies on (close to) one of the subspaces as:

$$\boldsymbol{y}_n = \underset{\boldsymbol{x} \in \boldsymbol{X}: DP_n(\boldsymbol{x}) \neq 0}{\arg\min} P_n(\boldsymbol{x}) \big(DP_n(\boldsymbol{x})^T DP_n(\boldsymbol{x})\big)^\dagger P_n(\boldsymbol{x})^T. \tag{65}$$

Given $\boldsymbol{y}_n$, we can compute the collection of normal vectors $\{\boldsymbol{b}_{n\ell} \in S_n^\perp\}_{\ell=1}^m$ from the derivatives of $p_{n\ell}(\boldsymbol{x})$ at $\boldsymbol{y}_n$. In order to find a point $\boldsymbol{y}_{n-1}$ in one of the remaining $(n-1)$ subspaces, but not in S_n, we find a new set of polynomials $\{p_{(n-1)\ell}(\boldsymbol{x})\}$ in the ideal of the algebraic set $\cup_{i=1}^{n-1} S_i$. Since the polynomial $p_{n\ell}(\boldsymbol{x})$ is factorable and one of its factors is precisely $\boldsymbol{b}_{n\ell}^T \boldsymbol{x}$, as in the case of hyperplanes, we can obtain the polynomials $\{p_{(n-1)\ell}(\boldsymbol{x})\}$ by polynomial division as

$$p_{n,\ell-1}(\boldsymbol{x}) = \frac{p_{n\ell}(\boldsymbol{x})}{\boldsymbol{b}_{n\ell}^T \boldsymbol{x}}. \tag{66}$$

By applying the same reasoning to the remaining subspaces, we obtain a set of normal vectors $\{\boldsymbol{b}_{i\ell}\}$ to each subspace S_i, $i = 1, \ldots, n$, from each projection Q^ℓ, $\ell = 1, \ldots, m$. If $m \geq K - k$ and the subspaces $\{Q^\ell\}_{\ell=1}^m$ are in general position, then we can immediately obtain a basis B_i for $S_i^\perp$ by applying PCA to the matrix of normal vectors $[\boldsymbol{b}_{n1}, \ldots, \boldsymbol{b}_{nm}] \in \mathbb{R}^{K \times m}$. If not, the matrix B_i still allows us to segment the data into different groups. Then, we can obtain a basis for S_i by applying PCA to the data points in S_i. We therefore have the following *polynomial differentiation algorithm* (PDA) for mixtures of subspaces of equal dimension $k < K - 1$.

Algorithm 5 (PDA for Mixtures of Subspaces of Equal Dimension $k < K - 1$)

obtain the number of subspaces n and their dimension k as in Theorem 3.

for $\ell = 1 : m$,

choose a $(k+1)$-dimensional subspace $Q^\ell \subset \mathbb{R}^K$;

build the data matrix $L'_{n\ell} \in \mathbb{R}^{N \times M_n(k+1)}$ from the projected data $\boldsymbol{X}' = \pi_{Q^m}(\boldsymbol{X})$;

solve for the vector of coefficients $\boldsymbol{c}'_{n\ell} \in \mathbb{R}^{M_n(k+1)}$ from $L'_{n\ell} \boldsymbol{c}'_{n\ell} = 0$;

set $p_{n\ell}(\boldsymbol{x}) = \boldsymbol{c}'^T_{n\ell} \nu_n(\pi_{Q^\ell}(\boldsymbol{x}))$;

end;

for $i = n : 1$,

do

$$P_i(\boldsymbol{x}) = [p_{i1}(\boldsymbol{x}), \ldots, p_{im}(\boldsymbol{x})] \in \mathbb{R}^{1 \times m}, \tag{67}$$

$$\boldsymbol{y}_i = \underset{\boldsymbol{x} \in \boldsymbol{X}: DP_i(\boldsymbol{x}) \neq 0}{\arg\min} P_i(\boldsymbol{x}) \big(DP_i(\boldsymbol{x})^T DP_i(\boldsymbol{x})\big)^\dagger P_i(\boldsymbol{x})^T, \tag{68}$$

$$\boldsymbol{b}_{i\ell} = \frac{Dp_{i\ell}(\boldsymbol{x})}{\|Dp_{i\ell}(\boldsymbol{x})\|}, \text{ for } \ell = 1, \ldots, m, \tag{69}$$

$$p_{i-1,\ell} = \frac{p_{i\ell}(\boldsymbol{x})}{\boldsymbol{b}_{i\ell}^T \boldsymbol{x}}, \text{ for } \ell = 1, \ldots, m, \tag{70}$$

$$B_i = PCA([\boldsymbol{b}_{i1}, \ldots, \boldsymbol{b}_{im}]) \tag{71}$$

end;

end;

assign point $\boldsymbol{x}^j$ to subspace S_i if $i = \arg\min_{\ell=1,\ldots,n} \|B_\ell^T \boldsymbol{x}^j\|$.

if $m < K - k$, **then**

obtain a basis for S_i by applying PCA to the data in S_i, for $i = 1, \ldots, n$.

end.

5 Estimating a mixture of subspaces of arbitrary dimensions $\{k_i\}_{i=1}^n$

Our analysis in Sections 3 and 4 shows that for subspaces of equal dimension $k \leq K-1$ one can always estimate a set of factorable polynomials from the data, either directly as in the case of hyperplanes where $k = K-1$, or after a suitable projection onto a $(k+1)$-dimensional subspace when $k < K-1$.

In this section, we consider the most general case in which each subspace can have a possibly different dimension, hence we cannot obtain a collection of factorable polynomials representing the data. Instead, as described in Section 2, we can only obtain a basis $\{p_{n\ell}(x)\}$ for those factorable polynomials and each element in the basis may not be factorable. In Section 5.1 we show that it is still possible to obtain a collection of normal vectors to one of the subspaces from the derivatives of the given polynomials $\{p_{n\ell}(x)\}$, even when they are not factorable. Given the normals to that subspace $\{b_{n\ell} \in S_n^\perp\}$ the rest of the problem is to divide the original polynomials by the linear forms defined by the normals in order to obtain the polynomials $p_{n-1,\ell}(x)$ defining the remaining $n-1$ subspaces. However, in this case we cannot perform polynomial division, because the given polynomials $\{p_{n\ell}(x)\}$ may not be factorable. In Section 5.2, we derive an algorithm that uses the data and the estimated normals to estimate the polynomials $\{p_{n-1,\ell}(x)\}$, without performing polynomial division.

5.1 Obtaining subspace bases by polynomial differentiation

Recall from Section 2 that given a set of points $X = \{x^j\}_{j=1}^N$ lying on a collection of subspaces $\{S_i \subset \mathbb{R}^K\}_{i=1}^n$, the algebraic set $Z = \cup_{i=1}^n S_i$ can be represented with a collection of polynomials $\{p_{n\ell}(x) = c_{n\ell}^T \nu_n(x)\}$ whose coefficients lie in the $(m = \dim(I(Z)))$-dimensional null space of the embedded data matrix $L_n \in \mathbb{R}^{N \times M_n}$, i.e., $L_n c_{n\ell} = 0$ for $\ell = 1, \ldots, m$. The GPCA problem is then equivalent to estimating a basis B_i for $S_i^\perp$, where $i = 1, \ldots, n$ from the set of not necessarily factorable polynomials $\{p_{n\ell}(x)\}_{\ell=1}^m$.

As we also hinted in Section 2, one could first try to find a change of basis for the null space of L_n that gives a set of factorable polynomials, and then apply some variation of the PDA to obtain the bases $\{B_i\}_{i=1}^n$. However, this amounts to solving a set of polynomials of degree n in several variables. Fortunately, similarly to the case of subspaces of equal dimension, we can exploit the local linear structure of the algebraic set Z to first obtain a basis for the orthogonal complement to each subspace by differentiating all the polynomials obtained from null(L_n) (factorable or not). We can do so thanks to the following (even more general) statement.

Lemma 3 *Let p be any polynomial in the ideal I of an algebraic set Z, i.e., $p(x) = 0, \forall x \in Z$. If $T_{x_0}Z$ is the (Zariski) tangent space to Z at a smooth point x_0, then the derivative of $p(x)$ at x_0 satisfies:*

$$t^T \frac{\partial p(x)}{\partial x}\Big|_{x_0} = 0, \quad \forall t \in T_{x_0}Z. \tag{72}$$

Furthermore, $(T_{x_0}Z)^\perp = \text{span}\Big\{\frac{\partial p(x)}{\partial x}\Big|_{x_0}, \forall p \in I\Big\}$.

Proof. Equation (72) is obvious since $p(x) \equiv 0$ on Z and the left hand side is merely a directional derivative along t at the regular point x_0. The fact that the derivatives span the entire normal space is the consequence of the general dimension theory for algebraic varieties [1, 8, 6]. ∎

Notice that, for a particular $p(x)$ in one of the homogeneous components, say $I_{n'}$ $(n' \geq n)$, of the ideal I, its derivative could be zero at x_0.[15] Nevertheless, if we evaluate the derivatives for all the polynomials in the ideal I, they will span the entire orthogonal complement to the tangent space. In fact, we can do better than this since, as we will show in the case with n subspaces, we only have to evaluate the derivatives for polynomials in I up to degree n.

The above lemma is particularly useful to the GPCA problem. Since the algebraic set Z that we are dealing with here is (locally) flat, the tangent space T and its orthogonal complement $T^\perp$ will be independent of the point at which they are evaluated.[16] To see this more clearly, let $\{y_i \in S_i\}_{i=1}^n$ be a set of n points each one lying on only one of the subspaces. Also let c_n be a vector in the null space of L_n. By construction, even though c_n may not correspond to a factorable polynomial, it can be written as a linear combination of vectors $c_{n\ell}$ which correspond to factorable polynomials, i.e., $c_n = \sum \alpha_\ell c_{n\ell}$. Then

$$\frac{\partial}{\partial x} c_n^T \nu_n(x)\Big|_{x=y_i} = \frac{\partial}{\partial x} \sum_\ell \alpha_\ell c_{n\ell}^T \nu_n(x)\Big|_{x=y_i} = \sum_\ell \alpha_\ell b_{i\ell}, \tag{73}$$

[15] For instance, for $g(x) \in I_n$, let $f(x) = g^2(x) \in I_{2n}$, and its derivative will be zero everywhere.

[16] For points in the same subspace S_i, T is in fact S_i itself; and $T^\perp$ is $S_i^\perp$.

where $b_{i\ell} \in S_i^\perp$ is a normal vector to subspace S_i. Therefore, although $c_n^T \nu_n(x)$ may not be factorable, its derivative at y_i still gives a vector normal to S_i. Combining this with the analysis in the preceding subsection, we have essentially proven the following theorem.

