**Generalized Principal Component Analysis (GPCA):**
**an Algebraic Geometric Approach to Subspace Clustering and Motion Segmentation**

by

René Esteban Vidal

B.S. (P. Universidad Católica de Chile) 1995
M.S. (University of California at Berkeley) 2000

A dissertation submitted in partial satisfaction of the
requirements for the degree of
Doctor of Philosophy

in

Engineering – Electrical Engineering and Computer Sciences

in the

GRADUATE DIVISION
of the
UNIVERSITY of CALIFORNIA at BERKELEY

Committee in charge:

Professor Shankar Sastry, Chair
Professor Jitendra Malik
Professor Charles Pugh

Fall 2003

The dissertation of René Esteban Vidal is approved:

Chair Date

Date

Date

University of California at Berkeley

Fall 2003

**Generalized Principal Component Analysis (GPCA):**
**an Algebraic Geometric Approach to Subspace Clustering and Motion Segmentation**


René Esteban Vidal

## Abstract

Generalized Principal Component Analysis (GPCA):
an Algebraic Geometric Approach to Subspace Clustering and Motion Segmentation

by

René Esteban Vidal
Doctor of Philosophy in Engineering – Electrical Engineering and Computer Sciences
University of California at Berkeley
Professor Shankar Sastry, Chair

Simultaneous data segmentation and model estimation refers to the problem of estimating a collection of models from sample data points, without knowing which points correspond to which model. This is a challenging problem in many disciplines, such as machine learning, computer vision, robotics and control, that is usually regarded as "chicken-and-egg". This is because if the segmentation of the data was known, one could easily fit a single model to each group of points. Conversely, if the models were known, one could easily find the data points that best fit each model. Since in practice neither the models nor the segmentation of the data are known, most of the existing approaches start with an initial estimate for the either the segmentation of the data or the model parameters and then iterate between data segmentation and model estimation. However, the convergence of iterative algorithms to the global optimum is in general very sensitive to initialization of both the number of models and the model parameters. Finding a good initialization remains a challenging problem.

This thesis presents a novel algebraic geometric framework for simultaneous data segmentation and model estimation, with the hope of providing a theoretical footing for the problem as well as an algorithm for initializing iterative techniques. The algebraic geometric approach presented in this thesis is based on eliminating the data segmentation part *algebraically* and then solving the model estimation part *directly* using all the data and without having to iterate between data segmentation and model estimation. The algebraic elimination of the data segmentation part is achieved by finding algebraic equations that are *segmentation independent*, that is equations that are satisfied by *all* the data regardless of the group or model associated with each point.

For the classes of problems considered in this thesis, such segmentation independent constraints are polynomials of a certain degree in several variables. The degree of the polynomials corresponds to the number of groups and the factors of the polynomials encode the model parameters associated with each group. The problem is then reduced to

1. *Computing the number of groups from data*: this question is answered by looking for polynomials with the smallest possible degree that fit all the data points. This leads to simple rank constraints on the data from which one can estimate the number of groups after embedding the data into a higher-dimensional linear space.

2. *Estimating the polynomials representing all the groups from data*: this question is trivially answered by showing that the coefficients of the polynomials representing the data lie in the null space of the embedded data matrix.

3. *Factoring such polynomials to obtain the model for each group*: this question is answered with a novel polynomial factorization technique based on computing roots of univariate polynomials, plus a combination of linear algebra with multivariate polynomial differentiation and division. The solution can be obtained in closed form if and only if the number of groups is less than or equal to four.

The theory presented in this thesis is applicable to segmentation problems in which the data has a piecewise constant, piecewise linear or piecewise bilinear structure and is well motivated by various problems in computer vision, robotics and control. The case of piecewise constant data shows up in the segmentation of static scenes based on different cues such as intensity, texture and motion. The case of piecewise linear data shows up computer vision problems such as detection of vanishing points, clustering of faces under varying illumination, and segmentation of dynamic scenes with linearly moving objects. It also shows up in control problems such as the identification of linear hybrid systems. The case of piecewise bilinear data shows up in the multibody structure from motion problem in computer vision, i.e., the problem of segmenting dynamic scenes with multiple rigidly moving objects.