Theorem 4 (Polynomial differentiation) *For the GPCA problem, if the given sample set X is such that* $\dim(\text{null}(L_n)) = \dim(I_n)$ *and one generic point y_i is given for each subspace S_i, then we have*

$$S_i^\perp = \text{span}\Big\{ \frac{\partial}{\partial x} c_n^T \nu_n(x) \Big|_{x=y_i}, \ \forall c_n \in \text{null}(L_n) \Big\}. \tag{74}$$

Theorem 4 states a useful fact about the ideal I associated with a union of subspaces. If we know the number of subspaces n and we are given a set of points $\{y_i \in S_i\}$, then in order to obtain the bases $\{B_i\}$ we do not have to evaluate the derivatives for all polynomials in I. It suffices to evaluate the derivatives of polynomials in I_n only. Therefore, as a consequence of the theorem we already have the sketch of an algorithm for computing a basis for $S_i^\perp$ (hence for S_i), given $\{y_i\}$:

- Compute a basis for the null space $\text{null}(L_n)$ using, for example, SVD.
- Evaluate the derivative of the (possibly nonfactorable) polynomial $c_n^T \nu_n(x)$ at y_i for each c_n in the basis of $\text{null}(L_n)$ to obtain a set of normal vectors in $S_i^\perp$.
- Compute a basis for $S_i^\perp$ by applying PCA to the normal vectors obtained in step 2. PCA should automatically give the dimension of each subspace $k_i = \dim(S_i)$ as:

$$\boxed{k_i = K - \text{rank}(DP_n(y_i)), \quad i = 1, \ldots, n.} \tag{75}$$

Example 7 (The $x-y$ plane and the z axis (revisited)) *As in Example 1, let us consider the case of $n = 2$ subspaces of $\mathbb{R}^3$ of dimension* $\dim(S_1) = 2$ *and* $\dim(S_2) = 1$ *represented as:*

$$S_1 = \{x \in \mathbb{R}^3 : x_3 = 0\} \quad \text{and} \quad S_2 = \{x \in \mathbb{R}^3 : x_1 = 0 \wedge x_2 = 0\}.$$

Then we can represent $Z = S_1 \cup S_2$ as the zero set of the two polynomials

$$p_{21}(x) = x_1 x_3 \quad \text{and} \quad p_{22}(x) = x_2 x_3.$$

The derivatives of these two polynomials are:

$$Dp_{21}(x) = \begin{bmatrix} x_3 \\ 0 \\ x_1 \end{bmatrix} \quad \text{and} \quad Dp_{22}(x) = \begin{bmatrix} 0 \\ x_3 \\ x_2 \end{bmatrix},$$

which evaluated at $y_1 = (1,1,0)^T \in S_1$ and $y_2 = (0,0,1)^T \in S_2$ yield

$$DP_2(y_1) = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 1 \end{bmatrix} \quad \text{and} \quad DP_2(y_2) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}.$$

By applying PCA to $DP_2(y_1)$ and $DP_2(y_2)$ we obtain a basis for $S_1^\perp$ and $S_2^\perp$ as

$$B_1 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \quad \text{and} \quad B_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}.$$

Remark 18 (Overestimating the number of subspaces) *Notice that one may replace n with any $n' > n$ in Theorem 4 and the conclusion still holds. Therefore, at least in principle, choosing a polynomial embedding of a higher degree n' does not prevent us from correctly computing the subspaces using the same method, as long as the given samples are sufficient for us to recover a basis for $I_{n'}$ from the null space of $L_{n'}$. In practice, we should try to use the lowest possible degree to avoid the high computational cost associated with a redundant embedding.*

Remark 19 (Underestimating the number of subspaces) *Let S_1 and S_2 be the x and y axis in $\mathbb{R}^3$, respectively. Then a basis for the set of polynomials of degree two that vanish on $S_1 \cup S_2$ is $\{x_1x_2, x_1x_3, x_2x_3, x_3^2\}$. However, since the two lines lie in the $x-y$ plane, there is a polynomial of degree one in null(L_1), $p_1(x) = x_3$, that also vanishes on $S_1 \cup S_2$. Furthermore, the derivative of $p_1(x)$ at points on the lines gives a single normal vector $[0, 0, 1]^T$, which is the normal to the $x - y$ plane. This example shows that in the case of subspaces of arbitrary dimensions one may underestimate both the number of subspaces and the dimension of the orthogonal bases. As stated in Remark 2, this situation happens whenever the ideal $I(Z)$ contains polynomials of degree $d < n$. However, one may still recover the correct structure by increasing the degree of the embedding as follows: increase the degree of embedding incrementally from $i = 1$ until L_i drops rank at $i = d \leq n$; for $i \geq d$ collect the derivatives (normal vectors) at every point in the data set; stop when the derivatives no longer increase the dimension for the orthogonal complements.*

Remark 20 (Duality) *Recall from our analysis in Section 4.1 that a GPCA problem with subspaces of equal dimension $k_1 = \cdots = k_n = k$ is* dual *to a GPCA problem with $k_1 = \cdots = k_n = K - k$. It turns out that Theorem 4 allows us to generalize this duality result to subspaces of arbitrary dimensions. To this end, we notice that the derivatives of the polynomials P_n evaluated at a point x on a subspace S_i gives a basis $B_i = \{b_{i\ell}\}$ for its orthogonal complement $S_i^\perp$:*

$$DP_n : x \in S_i \quad \mapsto \quad B_i \subset S_i^\perp. \tag{76}$$

Each vector $b \in B_i$ can be viewed as a co-sample, *i.e., as a sample point drawn from the complement subspace $S_i^\perp$ to S_i. Therefore, if we evaluate the derivatives of P_n at all the sample points $X = \{x\}$, we obtain a set of co-samples $B = \{b\}$ for the union of all the complement subspaces $\cup S_i^\perp$. Obviously, identifying $S_i^\perp$ from B is exactly a GPCA problem that is* dual *to the original problem. If we apply again the PDA algorithm to B, then the output of the algorithm will be exactly the bases for the subspaces $(S_i^\perp)^\perp = S_i$.*[17]

Remark 21 (Connection with spectral clustering) *Although GPCA can be viewed as a special clustering problem, many of the classical clustering algorithms such as spectral clustering cannot be directly applied. This is because in order for spectral clustering techniques to work well, we should be able to define a distance function that is small for pairs of points in the same subspace and large for points in different subspaces. Such a distance should therefore depend only 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 X clearly does not have this property.*[18]

However, thanks to the duality equation (76), one can compute a basis for $S_i^\perp$ at every point x_i in S_i. A distance function between a point x_i in S_i and x_j in S_j can be defined between the two bases:

$$\mathcal{D}_{ij} = \langle S_i^\perp, S_j^\perp \rangle, \tag{77}$$

where we use $\langle \cdot, \cdot \rangle$ to denote 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., $S_{ij} = \exp(-\mathcal{D}_{ij}^2)$, for the N samples in X. This allows one to apply the classical spectral clustering algorithms to group the sample points according to their subspaces. Here the duality plays a crucial role of converting a multilinear clustering problem to a standard spectral clustering problem.

5.2 Obtaining one point per subspace by polynomial division

From the results in the previous section, we notice that one can obtain a basis for each $S_i^\perp$ directly from the derivatives of the polynomials representing $\cup_{i=1}^n S_i$. However, in order to proceed we need to have one point per subspace, i.e., we need to know the vectors $\{y_i \in S_i\}_{i=1}^n$. In the case of hyperplanes, this could readily be done by intersecting a line $\mathcal{L}$ with each one of the subspaces. However, this solution does not generalize to the case of subspaces of arbitrary dimensions. Consider for example the case of data lying on three one-dimensional subspaces of $\mathbb{R}^3$. Then a randomly chosen line $\mathcal{L}$ may not intersect any of the one-dimensional subspaces. Furthermore, because polynomials in the null space of L_n are no longer factorable, their zero set is no longer a union of subspaces, hence the points of intersection with $\mathcal{L}$ may not lie in any of the subspaces.

In this section, we propose a generalization of the recursive PDA for mixtures of hyperplanes described in Section 3.3.3. To this end, let $\{p_{n\ell}(x)\}_{\ell=1}^m$ be the set of m polynomials whose coefficients are in the null space of the data matrix L_n. Also, let $\tilde{x}$ be the projection of a point $x \in \mathbb{R}^K$ onto its closest subspace. From Lemma 2 we have that the Euclidean distance from point x to its closest subspace is given by

$$\|x - \tilde{x}\| = n\sqrt{P_n(x)\big(DP_n(x)^T DP_n(x)\big)^\dagger P_n(x)^T} + O\big(\|x - \tilde{x}\|^2\big), \tag{78}$$

[17] This comes at no surprise at all once one realizes that the polynomials associated with the dual problem can be viewed as polynomials in the coordinate ring for the original subspaces. According to [6], Chapter 16, their derivatives are exactly tangent vectors on these subspaces.

[18] This explains why spectral clustering algorithms typically do not work well when there is intersection among different groups, which is unfortunately the case with mixtures of hyperplanes.

where $P_n(x) = [p_{n1}(x) \cdots p_{nm}(x)] \in \mathbb{R}^{1\times m}$, $DP_n(x) = [Dp_{n1}(x) \cdots Dp_{nm}(x)] \in \mathbb{R}^{K\times m}$, and $A^\dagger$ is the Moore-Penrose inverse of A. Therefore, we can choose a point y_n lying on (close to) one of the subspaces as:

$$y_n = \underset{x\in X: DP_n(x)\neq 0}{\arg\min} P_n(x)(DP_n(x)^T DP_n(x))^\dagger P_n(x)^T, \tag{79}$$

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

In order to find a point y_{n-1} in one of the remaining $(n-1)$ subspaces, but not in S_n, we need to find a new set of polynomials $\{p_{(n-1)\ell}(x)\}$ defining the algebraic set $\cup_{i=1}^{n-1} S_i$. In the case of hyperplanes this was done by polynomial division, which is equivalent to solving for a vector $c_{n-1} \in \mathbb{R}^{M_{n-1}}$ from a linear system of the form $\mathcal{D}_n(b_n)c_{n-1} = c_n$, where $b_n \in S_I^\perp$ (see Remark 13). In the case of subspaces of arbitrary dimensions, we cannot simply divide $p_{n\ell}(x)$ by $b_n^T x$ for $b_n \in S_i^\perp$, because the polynomials $\{p_{n\ell}(x)\}$ may not be factorable. Furthermore, they do not necessarily have $b_n^T x$ as a common factor. The following theorem resolves this difficulty by showing how to compute the polynomials associated to the remaining subspaces $\cup_{i=1}^{n-1} S_i$.

Theorem 5 (Polynomial division) *For the GPCA problem, if the given sample set X is such that* $\dim(\text{null}(L_n)) = \dim(I_n)$, *then the set of homogeneous polynomials of degree $(n-1)$ associated with the remainder of algebraic set $\cup_{i=1}^{n-1} S_i$ are exactly $\{c_{n-1}^T \nu_{n-1}(x)\}$ for all $c_{n-1} \in \mathbb{R}^{M_{n-1}}$ that satisfy*

$$L_n \mathcal{D}_n(b_n) c_{n-1} = 0, \tag{80}$$

where b_n can be any vector in $S_n^\perp$.

Proof. We first show the necessity. That is, any polynomial of degree $n-1$, $c_{n-1}^T \nu_{n-1}(x)$, that vanishes on $\cup_{i=1}^{n-1} S_i$ satisfies the above equation. Since a point x in the original algebraic set $\cup_{i=1}^{n} S_i$ belongs to either $\cup_{i=1}^{n-1} S_i$ or S_n, we have $c_{n-1}^T \nu_{n-1}(x) = 0$ or $b_n^T x = 0$ as long as $b_n \in S_n^\perp$. Hence $p(x) \doteq (c_{n-1}^T \nu_{n-1}(x))(b_n^T x) = 0$, and $p(x)$ must be a linear combination of polynomials in P_n. If we denote $p(x)$ as $c_n^T \nu_n(x)$, then the vector of coefficients c_n must be in the null space of L_n. From $c_n^T \nu_n(x) = (c_{n-1}^T \nu_{n-1}(x))(b_n^T x)$, the relationship between c_n and c_{n-1} can be written as $\mathcal{D}_n(b_n)c_{n-1} = c_n$. Since $L_n c_n = 0$, c_{n-1} needs to satisfy the following linear system of equations

$$L_n \mathcal{D}_n(b_n) c_{n-1} = 0. \tag{81}$$

We now show the sufficiency. That is, if c_{n-1} is a solution to (80), then $c_n = \mathcal{D}_n(b_n)c_{n-1}$ is in the null space of L_n. From the construction of $\mathcal{D}_n$, we have $c_n^T \nu_n(x) = (c_{n-1}^T \nu_{n-1}(x))(b_n^T x)$. Then for every $x \in \cup_{i=1}^{n-1} S_i$ not in S_n, we have $c_{n-1}^T \nu_{n-1}(x) = 0$, because $b_n^T x \neq 0$. Therefore, $c_{n-1}^T \nu_{n-1}(x)$ is a homogeneous polynomial of degree $(n-1)$ that vanishes on $\cup_{i=1}^{n-1} S_i$. ∎

Thus a collection of polynomials $\{p_{(n-1)\ell}(x)\}$ for $\cup_{i=1}^{n-1} S_i$ can be obtained from the null space of $L_n \mathcal{D}_n(b_n) \in \mathbb{R}^{N\times M_{n-1}}$. By applying the same reasoning to the remaining subspaces, we obtain the following recursive *polynomial differentiation algorithm* (PDA-rec) for finding one point per subspace and computing the matrices of normal vectors. In the case in which all the subspaces are hyperplanes, Algorithm 6 reduces exactly to Algorithm 3.

Remark 22 (Avoiding polynomial division) *Similarly to the case of hyperplanes (see Remark 6), one may avoid computing P_i by choosing the points y_i with a heuristic function. Since a point in $\cup_{\ell=i}^{n} S_\ell$ must satisfy $\|B_i^T x\| \cdots \|B_n^T x\| = 0$, we can choose a point y_{i-1} on $\cup_{\ell=1}^{i-1} S_\ell$ as:*

$$y_{i-1} = \underset{x\in X: DP_n(x)\neq 0}{arg\,\min} \frac{\sqrt{P_n(x)(DP_n(x)^T DP_n(x))^\dagger P_n(x)^T} + \delta}{\|B_i^T x\| \cdots \|B_n^T x\| + \delta}, \tag{85}$$

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

6 Optimal GPCA in the presence of noise

In the previous sections, we addressed the GPCA problem in a purely algebraic fashion and proposed various algorithms for estimating a collection of subspaces using polynomial factorization or polynomial differentiation and division. In essence, all the algorithms we have presented so far use *linear algebraic* techniques to solve for the bases $B_i = [b_{i1}, \ldots, b_{i(K-k_i)}]$ of $S_i^\perp$, where $i = 1, \ldots, n$, from a set of nonlinear equations of the form (see equation (4))

$$p_{n\sigma}(x^j) = \prod_{i=1}^{n} (b_{i\sigma(i)}^T x^j) = 0 \qquad \text{for} \quad j = 1, \ldots, N, \tag{86}$$

Algorithm 6 (Polynomial Differentiation Algorithm (PDA) for Mixtures of Subspaces)

given the number of subspaces n, form the embedded data matrix $L_n \in \mathbb{R}^{N \times M_n}$.

for $i = n : 1$,

solve $L_i \mathbf{c} = 0$ to obtain a basis $\{\mathbf{c}_{i\ell}\}_{\ell=1}^{r_i}$ of $null(L_i)$;

set $p_{i\ell}(\boldsymbol{x}) = \mathbf{c}_{i\ell}^T \nu_n(\boldsymbol{x})$ and $P_i(\boldsymbol{x}) = [p_{i1}(\boldsymbol{x}) \cdots p_{ir_i}(\boldsymbol{x})] \in \mathbb{R}^{1 \times r_i}$;

do

$$\boldsymbol{y}_i = \arg\min_{\boldsymbol{x} \in \boldsymbol{X} : DP_i(\boldsymbol{x}) \neq 0} P_i(\boldsymbol{x}) \big(DP_i(\boldsymbol{x})^T DP_i(\boldsymbol{x})\big)^\dagger P_i(\boldsymbol{x})^T, \tag{82}$$

$$B_i = PCA\big(DP_i(\boldsymbol{y}_i)\big), \tag{83}$$

$$L_{i-1} = L_i \mathcal{D}_i(\boldsymbol{b}_i), \qquad \text{with } \boldsymbol{b}_i \text{ the first column of } B_i, \tag{84}$$

end;

end;

assign point $\boldsymbol{x}^j$ to subspace S_i if $i = \arg\min_{\ell=1,\ldots,n} \|B_\ell^T \boldsymbol{x}^j\|$.

with σ representing a particular choice of one normal vector $\boldsymbol{b}_{i\sigma(i)}$ from basis B_i.

However, the algebraic algorithms provide a "linear" solution to the GPCA problem at the cost of neglecting the nonlinear constraints that the entries of each one of the vector of coefficients $\boldsymbol{c}_n \in \mathbb{R}^{M_n}$ must satisfy, the so-called Brill's equations (see Remark 8). In the presence of noise, one could set up an optimization problem that searches directly for the bases $\{B_i\}$, instead of the searching first for the polynomials $\{p_{n\sigma}\}$. This can be done by minimizing the violation of the above algebraic equations (or some variation of them) in a least squares sense. For example, we could minimize the algebraic error

$$E_A(B_1, \ldots, B_n) = \sum_{j=1}^{N} \prod_{i=1}^{n} \|B_i^T \boldsymbol{x}^j\|^2, \tag{87}$$

which should be zero if there was no noise. Minimizing this algebraic error in fact provides a more robust estimate of the subspace bases, because it uses a minimal representation of the unknowns. However, the solution to this optimization problem may be biased, because the algebraic error in (87) does not coincide with the negative log-likelihood (up to constant factors) of the data given the parameters.

In this section, we derive an optimal algorithm for reconstructing the subspaces when the sample data points are corrupted with i.i.d. zero-mean Gaussian noise. We show that the optimal solution can be obtained by minimizing a function similar to the algebraic error in (87), but properly normalized. Since our derivation is based on segmentation independent constraints, we do not need to model the membership of each data point with a probability distribution. This represents a great advantage over EM-like techniques, because we do not need to iterate between the Expectation and Maximization steps. In fact, our approach eliminates the Expectation step *algebraically* and solves the GPCA problem by directly optimizing over the subspace parameters (Maximization step).

Let $\boldsymbol{X} = \{\boldsymbol{x}^j \in \mathbb{R}^K\}_{j=1}^N$ be the given set of data points, which are generated by adding i.i.d. zero-mean Gaussian noise to a set of noise free points $\{\tilde{\boldsymbol{x}}^j \in \mathbb{R}^K\}_{j=1}^N$ lying on a collection of subspaces $\{S_i \subset \mathbb{R}^K\}_{i=1}^n$ of dimension $k_i = \dim(S_i)$, where $0 < k_i < K$, for $i = 1, \ldots, n$. Given the number of subspaces n, the subspace dimensions $\{k_i\}_{i=1}^n$, and the collection of noisy data points $\boldsymbol{X}$, we would like to find a basis $B_i \in \mathbb{R}^{K \times (K-k_i)}$ for $S_i^\perp$ by minimizing the error between the data and the noise free points

$$\sum_{j=1}^{N} \|\tilde{\boldsymbol{x}}^j - \boldsymbol{x}^j\|^2 \tag{88}$$

subject to the fact that the noise free points $\{\tilde{\boldsymbol{x}}^j\}_{j=1}^N$ must lie on one of the subspaces.

To this end, notice that for each noise free data point $\tilde{\boldsymbol{x}}^j$ there exists a subspace S_i such that $\tilde{\boldsymbol{x}}^j \in S_i$. In other words, there exists a matrix B_i such that $B_i^T \tilde{\boldsymbol{x}}^j = 0$. Therefore, a point $\boldsymbol{x}^j$ belongs to one of the subspaces if and only if

$$p_n(\tilde{\boldsymbol{x}}^j) = \prod_{i=1}^{n} \|B_i^T \tilde{\boldsymbol{x}}^j\| = 0. \tag{89}$$

Therefore, we can estimate the bases $B_1, B_2, \ldots, B_n$ by solving the following constrained optimization problem

$$\begin{array}{lll} \min & \sum_{j=1}^{N} \|\tilde{x}^j - x^j\|^2 & \\ \text{subject to} & \prod_{i=1}^{n} \|B_i^T \tilde{x}^j\| = 0 & j = 1, \ldots, N \\ & B_i^T B_i = I \in \mathbb{R}^{(K-k_i)\times(K-k_i)} & i = 1, \ldots, n, \end{array} \tag{90}$$

where the last constraint forces the columns of each basis B_i to be orthonormal.

By using Lagrangian multipliers λ^j for each constraint on $\tilde{x}^j$ and a matrix of Lagrange multipliers $\Lambda_i = \Lambda_i^T \in \mathbb{R}^{(K-k_i)\times(K-k_i)}$ for each constraint on B_i, the above optimization problem is equivalent to minimizing

$$\sum_{j=1}^{N} \|\tilde{x}^j - x^j\|^2 + \sum_{j=1}^{N} \lambda^j \prod_{i=1}^{n} \|B_i^T \tilde{x}^j\| + \sum_{i=1}^{n} \text{trace}(\Lambda_i (I - B_i^T B_i)). \tag{91}$$

After taking partial derivatives with respect to $\tilde{x}^j$ and setting them to zero we obtain

$$2(\tilde{x}^j - x^j) + \lambda^j Dp_n(\tilde{x}^j) = 0. \tag{92}$$

After multiplying on the left by $Dp_n(\tilde{x}^j)^T$ and $(\tilde{x}^j - x^j)^T$ we obtain

$$\lambda^j = 2\frac{Dp_n(\tilde{x}^j)^T x^j}{\|Dp_n(\tilde{x}^j)\|^2} \tag{93}$$

$$\|\tilde{x}^j - x^j\|^2 = \frac{1}{2} x^{jT} Dp_n(\tilde{x}^j) \lambda^j, \tag{94}$$

where we have used the fact that

$$\tilde{x}^{jT} Dp_n(\tilde{x}^j) = \tilde{x}^{jT} \sum_{i=1}^{n} \prod_{\ell \neq i} \frac{\|B_\ell^T \tilde{x}^j\|}{\|B_i^T \tilde{x}^j\|} B_i B_i^T \tilde{x}^j = n \prod_{i=1}^{n} \|B_i^T \tilde{x}^j\| = n p_n(\tilde{x}^j) = 0. \tag{95}$$

After substituting (93) and (94) on the objective function (88) we obtain

$$\tilde{E}_O(\{\tilde{x}^j\}, \{B_i\}) = \sum_{j=1}^{N} \frac{\left(x^{jT} Dp_n(\tilde{x}^j)\right)^2}{\|Dp_n(\tilde{x}^j)\|^2}. \tag{96}$$

We can obtain an objective function on the bases only by considering first order statistics of $p_n(x^j)$. Since this is equivalent to setting $\tilde{x}^j = x^j$ in (96) and $x^{jT} Dp_n(x^j) = n p_n(x^j)$, we obtain the simplified objective function

$$E_O(B_1, \ldots, B_n) = \sum_{j=1}^{N} \frac{\left(n p_n(x^j)\right)^2}{\|Dp_n(x^j)\|^2} = \sum_{j=1}^{N} \frac{n^2 \prod_{i=1}^{n} \|B_i^T x^j\|^2}{\left\| \sum_{i=1}^{n} \prod_{\ell \neq i} \frac{\|B_\ell^T x^j\|}{\|B_i^T x^j\|} B_i B_i^T x^j \right\|^2}, \tag{97}$$

which is essentially the same as the algebraic error (87), but properly normalized according to the chosen noise model.

In summary, we can obtain an estimate of the bases $\{B_i\}_{i=1}^n$ by minimizing the objective function $E_O(B_1, \ldots, B_n)$ subject to the constraints $B_i^T B_i = I$, for $i = 1, \ldots, n$. One can use standard nonlinear optimization techniques to minimize E_O starting from the solution given by Algorithm 6, or any of the other GPCA algorithms depending on the case.

Remark 23 (Optimal error in the case of hyperplanes) *In the particular case of data lying on hyperplanes, we have that $B_i = b_i \in \mathbb{R}^K$ for $i = 1, \ldots, n$. Therefore, the objective function in (97) becomes*

$$E_O(b_1, \ldots, b_n) = \sum_{j=1}^{N} \frac{\left(n p_n(x^j)\right)^2}{\|Dp_n(x^j)\|^2} = \sum_{j=1}^{N} \frac{n^2 \prod_{i=1}^{n} (b_i^T x^j)^2}{\left\| \sum_{i=1}^{n} \prod_{\ell \neq i} (b_\ell^T x^j) b_i \right\|^2}, \tag{98}$$

as demonstrated in [21].

7 Initialization of iterative algorithms in the presence of noise

In this section, we briefly describe two algorithms for clustering subspaces: K-subspace and Expectation Maximization (EM). Both algorithms start with a random initialization for the subspace bases, and then iterate between the segmentation of the data and the estimation of the bases. Therefore, we can use either K-subspace or EM to improve the linear algebraic estimate given by GPCA (PFA or PDA).

7.1 The K-subspace algorithm

The K-subspace algorithm is an extension of the well-known K-means algorithm to the case of mixtures of subspaces. K-subspace minimizes a weighted square distance from point $\boldsymbol{x}^j$ to subspace S_i which is defined as

$$\sum_{i=1}^{n}\sum_{j=1}^{N} w_{ij}\|B_i^T\boldsymbol{x}^j\|^2 = \sum_{i=1}^{n}\sum_{j=1}^{N} w_{ij}\text{trace}(B_i^T\boldsymbol{x}^j\boldsymbol{x}^{jT}B_i) = \sum_{i=1}^{n}\text{trace}(B_i^T\Sigma_i B_i) \tag{99}$$

where the weights w_{ij} represent the membership of the data point j to subspace i and $\Sigma_i = \sum_{j=1}^{N} w_{ij}\boldsymbol{x}^j\boldsymbol{x}^{jT} \in \mathbb{R}^{K\times K}$ can be interpreted as the covariance matrix of the data points in subspace S_i. The K-subspace algorithm starts by randomly initializing the bases $\{B_i\}_{i=1}^n$. Then, the algorithm minimizes the error function (99) using a coordinate descent algorithm that iterates between the following two steps.