Professor Shankar Sastry
Dissertation Committee Chair

To my family,

Kathia, Amelia and Oscar

for their endless love, support and patience.

# Contents

# List of Figures

# List of Tables

## Acknowledgements

First and foremost, I would like to thank my research advisor, Professor Shankar Sastry for his extraordinary support in all aspects of my graduate life. His superb guidance, encouragement and enthusiasm, his pleasant personality, and the care and respect he shows for his students have made working with him a great pleasure. His commitment to the highest intellectual standards, the depth and breath of his research, and his broad and far-reaching vision have been and will continue to be a constant source of inspiration.

I would also like to thank Professor Jitendra Malik and Professor Charles Pugh for serving on my Dissertation Committee and Professor David Forsyth for serving on my Qualifying Examination Committee. Professor Malik introduced me to various segmentation problems in computer vision. His expertise, questions and suggestions have been very useful on improving my PhD work.

My deepest gratitude goes to Professor Yi Ma at UIUC, with whom I have had an extremely pleasant and fruitful collaboration over the past few years. His mathematical expertise, passion for geometry, attention to detail, and brilliant questions and suggestions have significantly improved the material of this thesis. I am also very grateful to Professor Soatto at UCLA, with whom I spent a fantastic year as a visiting student at the UCLA Vision Lab. His inspirational advice, constant motivation, and extraordinary care and support have been invaluable. I also thank Dr. Omid Shakernia at UC Berkeley for innumerable hours of inspiring discussions and joint work on vision-based control, omnidirectional vision, pursuit-evasion games, and so many other things.

My research has also benefited from discussions and interactions with Professor Jana Kŏsecká at George Mason University, who introduced me to the beauty of epipolar geometry and motivated me to do research in computer vision, Professor Kostas Daniilidis at the University of Pennsylvania, who introduced me to the geometry of omnidirectional cameras, and Professor Ruzena Bajcsy and Dr. Christopher Geyer at UC Berkeley, with whom I have had wonderful discussions about various problems in computer vision. I would also like to thank Professor George Pappas at the University of Pennsylvania and Professor John Lygeros at the University of Patras, who introduced me to the world of nonlinear and hybrid systems, Professor John Oliensis at the Stevens Institute of Technology, who shared with me his expertise in structure from motion and factorization methods during a summer internship at NEC Research Institute, Dr. Peter Cheeseman at NASA Ames, who introduced me to the connections between SLAM and vision during a summer internship at RIACS, NASA Ames, and Dr. Noah Cowan at Johns Hopkins University for wonderful discussions about vision, robotics, and vision-based control.

I am very grateful to have been surrounded by an extraordinary team of graduate students and researchers that have made my graduate life very pleasant and always challenging. I would like to thank all my colleagues in the Berkeley Aerial Robot Project, especially Professor João Hespanha, Shawn Hsu, Jin Kim, Shahid Rashid, Peter Ray, Shawn Schaffert, Cory Sharp, David Shim, and Bruno Sinopoli, my colleagues in the Berkeley Computer Vision Reading Group Parvez Ahammad, Aaron Ames, Marci Meingast and Jacopo Piazzi, and my colleagues in the UCLA Vision Lab Alessandro Bissacco, Gianfranco Doretto, Alessandro Duci, Paolo Favaro, Hailin Jin, Jason Meltzer, and Payam Saisan.

My former research advisor Professor Aldo Cipriano and my friend Dr. Julio Concha were fundamental on my coming to Berkeley. Without their extraordinary support, advice, encouragement, teaching and friendship I wouldn't have made it here in the first place. Many thanks!

Finally, I would like to thank my family for their endless love, support, patience, and care during all these years, especially Kathia, Amelia and Oscar, to whom this thesis is lovely dedicated.