In the first step it minimizes over $\{w_{ij}\}$ with $\{B_i\}$ held constant, which gives the following formula for the weights

$$w_{ij} = \begin{cases} 1 & i = \arg\min_{\ell=1,\ldots,n}\|B_\ell^T\boldsymbol{x}^j\|^2 \\ 0 & \text{otherwise} \end{cases}. \tag{100}$$

In the second step, K-subspace minimizes (99) over the bases $\{B_i\}_{i=1}^n$ with the weights $\{w_{ij}\}$ held constant. Since the bases are not uniquely defined, we impose the additional constraint that the columns of B_i are orthonormal, i.e., $B_i^TB_i = I$. By using a matrix of Lagrange multipliers $\Lambda_i = \Lambda_i^T \in \mathbb{R}^{(K-k_i)\times(K-k_i)}$ we can minimize the Lagrangian

$$\sum_{i=1}^{n}\text{trace}(B_i^T\Sigma_i B_i) + \sum_{i=1}^{n}\text{trace}(\Lambda_i(I - B_i^TB_i)). \tag{101}$$

After taking partial derivatives with respect to B_i and setting them to zero we obtain

$$\Sigma_i B_i = B_i\Lambda_i. \tag{102}$$

After multiplying by B_i^T on the left and noticing that $B_i^TB_i = I$, we obtain $B_i^T\Sigma_iB_i = \Lambda_i$. This implies that $\Lambda_i \succeq 0$, because $\Sigma_i \succeq 0$. Furthermore, after replacing $B_i^T\Sigma_iB_i = \Lambda_i$ on the the objective function (99) we obtain

$$\sum_{i=1}^{n}\sum_{j=1}^{N} w_{ij}\|B_i^T\boldsymbol{x}^j\|^2 = \sum_{i=1}^{n}\text{trace}(\Lambda_i). \tag{103}$$

Therefore, the objective function is minimized by choosing Λ_i as a diagonal matrix with the eigenvalues of Σ_i in the diagonal and B_i as the matrix of eigenvectors of Σ_i. Given the new B_i, one can recompute the weights w_{ij} and then iterate until convergence.

7.2 The Expectation Maximization algorithm

The EM algorithm assumes that the data points $\{\boldsymbol{x}^j\}_{j=1}^N$ are generated by firstly choosing one of the subspaces $\{S_i\}_{i=1}^n$, say subspace S_i, according to a multinomial distribution with parameters $\{0 \le \pi_i \le 1\}_{i=1}^n$, $\sum_{i=1}^n \pi_i = 1$, and secondly choosing a point $\boldsymbol{x}^j = \tilde{\boldsymbol{x}}^j + B_i\boldsymbol{s}_{ij}$, where $\tilde{\boldsymbol{x}}^j$ is a noise free point lying on S_i, and $\boldsymbol{s}_{ij}$ is zero-mean Gaussian noise with covariance $\sigma_i^2 I \in \mathbb{R}^{(K-k_i)\times(K-k_i)}$. Let $z_{ij} = 1$ denote the event that point j corresponds to subspace i. Then the complete log-likelihood (neglecting constant factors) on both the data $\boldsymbol{x}^j$ and the latent variables z_{ij} is given by

$$\log\prod_{j=1}^{N}\prod_{i=1}^{n}\left(\frac{\pi_i}{\sigma_i}\exp\left(-\frac{\|B_i^T\boldsymbol{x}^j\|^2}{2\sigma_i^2}\right)\right)^{z_{ij}} = \sum_{j=1}^{N}\sum_{i=1}^{n} z_{ij}(\log(\pi_i) - \log(\sigma_i)) - z_{ij}\frac{\|B_i^T\boldsymbol{x}^j\|^2}{2\sigma_i^2}.$$

E-step: Computing the expected log-likelihood. Given a current estimate for the parameters $\theta = \{(B_i, \sigma_i, \pi_i)\}_{i=1}^n$, we can compute the expected value of the latent variables

$$w_{ij} \doteq E[z_{ij}|\boldsymbol{x}^j, \theta] = P(z_{ij} = 1|\boldsymbol{x}^j, \theta) = \frac{\frac{\pi_i}{\sigma_i}\exp(-\frac{\|B_i^T\boldsymbol{x}^j\|^2}{2\sigma_i^2})}{\sum_{i=1}^n \frac{\pi_i}{\sigma_i}\exp(-\frac{\|B_i^T\boldsymbol{x}^j\|^2}{2\sigma_i^2})}.$$

Then the expected complete log-likelihood is given by

$$\sum_{j=1}^{N}\sum_{i=1}^{n} w_{ij}(\log(\pi_i) - \log(\sigma_i)) - w_{ij}\frac{\|B_i^T x^j\|^2}{2\sigma_i^2}.$$

M-step: Maximizing the expected log-likelihood. The Lagrangian for π_i is

$$\sum_{i=1}^{n}\sum_{j=1}^{N} w_{ij}\log(\pi_i) + \lambda(1 - \sum_{i=1}^{n}\pi_i) \quad \Rightarrow \quad \pi_i = \frac{\sum_{j=1}^{N} w_{ij}}{N}.$$

The Lagrangian for B_i is

$$\sum_{i=1}^{n} -\text{trace}\left(\frac{B_i^T\Sigma_i B_i}{2\sigma_i^2}\right) + \text{trace}(\Lambda_i(B_i^T B_i - I)) \quad \Rightarrow \quad \Sigma_i B_i = 2\sigma_i^2 B_i\Lambda_i.$$

Similarly to the K-subspace algorithm, the objective function for B_i becomes $-\sum_{i=1}^{n}\text{trace}(\Lambda_i)$, with $\Lambda_i \succeq 0$. Thus B_i is a matrix whose columns are the eigenvectors of Σ_i associated with the $(K - k_i)$ smallest eigenvalues. Finally, after taking derivatives of the expected log-likelihood with respect to σ_i we obtain

$$\sigma_i^2 = \frac{\sum_{j=1}^{N} w_{ij}\|B_i^T x^j\|^2}{\sum_{j=1}^{N} w_{ij}}.$$

If for all i $\sigma_i = \sigma$, then we have

$$\sigma^2 = \frac{\sum_{i=1}^{n}\sum_{j=1}^{N} w_{ij}\|B_i^T x^j\|^2}{N}.$$

8 Experiments on synthetic data

In this section, we evaluate the performance of PFA and PDA (algebraic and recursive) by comparing them with K-subspace and EM on synthetically generated data. The experimental setup consists of choosing $n = 2, 3, 4$ collections of $N = 200n$ points lying on randomly chosen $k = 2$ dimensional subspaces of $\mathbb{R}^3$. Zero-mean Gaussian noise from 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 $\{b_i\}_{i=1}^n$ and the estimates $\{\hat{b}_i\}_{i=1}^n$ is computed as

$$\text{error} = \frac{1}{n}\sum_{i=1}^{n}\text{acos}\left(b_i^T\hat{b}_i\right)\text{(degrees)}. \tag{104}$$

8.1 Error versus noise

Figure 4 (left) and Figure 4 (right) plot the mean error as a function of the noise level for PFA, PDA, K-subspace, and EM for a number of subspaces of $n = 4$. Similar results were obtained for $n = 2, 3$, though with smaller errors.

Notice that the estimates of PDA-alg with $m = 1$ line are only slightly better than those of PFA, while the estimates of PDA-alg with $m = 3$ and PDA-rec with $\delta = 0.02$ have an error of about 50% compared to PFA. For PDA-alg we observed that the error decreases as m increases, though the increase of performance was not significant for $m > 3$.

For PDA-rec 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. The best performance (among the purely algebraic algorithms) is obtained by PDA-rec, because it deals automatically with noisy data and outliers by choosing the points in an optimal fashion.

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-rec, both K-means and EM reduce the error to approximately 35-50% with respect to random initialization.

8.2 Error versus number of subspaces

Figure 5 plots the estimation error of PDA-rec as a function of the number of subspaces n, for different levels of noise. As expected, the error increases as a function of n.

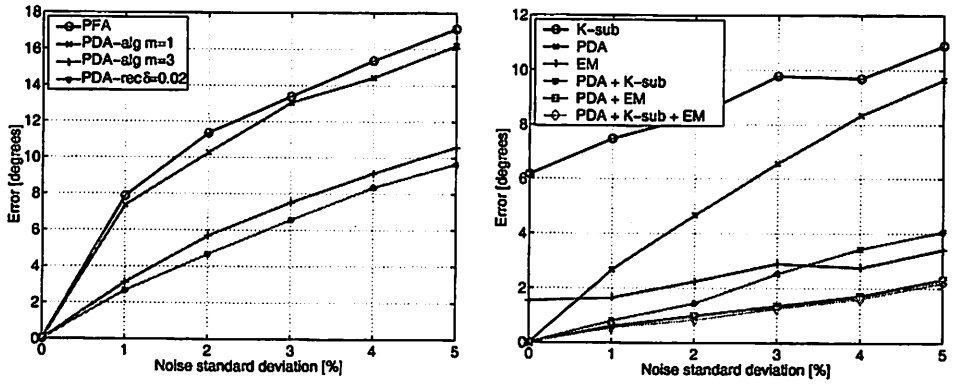


Figure 4: Error versus noise for data lying on $n = 4$ subspaces of $\mathbb{R}^3$ of dimension $k = 2$. Left: PFA, PDA-alg ($m = 1$ and $m = 3$) and PDA-rec ($\delta = 0.02$). Right: PDA-rec, K-subspace and EM randomly initialized, K-subspace and EM initialized with PDA-rec, and EM initialized with K-subspace initialized with PDA-rec.

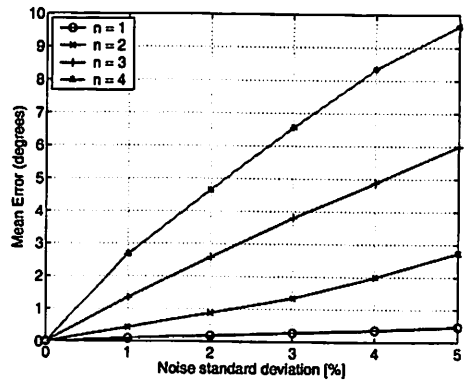


Figure 5: Error versus noise for PDA-rec ($\delta = 0.02$) for data lying on $n = 1, \ldots, 4$ subspaces of $\mathbb{R}^3$ of dimension $k = 2$.

8.3 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. Among the algebraic algorithms, the fastest one is PFA which directly factors $p_n(x)$ given c_n. The extra cost of PDA-alg and PDA-rec relative to PFA is on building the polynomial $q_n(t)$ and computing $Dp_n(x)$ for all $x \in X$, respectively. Overall, PDA-rec gives half of the error of PFA in about twice as much time. Notice also that PDA-rec reduces the number of iterations of K-subspace and EM to approximately 1/3 and 1/2, respectively. The computing times for K-subspace and EM are also reduced even if the extra time spent on initialization with PDA-rec or PDA-rec + K-subspace is included.

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

Algorithm	$L_n c = 0$	PFA	PDA-alg	PDA-alg	PDA-rec
Time (sec.)	0.0854	0.1025	0.1765	0.3588	0.1818
# Iterations		1	1	3	1
Algorithm	K-sub	PDA-rec +K-sub	EM	PDA-rec +EM	PDA-rec+ K-sub+EM
Time (sec.)	0.4637	0.2525	1.0408	0.6636	0.7528
# Iterations	19.7	7.1	30.8	17.1	15.0

9 Applications of GPCA in computer vision

This section presents applications of GPCA in computer vision problems, such as vanishing point detection, 2D and 3D motion segmentation, and face clustering with varying illumination.

9.1 Detection of vanishing points

Given a collection of parallel lines in 3D, 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. Given n set of parallel lines, we represent their images in projective space as $\{\boldsymbol{\ell}_j \in \mathbb{P}^2\}_{j=1}^N$ and the vanishing points as $\{\boldsymbol{v}_i \in \mathbb{P}^2\}_{i=1}^n$. Since for each line j there exists a vanishing point $\boldsymbol{v}_i$ 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 6 (left) shows an example from the Corel Database with $n = 3$ sets of $N = 30$ manually extracted parallel lines. 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 recursive PDA to the set of lines assuming unknown segmentation and used the resulting vanishing points to initialize K-subspace. The vanishing points estimated by PDA and PDA + K-subspace are shown in Figure 6 (center) and compared 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 6 (right) shows the segmentation of the lines obtained by PDA. There is only one misclassified line, the top horizontal line in the image, because it approximately passes through two of the vanishing points.

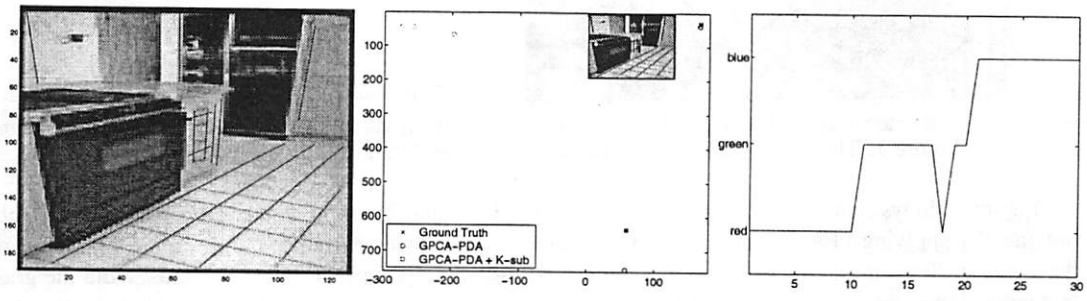


Figure 6: Detecting vanishing points using GPCA. Left: Image #364081 from the Corel database with 3 sets of 10 parallel lines superimposed. Center: Comparing the vanishing points estimated by PDA and PDA followed by K-subspace with the ground truth. Right: Segmentation of the 30 lines given by PDA.

9.2 Segmentation of 2-D translational motions from image intensities

In this section, we apply GPCA to the problem of segmenting the 2-D motion field of a video sequence from measurements of the partial derivatives of the image intensities. We assume that the scene can be modeled as a mixture of purely translational 2-D motion models.[19] That is, we assume that the optical flow at every pixel in the image sequence, $\boldsymbol{u} = [\mathrm{u}, \mathrm{v}, 1] \in \mathbb{P}^2$, can take one out of n possible values $\{\boldsymbol{u}_i\}_{i=1}^n$. Furthermore, we assume that the number of motion models is unknown. If we assume that the surface of each object is Lambertian, then the optical flow of pixel $\boldsymbol{x} = [x_1, x_2, 1]^T \in \mathbb{P}^2$ is related to the partials of the image intensity $\boldsymbol{y} = [I_{x_1}, I_{x_2}, I_t]^T \in \mathbb{R}^3$ at $\boldsymbol{x}$ by the well-known *brightness constancy constraint*

$$\boldsymbol{y}^T \boldsymbol{u} = I_{x_1} \mathrm{u} + I_{x_2} \mathrm{v} + I_t = 0. \tag{105}$$

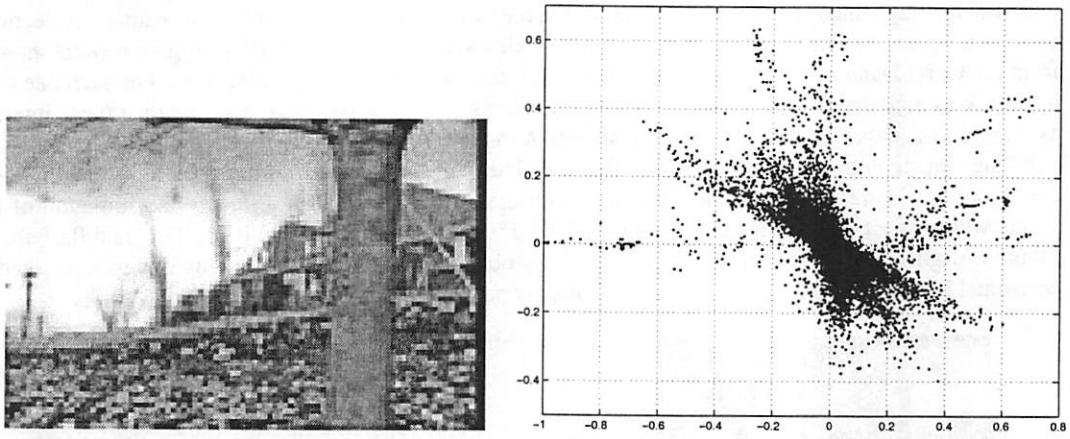
Given the vector of partial derivatives $\boldsymbol{y}$ of an arbitrary pixel $\boldsymbol{x}$ in the scene, there exists an optical flow $\boldsymbol{u}_i$ such that $\boldsymbol{y}^T \boldsymbol{u}_i = 0$. Thus the following *multibody brightness constancy constraint* must be satisfied by *all* the pixels in the image

$$\boxed{g_n(\boldsymbol{y}) = (\boldsymbol{u}_1^T \boldsymbol{y})(\boldsymbol{u}_2^T \boldsymbol{y}) \cdots (\boldsymbol{u}_n^T \boldsymbol{y}) = \prod_{i=1}^n (\boldsymbol{u}_i^T \boldsymbol{y}) = \tilde{\boldsymbol{u}}^T \nu_n(\boldsymbol{y}) = 0.} \tag{106}$$

[19] We generalize to the affine motion model in [23].

The multibody brightness constancy constraint, $g_n(\boldsymbol{y})$, is a homogeneous polynomial of degree n on $\boldsymbol{y}$. We denote its vector of coefficients $\tilde{\boldsymbol{u}} \in \mathbb{R}^{M_n}$, where $M_n = (n+1)(n+2)/2$, as the *multibody optical flow* associated with the scene. Therefore, the segmentation of purely translational motion models from image intensities can be interpreted as a GPCA problem with $k = 2$ and $K = 3$, i.e., the segmentation of planes in $\mathbb{R}^3$. The optical flows $\{\boldsymbol{u}_i\}_{i=1}^n$ correspond to the normals to the planes, and the image partial derivatives $\{\boldsymbol{y}^j\}_{j=1}^N$ are the data points. Therefore, we can use any of the GPCA algorithms for hyperplanes (PFA or PDA) to determine the number of motion models n and the motion models $\{\boldsymbol{u}_i\}_{i=1}^n$ from the image derivatives $\{\boldsymbol{y}^j\}_{j=1}^N$.

Figure 7 shows a frame of the flower garden and the corresponding image data projected onto the I_{x_1}-I_t plane to facilitate visualization. We observe from 7(b) that the image partial derivatives lie approximately on three planes passing through the origin. Notice that the image data is quite noisy and contains many outliers.



(a) A frame from the flower garden sequence (b) Image data projected onto the I_x-I_z plane

Figure 7: The flower garden sequence and its image derivatives projected onto the I_x-I_z plane.

Figure 8 shows segmentation results for frames 1, 11, 21 and 31 of the flower garden sequence. This results are obtained by applying PFA (Algorithm 1) to the image data, followed by the EM algorithm for mixtures of subspaces described in Section 7.2. The sequence is segmented into three groups: the tree, the houses and the grass.[20] Notice that even though the purely translational motion model is fairly simplistic and clearly inappropriate for this sequence, the GPCA algorithm gives a relatively good segmentation that can be easily improved with some post-processing that incorporates spatial constraints. A better segmentation can also be obtained by using a richer motion model, such as the affine motion model, as we study in [23].

We now apply GPCA to the segmentation of dynamic scenes with translucent motions. We consider a scene in which a semi-transparent screen is moving horizontally in front of a hand that is moving vertically. In this case, there is no notion of a connected group of pixels moving together. In fact, almost every pixel moves independently from its neighbors, yet there are two groups of pixels moving together. Notice that any segmentation algorithm based on computing either optical flow, or an affine model [25], or a motion profile [16], from a local neighborhood would fail, since there is no local neighborhood containing a single motion model. Figure 9 shows the segmentation of the first five frames of the sequence using GPCA followed by EM. The algorithm is able to segment out a reasonably good outline of the moving hand. Notice that it is not possible to obtain the whole hand as a single group, because the hand has no texture.

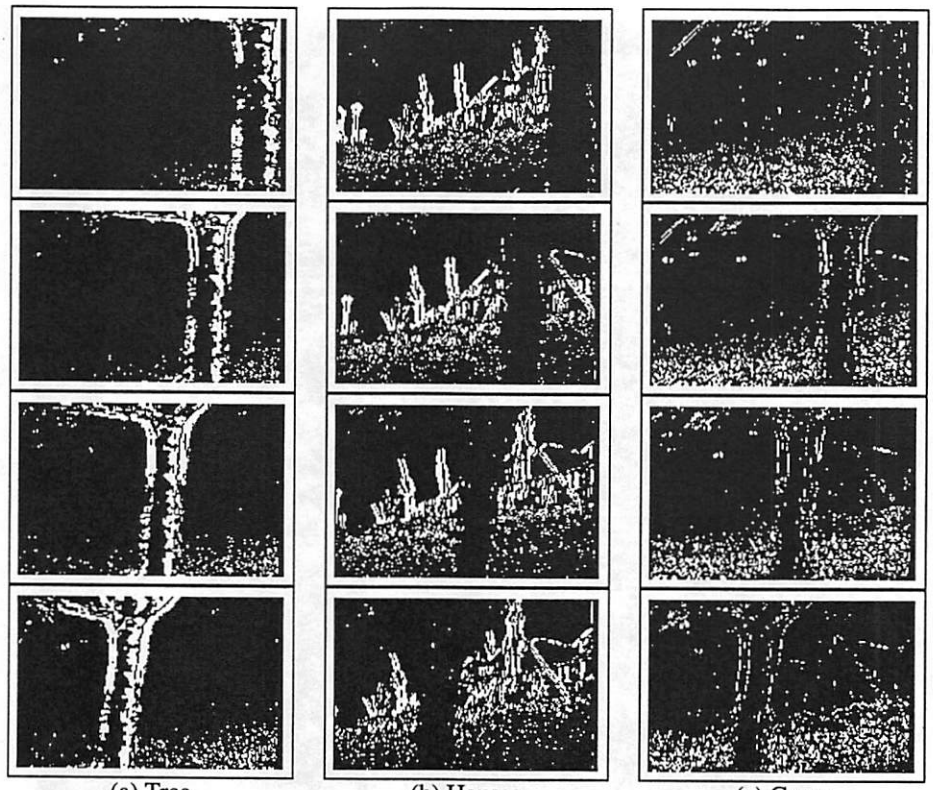
9.3 Segmentation of 2-D affine motions from feature points or optical flow

In this section, we consider the problem of segmenting a collection of 2-D affine motions from measurements of either the optical flow at each pixel, or the position of a set of feature points in two frames of a video sequence, and show that they are GPCA problems with $K = 5$ and $k = 3$.

9.3.1 2-D Motion segmentation from optical flow

Let $\{\boldsymbol{u}_j \in \mathbb{P}^2\}_{j=1}^N$ be N measurements of the optical flow at the N pixels $\{\boldsymbol{x}_j \in \mathbb{P}^2\}_{j=1}^N$. We assume that the optical flow field can be modeled as a collection of n 2-D affine motion models $\{A_i \in \mathbb{R}^{3\times 3}\}_{i=1}^n$. That is, for each optical

[20] We did not cluster pixels without texture, such as pixels in the sky, because the image derivatives are approximately zero for those pixels, i.e., $\boldsymbol{y} \approx 0.$, and hence they can be assigned to either of the three models.



(a) Tree (b) Houses (c) Grass

Figure 8: Segmenting frames 1, 11, 21 and 31 of the the flower garden sequence using GPCA applied to the image derivatives.

flow $\boldsymbol{u}_j$ there exists an affine motion A_i such that

$$\boldsymbol{u}_j = A_i \boldsymbol{x}_j = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ 0 & 0 & 1 \end{bmatrix} \boldsymbol{x}_j. \tag{107}$$

In other words, the optical flow $\boldsymbol{u} = [\mathrm{u}, \mathrm{v}, 1]^T$ at pixel $\boldsymbol{x} = [x_1, x_2, 1] \in \mathbb{P}^2$ is related to the affine motion parameters $a_{11}, a_{12}, \ldots, a_{23}$ by

$$a_{11}x_1 + a_{12}x_2 + a_{13} - \mathrm{u} = 0 \tag{108}$$

$$a_{21}x_1 + a_{22}x_2 + a_{23} - \mathrm{v} = 0. \tag{109}$$

Equations (108) and (109) define a three-dimensional subspace of a five-dimensional space with coordinates $[x_1, x_2, 1, \mathrm{u}, \mathrm{v}]^T \in \mathbb{R}^5$. The estimation of those subspaces from data points $[x_1, x_2, 1, \mathrm{u}, \mathrm{v}]^T$ lying on those subspaces is simply a GPCA problem with $k_1 = \cdots = k_n = k = 3$ and $K = 5$.