# Chapter 1

# Introduction

## 1.1 Motivation

A wide variety of problems in engineering, applied mathematics and statistics can be phrased as an *inference* problem, that is a problem in which one is supposed to infer a *model* that explains a given a collection of data points. In many cases, the data can be explained by a single *smooth* model that can be as simple as the mean of the data as illustrated in Figure 1.1(a), or a hyperplane containing the data as illustrated in Figure 1.1(b), or as complex as an arbitrary manifold as illustrated in Figure 1.1(c). The second case shows up, for example, in face recognition where one assumes that the intensities in the image of a face under varying illumination lie on a linear subspace of a higher-dimensional space. The third case shows up, for example, in the *identification of linear dynamical systems*, where one is supposed to estimate the parameters $A$ and $C$, and the state trajectory $\{x_t, t = 1, 2, \ldots\}$ of a linear dynamical system

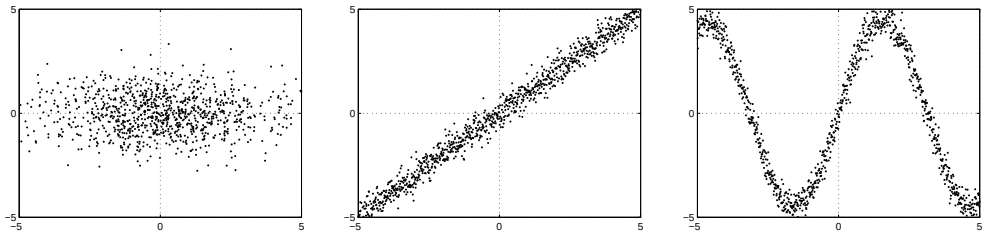
$$x_{t+1} = Ax_t \tag{1.1}$$

$$y_t = Cx_t \tag{1.2}$$

from the measured output $\{y_t, t = 1, 2, \ldots\}$. The third case also shows up in the *structure from motion* problem in computer vision, where one is supposed to estimate the *motion* (rotation and translation) of a camera observing a cloud of points in 3-D space from two perspective views $\{(\boldsymbol{x}_1^j, \boldsymbol{x}_2^j)\}_{j=1}^N$ of such points. The camera motion and the image points are related by the *epipolar constraint*

$$\boldsymbol{x}_2^{jT} F \boldsymbol{x}_1^j = 0, \tag{1.3}$$

where the so-called *fundamental matrix* $F$ is a rank-2 matrix encoding the motion parameters.



(a) Inferring the mean (b) Inferring a hyperplane (c) Inferring a manifold

Figure 1.1: Inferring a constant, linear and nonlinear model from a collection of data points.

When the equations relating the data points and the model parameters are linear on the latter, the inference problem becomes relatively simple and can be usually solved using tools from linear algebra and optimization, such as least squares. When the equations relating the data and the model parameters are nonlinear, the inference problem is more challenging and one needs to exploit the algebraic, geometric or statistical structure of the problem in order to render it tractable. For example, in the structure from motion problem one can exploit the fact that $F \in so(3) \times SO(3)$ to obtain a linear solution for the translation and rotation of the camera.

When the inference problem is not tractable, one can resort to some sort of approximation. The most natural approximation is to assume that the data is generated by a finite mixture of *simpler* (tractable) smooth sub-models. For example, in *intensity-based image segmentation*, one could model the image brightness as a piecewise constant function taking on a finite number of gray levels. The inference problem is that of estimating the number of the gray levels, their values, and the assignment of pixels to gray levels. A second example, which we will later call *generalized principal component analysis*, could be to approximate a manifold with a mixture of linear sub-models as illustrated in Figure 1.2. This case shows up in the face recognition example, where the images of multiple faces under varying illumination span multiple linear subspaces of a higher-dimensional space, and the task is to recognize how many faces are present in a given dataset and the subspace associated with each image. Similarly, one could think of approximating the nonlinear dynamics of an unmanned aerial vehicle (UAV) with a *linear hybrid system*, i.e., a mixture of linear dynamical sub-models of the type (1.1) and (1.2) connected by switches from one sub-model to the other. One could have, for example, a different linear sub-model for take off, landing, hovering, pirouette, etc., and would like to estimate such linear sub-models from measurements for the position, orientation and velocity of the UAV, without knowing which measurement corresponds to which linear sub-model.