According to our discussion in Section 4.1, we can reduce this problem to a GPCA problem with $k = 3$ and $K = 4$ by first projecting the data onto a four-dimensional space. The particular structure of equations (108) and (109) with normal vectors $[a_{11}, a_{12}, a_{13}, -1, 0]^T$ and $[a_{21}, a_{22}, a_{23}, 0, -1]^T$ suggests to project the data onto two four-dimensional subspaces with coordinates $[x_1, x_2, x_3, \mathrm{u}]^T$ and $[x_1, x_2, x_3, \mathrm{v}]^T$. Each one of these two projections allows us to compute the first and second row of each affine motion model, respectively, by using any of the GPCA algorithms for hyperplanes described in Section 3.

However, it could happen that, even though the affine motions $\{A_i\}_{i=1}^n$ are different from each other, a particular row could be common to two or more affine motions. Therefore, in general we will obtain $n_1 \leq n$ first rows and $n_2 \leq n$ second rows from each one of the two projections. Furthermore, even if $n_1 = n_2 = n$, we still do not know which first and second rows correspond to the *same* affine motion model.

Fortunately, we can exploit the structure of the problem in order to find the affine motions $\{A_i\}_{i=1}^n$ from the collection of first and second rows, $\{\boldsymbol{a}_{1i} \in \mathbb{R}^3\}_{i=1}^{n_1}$ and $\{\boldsymbol{a}_{2i} \in \mathbb{R}^3\}_{i=1}^{n_2}$, respectively. To this end, let $\boldsymbol{\ell}_1 \in \mathbb{R}^N$ and

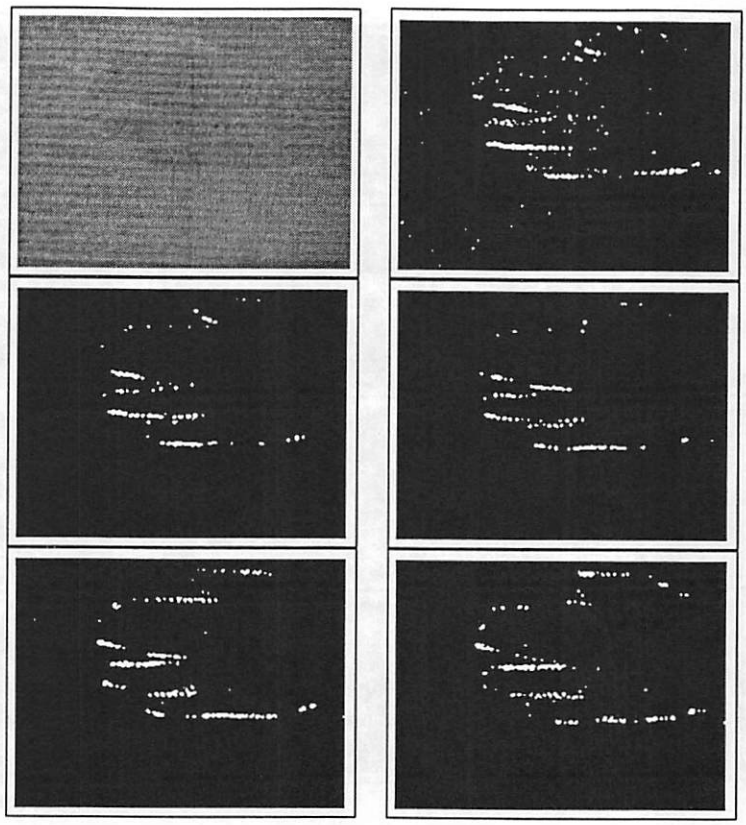


Figure 9: Segmenting a sequence with a hand moving behind a moving semi-transparent screen using GPCA applied to the image derivatives.

$\boldsymbol{\ell}_2 \in \mathbb{R}^N$ be vectors of labels giving the segmentation of the data according to each projection, i.e.,

$$\boldsymbol{\ell}_1(j) = i \quad \text{if} \quad i = \arg\min_{i=1,\ldots,n_1} |\boldsymbol{a}_{1i}^T \boldsymbol{x}_j - \mathrm{u}_j| \quad j = 1,\ldots,N \tag{110}$$

$$\boldsymbol{\ell}_2(j) = i \quad \text{if} \quad i = \arg\min_{i=1,\ldots,n_2} |\boldsymbol{a}_{2i}^T \boldsymbol{x}_j - \mathrm{v}_j| \quad j = 1,\ldots,N. \tag{111}$$

Then the N rows of the matrix of labels $[\boldsymbol{\ell}_1 \ \boldsymbol{\ell}_2] \in \mathbb{R}^{N\times 2}$ will take on only n different values. Let the rows of $\mathcal{L} = [\ell_{ij}] \in \mathbb{R}^{n\times 2}$ be the n different rows in $[\boldsymbol{\ell}_1 \ \boldsymbol{\ell}_2]$. Then the affine motion models can be computed as

$$\boxed{A_i = \begin{bmatrix} \boldsymbol{a}_{1\ell_{i1}}^T \\ \boldsymbol{a}_{2\ell_{i2}}^T \\ e_3^T \end{bmatrix} \quad i = 1,\ldots,n.} \tag{112}$$

We therefore have the following algorithm (Algorithm 7) for segmenting a collection of 2-D affine motion models from optical flow measurements.

9.3.2 2-D Motion segmentation from feature points

Let $\{\boldsymbol{x}_1^j \in \mathbb{P}^2\}_{j=1}^N$ and $\{\boldsymbol{x}_2^j \in \mathbb{P}^2\}_{j=1}^N$ be a collection of N feature points in two frames of a video sequence. We assume that the motion of those features can be modeled as a collection of n 2-D affine motion models $\{A_i \in \mathbb{R}^{3\times 3}\}_{i=1}^n$. That is, for each feature pair $(\boldsymbol{x}_1^j, \boldsymbol{x}_2^j)$ there exist a 2-D affine motion A_i such that

$$\boldsymbol{x}_2^j = A_i \boldsymbol{x}_1^j = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ 0 & 0 & 1 \end{bmatrix} \boldsymbol{x}_1^j. \tag{113}$$

We notice that if we replace $\boldsymbol{x}_2 = \boldsymbol{u}$ in the above equations, then the problem of segmenting 2-D affine motion models from feature points becomes identical to the problem of segmenting 2-D affine motion models from optical flow. We can therefore use Algorithm 7 to estimate the collection of affine motion models from the given feature points.

Algorithm 7 (Segmentation of 2-D affine motions from optical flow)

Given measurements of optical flow $\{\boldsymbol{u}_j\}_{j=1}^N$ at the N pixels $\{\boldsymbol{x}_j\}_{j=1}^N$, recover the number of affine motion models n, the affine matrix A_i associated with motion i, and the segmentation of the image measurements as follows:

1. **Affine motions.** Estimate the number of affine motions and the affine matrices as follows:
   (a) Apply a GPCA algorithm with $k = 3$ and $K = 4$ to the data $\{[\boldsymbol{x}_j, \mathrm{u}_j]^T\}_{j=1}^N$ to determine the number of different first rows $n_1 \leq n$ in the affine matrices $\{A_i\}_{i=1}^n$ and the n_1 different first rows $\{\boldsymbol{a}_{1i} \in \mathbb{R}^3\}_{i=1}^{n_1}$. Cluster the data into n_1 groups and define a vector of labels $\boldsymbol{\ell}_1 \in \mathbb{R}^N$ such that $\boldsymbol{\ell}_1(j) = i$ if point $\boldsymbol{x}_j$ belongs to group i.
   (b) Repeat step 1(a) with data $\{[\boldsymbol{x}_j, \mathrm{v}_j]^T\}_{j=1}^N$ to obtain $n_2 \leq n$ different second rows $\{\boldsymbol{a}_{2i} \in \mathbb{R}^3\}_{i=1}^{n_2}$ and the corresponding vector of labels $\boldsymbol{\ell}_2 \in \mathbb{R}^N$.
   (c) Extract the n different rows from the matrix of labels $[\boldsymbol{\ell}_1\ \boldsymbol{\ell}_2] \in \mathbb{R}^{N\times 2}$ into the matrix $\mathcal{L} = [\ell_{ij}] \in \mathbb{R}^{n\times 2}$ and use them to compute the affine motions $\{A_i\}_{i=1}^n$ as in (112).
2. **Segmentation of the image measurements.** Assign image measurement $(\boldsymbol{x}_j, \boldsymbol{u}_j)$ to the affine motion A_i that minimizes $\|\boldsymbol{u}_j - A_i\boldsymbol{x}_j\|^2$.

9.4 Segmentation of 3-D translational motions from feature points

In this section, we apply GPCA to the problem of segmenting the 3-D motion of multiple objects undergoing a purely translational motion. We assume that the scene can be modeled as a mixture of purely translational motion models,[21] $\{T_i\}_{i=1}^n$, where $T_i \in \mathbb{R}^3$ represents the translation of object i relative to the camera between the two consecutive frames.

Given the images $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ of a point in object i in the first and second frame, respectively, the rays $\boldsymbol{x}_1$, $\boldsymbol{x}_2$ and T_i are coplanar, as illustrated in Figure 10. Therefore $\boldsymbol{x}_1$, $\boldsymbol{x}_2$ and T_i must satisfy the well-known epipolar constraint for linear motions

$$\boldsymbol{x}_2^T(T_i \times \boldsymbol{x}_1) = 0. \tag{114}$$

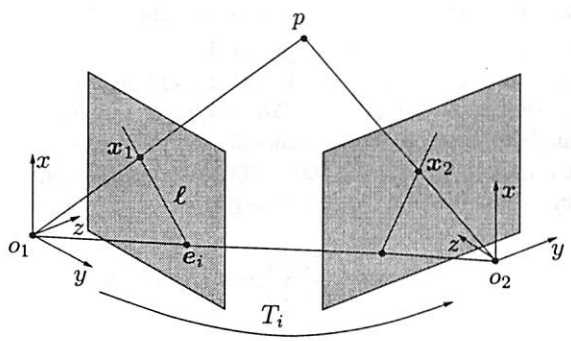


Figure 10: Epipolar geometry: Two projections $\boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathbb{R}^3$ of a 3-D point p from two vantage points. The relative Euclidean transformation between the two vantage points is given by $T_i \in \mathbb{R}^3$. The intersection of the line (o_1, o_2) with each image plane is the so-called epipole $\boldsymbol{e}_i$. The epipolar line $\boldsymbol{\ell}$ is the intersection of the plane (p, o_1, o_2) with the first image plane.

In the case of an uncalibrated camera, the epipolar constraint reads $\boldsymbol{x}_2^T(\boldsymbol{e}_i \times \boldsymbol{x}_1) = 0$, where $\boldsymbol{e}_i \in \mathbb{R}^3$ is known as the *epipole* and is linearly related to the translation vector $T_i \in \mathbb{R}^3$. Since the epipolar constraint can be conveniently rewritten as

$$\boldsymbol{e}_i^T(\boldsymbol{x}_2 \times \boldsymbol{x}_1) = 0, \tag{115}$$

[21] We generalize to the case of arbitrary rotation and translation in [24, 22], where we consider the problem of segmenting a mixture of fundamental matrices.

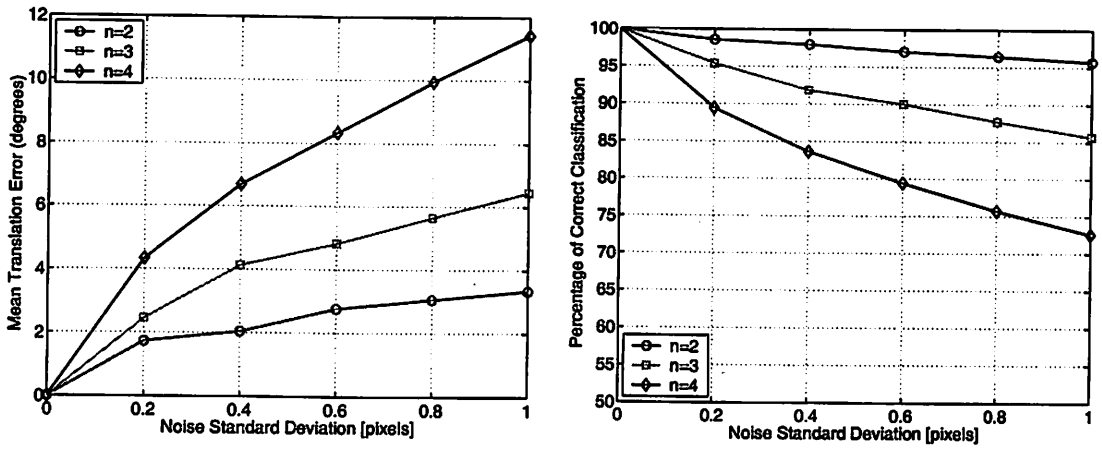


Figure 11: Performance of PDA on segmenting 3-D translational motions. Left: Estimation error as a function of noise for $n = 2, 3, 4$ motions. Right: Percentage of correctly classified image pairs as a function of noise for $n = 2, 3, 4$ motions.

where $e_i \in \mathbb{R}^3$ represents the epipole associated with the i^{th} motion, $i = 1, \ldots, n$, if we define the vector $\ell = (x_2 \times x_1) \in \mathbb{R}^3$ as a data point, then we have that $e_i^T \ell = 0$. Therefore, given any image pair (x_1, x_2) corresponding to one of the n moving objects, the vector $\ell = x_2 \times x_1$, the so-called *epipolar line*, satisfies the following homogeneous polynomial of degree n

$$\boxed{g_n(\ell) = (e_1^T \ell)(e_2^T \ell) \cdots (e_n^T \ell) = \prod_{i=1}^{n} (e_i^T \ell) = \tilde{e}^T \nu_n(\ell) = 0.} \tag{116}$$

We denote the vector of coefficients $\tilde{e} \in \mathbb{R}^{M_n}$, where $M_n = (n+1)(n+2)/2$, as the *multibody epipole*. We conclude that the segmentation of linearly moving objects can be interpreted as a GPCA problem with $k = 2$ and $K = 3$, where the epipoles $\{e_i\}_{i=1}^n$ correspond to the normal to the planes and the epipolar lines $\{\ell^j\}_{j=1}^N$ are the data points.

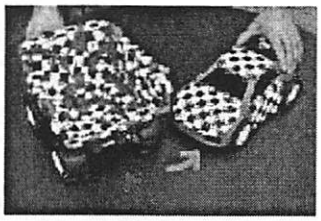
Therefore, given a set of images $\{(x_1^j, x_2^j)\}_{j=1}^N$ of a collection of N points in 3D undergoing n *distinct* linear motions $e_1, \ldots, e_n \in \mathbb{R}^3$, one can use the set of epipolar lines $\ell^j = x_2^j \times x_1^j$, where $j = 1, \ldots, N$, to estimate the number of motions n and the epipoles e_i using the GPCA algorithms for hyperplanes (PFA or PDA).

Figure 11 shows the performance of recursive PDA on synthetic image data. We choose $n = 2, 3, 4$ collections of $N = 100n$ image pairs undergoing a purely translational motion. Zero-mean Gaussian noise from 0 to 1 pixel s.t.d. is added to the image data for an image size of 500 × 500. We run 1000 trials for each noise level. For each trial the error between the true (unit) epipoles $\{e_i\}_{i=1}^n$ and the estimates $\{\hat{e}_i\}_{i=1}^n$ is computed as

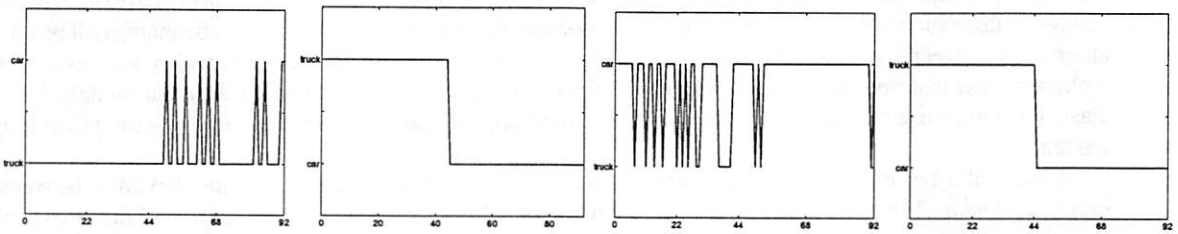
$$\text{error} = \frac{1}{n} \sum_{i=1}^{n} \text{acos}\left(e_i^T \hat{e}_i\right) \text{(degrees)}. \tag{117}$$

As expected, the performance deteriorates as the level of noise or the number of motion increases. The maximum error is of 12° for $n = 4$ motions. Notice also that the percentage of correctly classified image pairs reduces as the noise or the number of motions increases. The percentage of correct classification is always above 70%.

We now apply PFA and recursive PDA to a sequence with $n = 2$ linearly moving objects (a truck and a car) and $N = 92$ features (44 for the truck and 48 for the car), as shown in Figure 12 (a). When PFA is applied with an ordering of $(3, 1, 2)$ for the coordinates of the data, then a perfect segmentation is obtained (Figure 12 (c)), and the error in the translation estimates is 1.2° for the truck and 3.3° for the car. However, if an ordering of $(1, 2, 3)$ or $(2, 3, 1)$ is chosen, PFA gives a very poor segmentation of the data, as shown in Figures 12 (b) and (d), respectively. This shows that the performance of PFA with noisy data depends on the choice of the ordering of the variables, because the polynomial $q_n(t)$ is built from the last two coordinates only. On the other hand, if we apply PDA to the data we obtain a perfect segmentation, regardless of the ordering of the coordinates, as shown in Figure 12 (e). The mean error of PDA is 5.9° for the truck and 1.7° for the car.



(a) First frame



(b) PFA: $(1, 2, 3)$ (c) PFA: $(3, 1, 2)$ (d) PFA: $(2, 3, 1)$ (e) PDA: any order

Figure 12: Segmenting 3-D translational motions using GPCA. Segmentation obtained by PFA and PDA using different changes of coordinates.

9.5 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-dimensional subspace. 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.[22] 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.

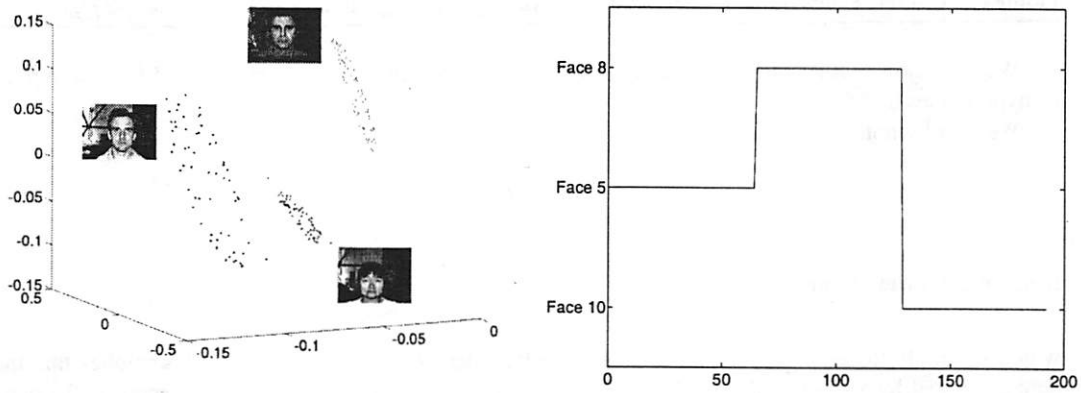


Figure 13: Clustering a subset of the Yale Face Database B consisting of 64 frontal views under varying lighting conditions for subjects 5, 8 and 10. Left: Image data projected onto the three principal components. Right: Clustering of the images using PDA.

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 $K =$

[22] The homogeneous coordinates of a vector $\boldsymbol{x} \in \mathbb{R}^K$ are $[\boldsymbol{x}^T\ 1]^T \in \mathbb{R}^{K'+1}$.

30×40 pixels. Then we projected the data onto the first $K' = 3$ principal components, as shown in Figure 13 (left). We applied GPCA to this data set in homogeneous coordinates ($\mathbb{R}^4$). We fitted $n = 3$ ($k = 3$)-dimensional subspaces to the data using the recursive PDA algorithm. Given the subspace normals, we clustered the images by assigning each image to its closest subspace. Figure 13 (right) shows the segmentation results.

10 Application of GPCA to identification of linear hybrid systems

Hybrid systems are mathematical models that can be used to describe continuous phenomena that exhibit discontinuous behavior due to sudden changes of dynamics. For instance, the continuous trajectory of a bouncing ball results from the alternation between free fall and elastic contact. However, hybrid dynamical models can also be used to approximate a phenomenon that does not itself exhibit discontinuous behavior, by concatenating different models from a simple class. For instance, a non-linear dynamical system can be approximated by switching among various linear dynamical models.

A particular but important class of hybrid systems is obtained by assuming that the dynamics between discrete events are *linear*. This class of systems is important not only because the analysis and design of linear control systems is well understood, but also because many real processes can be approximated arbitrarily well by models in this class.

In this section, we look at the problem of modeling input/output by a piecewise linear (hybrid) dynamical models: Given input/output data, we want to simultaneously estimate the number of underlying linear models, the parameters of each model, the discrete state, and possibly the switching mechanism that governs the transitions from one linear model to another.

For simplicity, we will concentrate on a class of discrete-time linear hybrid systems, known as *piecewise autoregresive exogenous* systems (PWARX). The evolution of a PWARX system is determined by a collection of n ARX models $\{\Sigma_i\}_{i=1}^n$ of the form

$$y_t = a_1 y_{t-1} + a_2 y_{t-2} + \cdots + a_{n_a} y_{t-n_a} + c_1 u_{t-1} + c_2 u_{t-2} + \cdots + c_{n_c} u_{t-n_c} \tag{118}$$

where $\{u_t, t = 1, 2, \dots\}$ is the *input*, $\{y_t, t = 1, 2, \dots\}$ is the *output*, and $\{a_i\}_{i=1}^{n_a}$ and $\{c_i\}_{i=1}^{n_c}$ are the *model parameters*. The ARX models are connected by switches indexed by a number of *discrete states* $\lambda_t \in \{1, 2, \dots, n\}$. The evolution of the discrete state λ_t can be described in a variety of ways. In this section, we will restrict ourselves to the class of *Jump-linear systems (JLS)*, in which λ is a deterministic but unknown input that is piecewise constant and finite-valued.

We consider the following identification problem:

Problem 3 (Identification of PWARX models)

Given input/output data $\{u_t, y_t\}_{t=1}^T$ generated by a PWARX model with known dimensions n_a and n_c, estimate the number of discrete states n, the model parameters $\{a_i\}_{i=1}^{n_a}$, $\{c_i\}_{i=1}^{n_c}$ and the discrete state $\{\lambda_t\}_{t=0}^T$.

We now show that Problem 3 is simply a GPCA problem with $K = n_a + n_c + 1$ and $k = n_a + n_c$, i.e., clustering of hyperplanes in $\mathbb{R}^K$.

We start by noticing that if we let

$$x_t = (u_{t-n_c}, \dots, u_{t-1}, y_{t-n_a}, \dots y_{t-1}, -y_t)^T \in \mathbb{R}^{n_a+n_c+1} \tag{119}$$

$$b = (c_{n_c}, \cdots, c_1, a_n, \dots a_1, 1)^T \in \mathbb{R}^{n_a+n_c+1}, \tag{120}$$

then we can write equation (118) as

$$b^T x_t = 0 \qquad t \geq n, \tag{121}$$

which is simply the equation of a hyperplane in $\mathbb{R}^K$, where $K = n_a + n_c + 1$. This implies that the input/output data generated by a single ARX models lives in a hyperplane whose normal vector b encodes the parameters of the ARX model. Therefore if we are given a PWARX model generated with ARX models $\{\Sigma_i\}_{i=1}^n$, then we can represent the PWARX model as a collection of hyperplanes with normal vectors $\{b_i\}_{i=1}^n$ encoding the model parameters. Furthermore, the input/output data generated by the PWARX model must live in the union of all the hyperplanes $\cup_{i=1}^n S_i$, where $S_i = \{x : b_i^T x = 0\}$. In fact, when λ_t switches from $\lambda_t = i$ to $\lambda_{t+1} = j$, the input/output data jumps from hyperplane S_i to hyperplane S_j.