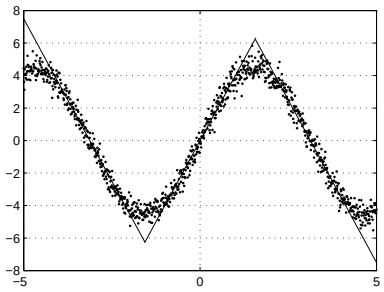



Figure 1.2: Approximating of a nonlinear manifold with a piecewise linear model.

However, there is a wide variety of inference problems in which using a mixture of sub-models is not merely a modeling assumption, but an intrinsic characteristic of the problem. Consider for example the problem of estimating the motion (translation and rotation) of multiple moving objects from a collection of image measurements collected by a moving perspective camera, i.e., the *multibody structure from motion* problem in computer vision. In this case, the objective is to find a collection of motion sub-models $\{F_i\}_{i=1}^n$ fitting a set of image measurements $\{(\boldsymbol{x}_1^j, \boldsymbol{x}_2^j)\}_{j=1}^N$, without knowing which sub-model $F_i$ corresponds to which measurement $(\boldsymbol{x}_1^j, \boldsymbol{x}_2^j)$ as illustrated in Figure 1.3.

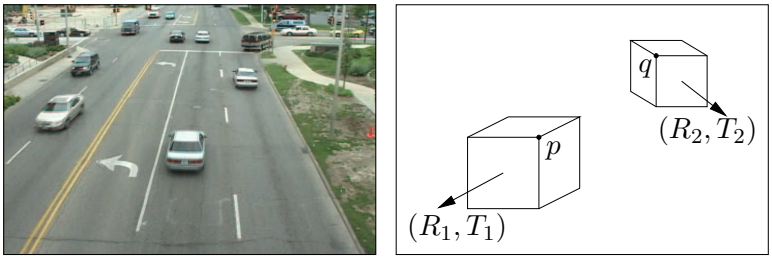



Figure 1.3: A traffic surveillance sequence with multiple moving objects. The motion of each object is represented with a different rotation and translation, $(R, T)$, relative to the camera frame.

In either case, a modeling assumption or an intrinsic characteristic of the problem, the estimation of a mixture of smooth sub-models from a collection of data points is a rather challenging problem, because one needs to simultaneously estimate

1. The number of sub-models in the mixture;
2. The parameters of each sub-model;
3. The segmentation of the data, i.e., the association between data points and sub-models.

It is important to notice that if the segmentation of the data was known, then the estimation of the parameters of each sub-model would be simple, because by assumption each sub-model is tractable. Conversely, if the parameters of each sub-model were known, then the segmentation of the data would be trivial, because one could just assign each point to the closest sub-model. Since in practice neither the model parameters nor the segmentation of the data are known, the estimation of a mixture model is usually though of as a "chicken-and-egg" problem: in order to estimate the sub-models one needs to first segment the data and in order to segment the data one needs to know the sub-model associated with each data point. The main challenge is then the simultaneous estimation of both the membership of each data point and the parameters of the sub-model for each class.

Statistical approaches to simultaneous data segmentation and model estimation assume that the data points are generated by a mixture of probabilistic sub-models. The problem is then equivalent to

1. Learning the number of sub-models and their parameters (e.g., mean and covariance);
2. Assigning points to sub-models based on the posterior probability of a point belonging to a sub-model.

However, the estimation of the mixture model is in general a hard problem which is usually solved using the Expectation Maximization (EM) algorithm [14]. The EM algorithm is an iterative procedure in which one first estimates the segmentation of the data given a prior on the parameters of each sub-model (E-step) and then maximizes the expected log-likelihood of the model parameters given a prior on the grouping of the data (M-step). The main disadvantage of this iterative procedure is that its convergence to the global optimum is in general very sensitive to initialization, because the complete log-likelihood function presents several local maxima. Furthermore, most iterative algorithms rely on prior knowledge about the number of sub-models to be estimated, and their performance deteriorates when the given number of sub-models is incorrect. One may therefore ask:

*Is there an algebraic way of initializing statistical approaches to data segmentation?*

Furthermore, since some information about the number of sub-models must also be contained in the data, we may ask

*Is there an algebraic way of obtaining an initial estimate for the number of sub-models?*

To our surprise, these questions have never been addressed in an analytic fashion. Most of the currently existing methods[1]

[1] We will provide a more detailed review of each one of these algorithms in the introduction section of each chapter.