Notice also that if we are given input/output (dynamic) data $\{u_t, y_t\}_{t=1}^T$ generated by a PWARX model, then we can always generate a new set of (static) data points $\{x_t\}_{t=n_a}^T$. Therefore, according to our analysis in Section 3, if $T - n_a + 1 \geq M_n(K) - 1$, and the evolution of the discrete state is such that the the discrete mode $i = 1, \dots, n$ is

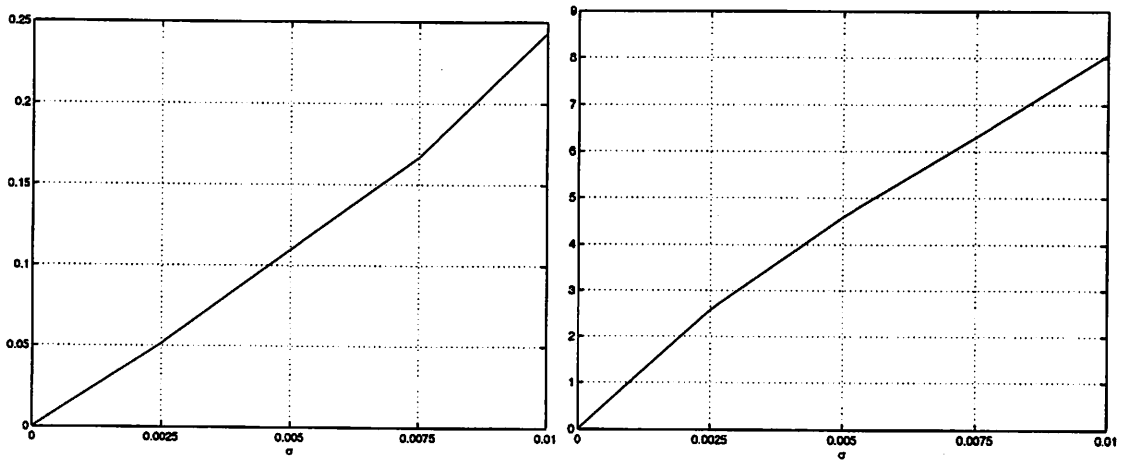


Figure 14: Mean error over 1000 trials for the identification of the model parameters (top) and the discrete state (bottom) as a function of the standard deviation of the measurement error σ.

visited at least $k = n_a + n_c$ times in the time interval $1 \leq t \leq T$, then one can estimate the number of discrete states n and the model parameters $\{b_i\}_{i=1}^n$ uniquely using either the polynomial factorization or the polynomial differentiation algorithms. Then, given the model parameters, one can determine the discrete state as

$$\lambda_t = \arg \min_{i=1,\ldots,n} (b_i^T x_t)^2 \qquad \text{for} \quad t \geq n. \tag{122}$$

We present simulation results on the identification of PWARX systems with $n = 3$ discrete states. Each ARX model has dimensions $n_a = 2$ and $n_c = 1$ and is corrupted with i.i.d. zero-mean Gaussian noise w_t with standard deviation σ as

$$y_t = a_1(\lambda_t) y_{t-1} + a_2(\lambda_t) y_{t-2} + c_1(\lambda_t) u_{t-1} + w_t. \tag{123}$$

For each trial, the model parameters (a_1, a_2) for each discrete state are randomly chosen so that the poles of each linear system are uniformly distributed on the annulus $0.8 \leq \|z\| \leq 1 \subset \mathbb{C}$. The model parameter c_1 for each discrete state is chosen according to a zero-mean unit variance Gaussian distribution. The value of the discrete state was chosen as

$$\lambda_t = \begin{cases} 1 & 1 \leq t \leq 30 \\ 2 & 31 \leq t \leq 60 \\ 3 & 61 \leq t \leq 100 \end{cases} \tag{124}$$

The input sequence $\{u_t\}$ was drawn from a zero-mean unit variance Gaussian distribution. The noise w_t was drawn from a zero-mean Gaussian noise with standard deviation $\sigma \in [0, 0.01]$, which simulate a measurement error of about 1%. Figure 14 shows the mean error on the estimation of the model parameters[23] and the discrete state[24], respectively, as a function of σ. Both the model parameters and the continuous state are correctly estimated with an error that increases approximately linearly with the amount of noise. Notice that the discrete state is incorrectly estimated approximately 8% of the times for $\sigma = 0.01$. Notice also that there is no error for $\sigma = 0$. Figure 15 shows the reconstruction of the discrete trajectory for a particular trial with $\sigma = 0.01$. Notice that there are 5 time instances in which the estimates of the discrete state are incorrect.

11 Conclusions and open issues

We have proposed a novel approach to the identification of mixtures of subspaces, the so-called *Generalized Principal Component Analysis* (GPCA) problem.

[23] The error between the estimated model parameters $(\hat{a}_1, \hat{a}_2, \hat{c}_1)$ and the true model parameters (a_1, a_2, c_1) was computed as $\|(\hat{a}_1, \hat{a}_2, \hat{c}_1) - (a_1, a_2, c_1)\|$, averaged over the number of models and trials.

[24] The error between the estimated discrete state $\hat{\lambda}_t$ and the true discrete state λ_t was computed as the number of times in which $\hat{\lambda}_t \neq \lambda_t$, averaged over the number of trials.

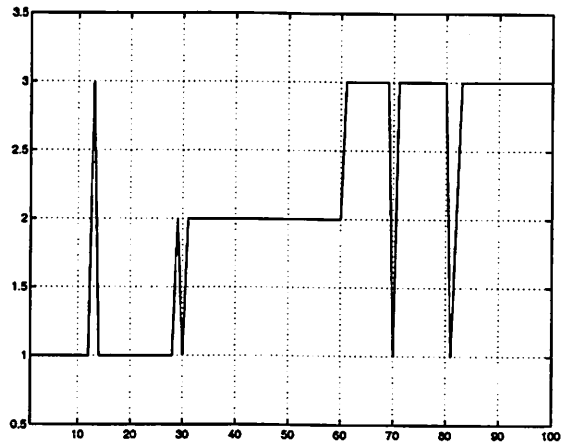


Figure 15: Evolution of the estimated discrete state $\hat{\lambda}_t$.

In the absence of noise, we casted GPCA in an algebraic geometric framework in which the collection of subspaces is represented by a set of homogeneous polynomials whose degree n corresponds to the number of subspaces and whose factors (roots) encode the subspace parameters. In the case of n subspaces of equal dimension k, we derived rank constraints on the data from which one can estimate the number of subspaces n and their dimension k. We then proposed two algorithms for estimating the subspaces from sample data. The *polynomial factorization algorithm* (PFA) is designed for subspaces of co-dimension one, i.e., hyperplanes, and obtains a basis for each hyperplane from the roots of a polynomial of degree n in one variable and from the solution of a collection of linear systems in n variables. The *polynomial differentiation algorithm* (PDA) is designed for subspaces of arbitrary dimensions and obtains a basis for each subspace by evaluating the derivatives of the set of polynomials representing the subspaces at a collection of n points in each one of the subspaces. The points are chosen automatically from points in the dataset that minimize a certain distance function.

In the presence of noise, we casted GPCA as a constrained nonlinear least squares problem which minimizes the error between the noisy points and their projections subject to all mixture constraints. By converting this constrained problem into an unconstrained one, we obtained an optimal function from which the subspaces can be recovered using standard non-linear optimization techniques.

We applied GPCA to a variety of estimation problems in which the data comes simultaneously from multiple (approximately) linear models. We first presented experiments on low-dimensional data showing that the polynomial differentiation algorithm gives about half of the error of the polynomial factorization algorithm. We also showed that the polynomial differentiation algorithm improves the performance of iterative techniques, such as K-subspace and EM, by about 50% with respect to random initialization. We then presented various applications of GPCA on computer vision problems such as vanishing point detection, 2-D and 3-D motion segmentation, and face clustering under varying illumination.

Open issues include a detailed analysis of the robustness of all the GPCA algorithms in the presence of noisy data. At present, the GPCA algorithms work well when the number and dimension of the subspaces is small, but the performance deteriorates as the number of subspaces increases. This is because all the algorithms start by estimating a collection of polynomials in a linear fashion, thus neglecting the nonlinear constraints among the coefficients of those polynomials, the so-called Brill's equations. Another open issue has to do with the estimation of the number of subspaces n and their dimensions $\{k_i\}_{i=1}^n$. In the case of hyperplanes and/or subspaces of equal dimension k, we derived formulas for estimating n and k in the absence of noise. However, the formulas are based on rank constraints that are hard to verify in the presence of noise. In order to estimate n and k in a robust fashion, one could for example combine our rank constraints with model selection techniques, similarly to [13]. Furthermore, in the case of subspaces of arbitrary dimensions, the estimation of the number of subspaces is still an open question. In fact, as mentioned in Remark 19, it is possible to underestimate the number of subspaces even in the noise free case. Finally, throughout the paper we hinted connections of GPCA with Kernel Methods, e.g., the Veronese map gives an embedding that satisfies the modeling assumptions of KPCA (see Remark 5), and with spectral clustering techniques, e.g., the polynomial differentiation algorithm allowed us to define a similarity matrix in Remark 21. Further exploring these connections and others will be the subject of future research.

References

[1] J. Bochnak, M. Coste, and M. F. Roy. *Real Algebraic Geometry*. Springer, 1998.

[2] T.E. Boult and L.G. Brown. Factorization-based segmentation of motions. In *Proc. of the IEEE Workshop on Motion Understanding*, pages 179–186, 1991.

[3] M. Collins, S. Dasgupta, and R. Schapire. A generalization of principal component analysis to the exponential family. In *Neural Information Processing Systems*, volume 14, 2001.

[4] J. Costeira and T. Kanade. A multibody factorization method for independently moving objects. *International Journal of Computer Vision*, 29(3):159–179, 1998.

[5] L. de Lathauwer, B. de Moor, and J. Vandewalle. A multilinear singular value decomposition. *SIAM Journal of Matrix Analysis*, 21(4):1253–1278, 2000.

[6] D. Eisenbud. *Commutative Algebra: with a view towards algebraic geometry*. GTM. Springer, 1996.

[7] I.M. Gelfand, M.M. Kapranov, and A.V. Zelevinsky. *Discriminants, Resultants, and Multidimensional Determinants*. Birkhäuser, 1994.

[8] J. Harris. *Algebraic Geometry: A First Course*. Springer-Verlag, 1992.

[9] R. Hartshorne. *Algebraic Geometry*. Springer, 1977.

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

[11] I. Jollife. *Principal Component Analysis*. Springer-Verlag, New York, 1986.

[12] K. Kanatani. Motion segmentation by subspace separation and model selection. In *IEEE International Conference on Computer Vision*, volume 2, pages 586–591, 2001.

[13] K. Kanatani and C. Matsunaga. Estimating the number of independent motions for multibody motion segmentation. In *Asian Conference on Computer Vision*, pages 7–12, 2002.

[14] S. Lang. *Algebra*. Addison-Wesley Publishing Company, 3rd edition, 1993.

[15] B. Scholkopf, A. Smola, and K.-R. Muller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10:1299–1319, 1998.

[16] J. Shi and J. Malik. Motion segmentation and tracking using normalized cuts. In *IEEE International Conference on Computer Vision*, pages 1154–1160, 1998.

[17] M. Shizawa and K. Mase. A unified computational theory for motion transparency and motion boundaries based on eigenenergy analysis. In *International Conference on Computer Vision and Pattern Recognition*, pages 289–295, 1991.

[18] S. Smale. Complexity theory and numerical analysis. *Acta Numerica*, January 2000.

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

[20] M. Tipping and C. Bishop. Probabilistic principal component analysis. *Journal of the Royal Statistical Society*, 61(3):611–622, 1999.

[21] R. Vidal, Y. Ma, and S. Sastry. Generalized principal component analysis (GPCA). In *International Conference on Computer Vision and Pattern Recognition*, 2003.

[22] R. Vidal, Y. Ma, S. Soatto, and S. Sastry. Two-view multibody structure from motion. *International Journal of Computer Vision*, 2004.

[23] R. Vidal and S. Sastry. Segmentation of dynamic scenes from image intensities. In *IEEE Workshop on Motion and Video Computing*, pages 44–49, 2002.

[24] R. Vidal, S. Soatto, Y. Ma, and S. Sastry. Segmentation of dynamic scenes from the multibody fundamental matrix. In *ECCV Workshop on Visual Modeling of Dynamic Scenes*, 2002.

[25] J. Wang and E. Adelson. Layered representation for motion analysis. In *International Conference on Computer Vision and Pattern Recognition*, pages 361–366, 1993.