1. Use a random initialization for the sub-model parameters.

2. Use some other iterative algorithm for initialization, such as K-means, that alternates between data segmentation and model estimation, also starting from a random initialization.

3. Use spectral clustering techniques which are based on thresholding the eigenvectors of a matrix whose $ij$ entry represents a measure of the similarity between points $i$ and $j$, the so-called *similarity matrix*. Questions such as which and how many eigenvectors to use? and how to combine those eigenvectors to obtain an initial segmentation? are still open problems.

4. Use some ad-hoc procedure that depends on the particular problem being solved. For example, in 2-D motion segmentation it is customary to fit a single affine motion model to the whole scene and then fit a second model to the outliers and so on.

In a sense, all these techniques attempt to do clustering first to then obtain an estimate of the sub-model parameters, and then iterate between these two stages. Therefore, none of them attempts to directly resolve the "chicken-and-egg" dilemma of clustering versus model estimation. In other words, none of them is able to estimate all the sub-models simultaneously using all the data, without previous knowledge about the segmentation of the data points.

According to [18], "It is hard to see that there could be a comprehensive theory of segmentation ... There is certainly no comprehensive theory of segmentation at time of writing ...".

## 1.2 Dissertation contributions

This thesis represents a first step towards our long term goal of developing a mathematical theory of data segmentation. In particular, we are interested in answering the following questions.

1. Are there classes of segmentation problems that can be solved analytically?

2. Under what conditions can these classes of segmentation problems be solved in closed form?

3. Under what conditions do these classes of segmentation problems have a unique solution?

4. Is there an algebraic formula for determining the number of sub-models?

In this thesis, we provide a complete answer to the last three questions for the following classes of segmentation problems (see Figure 1.4).

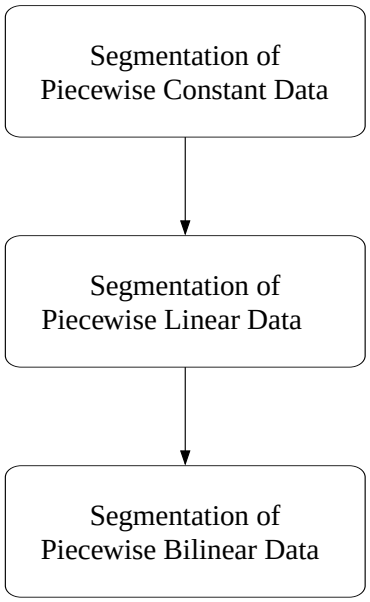
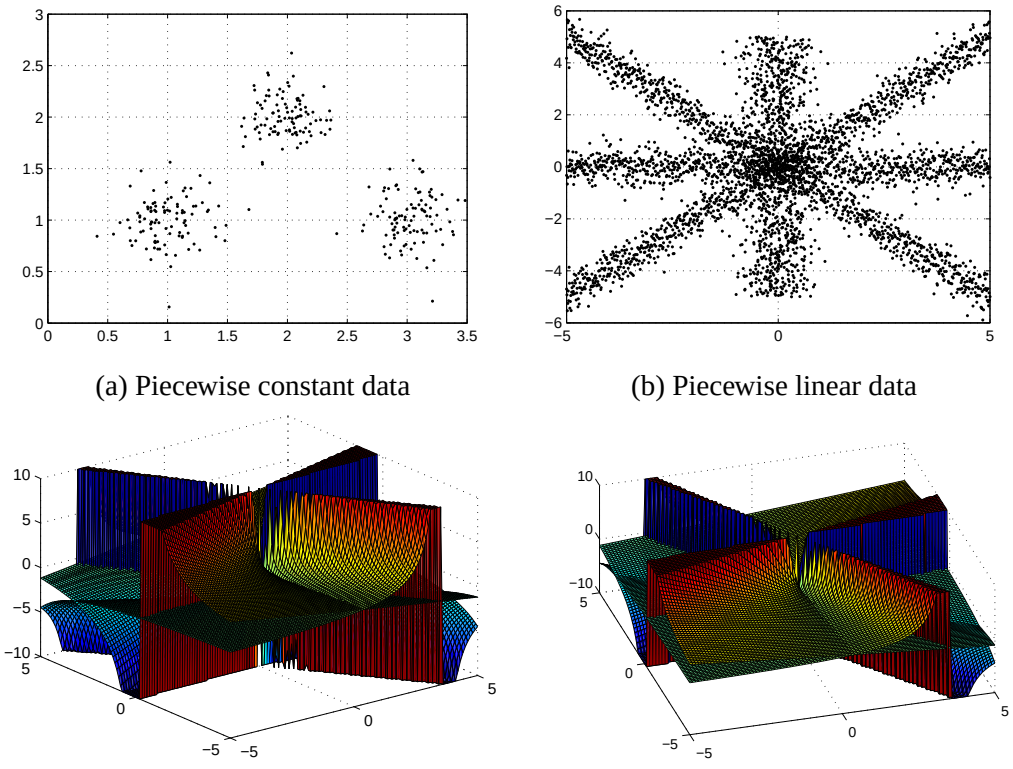



Figure 1.4: A hierarchy of segmentation problems.

1. **Piecewise constant data**: In this case, we assume that the data points are clustered around a finite collection of *cluster centers* as illustrated in Figure 1.5(a). This case shows up in a variety of applications in computer vision, including image segmentation problems based on intensity, texture, motion, etc. We will denote this case as *Polynomial Segmentation* (Polysegment), since our solution will be based on computing roots of univariate polynomials.

2. **Piecewise linear data**: In this case, we assume that the data points lie on a finite collection of linear subspaces, as illustrated in Figure 1.5(b) for the case of lines in $\mathbb{R}^2$. We will denote this case as *Generalized Principal Component Analysis* (GPCA), since it is a natural generalization of PCA [29], which is the problem of estimating a *single* linear subspace from sample data points. GPCA shows up in a variety of applications in computer vision, including vanishing point detection, segmentation of linearly moving objects, face recognition, etc.

3. **Piecewise bilinear data**: In this case, we assume that the data lies on a finite collection of manifolds with bilinear structure, i.e., the data points $(\boldsymbol{x}_1, \boldsymbol{x}_2)$ satisfy equations of the form $\boldsymbol{x}_2^T F \boldsymbol{x}_1 = 0$, where $F$ is a matrix representing the model parameters. We show an example of a mixture of two bilinear surfaces for $\boldsymbol{x}_1 \in \mathbb{R}^2$ and $\boldsymbol{x}_2 \in \mathbb{R}$ in Figures 1.5(c)-(d). We will denote this case as *Multibody Structure from Motion*, since it very much related to the 3-D motion segmentation problem in computer vision.



(a) Piecewise constant data

(b) Piecewise linear data

(c) Piecewise bilinear data

(d) Piecewise bilinear data

Figure 1.5: Inferring different piecewise smooth models from data.

The main contribution of this thesis is to show that for these three classes of segmentation problems the ”chicken-and-egg” dilemma can be completely solved using algebraic geometric techniques. In fact, it is possible to use all the data points simultaneously to recover all the model parameters without previously segmenting the data. In the absence of noise, this can be done in polynomial time using linear techniques and the solution can be obtained in closed form if and only if the number of groups is less than or equal to 4. In the presence of noise, the algebraic solution leads to an optimal objective function that depends on the model parameters and not on the segmentation of the data. Alternatively, the algebraic solution can be used as an initialization for any of the currently existing iterative techniques. Although these three classes of segmentation problems may seem quite different from each other, we will show that they are strongly related. In fact, we will show that the piecewise bilinear case can be reduced to a collection of piecewise linear problems. Similarly we will show that the piecewise linear case can be reduced to a collection of piecewise constant problems. The following sections give a more detailed account of our contributions for each class of data segmentation problems.

### 1.2.1 Piecewise constant data: polynomial segmentation

We propose a simple analytic solution to the segmentation of piecewise constant data and show that it provides a solution to the well known *eigenvector segmentation problem*. We start by analyzing the one-dimensional case and show that, in the absence of noise, one can determine the number of groups $n$ from a rank constraint on the data. Given $n$, the segmentation of the measurements can be obtained from the roots of a polynomial of degree $n$ in one variable. Since the coefficients of the polynomial are computed by solving a linear system, we show that there is a unique global solution to the one-dimensional segmentation problem, which can be obtained in closed form if and only if $n \leq 4$. This purely algebraic solution is shown to be robust in the presence of noise and can be used to initialize an optimal algorithm. We derive such an optimal objective function for the case of zero-mean Gaussian noise on the data points.

We then study the case of piecewise constant data in dimension two. We show that the *same* one-dimensional technique can be applied in the two-dimensional case after embedding the data into the complex plane. The only difference is that now the polynomial representing the data will have complex coefficients and complex roots. However, the cluster centers can still be recovered from the real and imaginary parts of the complex cluster centers. We then study the case of piecewise constant data in a higher-dimensional space and show that it can be reduced to a collection of one or two-dimensional clustering problems.

We present applications of polynomial segmentation on computer vision problem such as image segmentation based on intensity or texture, 2-D motion segmentation based on feature points, 3-D motion segmentation based on optical flow, and face clustering with varying expressions.

### 1.2.2 Piecewise linear data: generalized principal component analysis

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.

### 1.2.3 Piecewise bilinear data: multibody structure from motion

We present an algebraic geometric approach to segmenting static and dynamic scenes from image intensities (2-D motion segmentation) or feature points (3-D motion segmentation).

In the 2-D motion segmentation case, we introduce the *multibody affine constraint* as a geometric relationship between multiple affine motion models and the image intensities generated by them. This constraint is satisfied by all the pixels in the image, regardless of the motion model associated with each pixel, and combines all the motion parameters into a single algebraic structure, the so-called *multibody affine matrix*. Given the image data, we show that one can estimate the number of motion models from a rank constraint and the multibody affine matrix from a linear system. Given the multibody affine matrix, we show that the optical flow at each pixel can be obtained from the partial derivatives of the multibody affine constraint. Given the optical flow at each pixel, we show that the estimation of the affine motion models can be done by solving two GPCA problems. In the presence of noise, we derive an optimal algorithm for segmenting dynamic scenes from image intensities, which is based on minimizing the negative log-likelihood subject to all multibody affine constraints. Our approach is based solely on image intensities, hence it does not require feature tracking or correspondences. It is therefore a natural generalization of the so-called *direct methods* in single-body structure from motion to the case of multiple motion models.

In the 3-D motion segmentation case, we introduce the so-called *multibody epipolar constraint* and its associated *multibody fundamental matrix* as natural generalizations of the epipolar constraint and of the fundamental matrix to multiple moving objects. We derive a rank constraint on the image points from which one can estimate the number of independently moving objects as well as linearly solve for the multibody fundamental matrix. We prove that the epipoles of each independent motion lie exactly in the intersection of the left null space of the multibody fundamental matrix with the so-called Veronese surface. Given the multibody fundamental matrix, we show that the epipolar lines can be recovered from the derivatives of the multibody epipolar constraint and that the epipoles can be computed by applying GPCA to the epipolar lines. Given the epipoles and epipolar lines, the estimation of individual fundamental matrices becomes a linear problem. The segmentation of the data is then automatically obtained from either the epipoles and epipolar lines or from the fundamental matrices. In the presence of noise, we derive the optimal error function for simultaneously estimating all the fundamental matrices from a collection of feature points, without previously segmenting the image data. Our results naturally generalize the so-called *feature based* methods in single-body structure from motion to the case of multiple rigidly moving objects.

## 1.3 Thesis outline

This thesis is organized in the following four chapters.

- **Chapter 2, Polynomial Segmentation,** covers the segmentation of piecewise constant data. Section 2.2 covers the segmentation of one-dimensional data. This case is the simplest segmentation problem, yet it allows to illustrate most, if not all, the concepts of the overall theory presented in this thesis. Thus we recommend the reader to clearly understand all the details before jumping into the the remaining chapters. In spite of its simplicity, the one-dimensional case is strongly related with the spectral clustering techniques that we mentioned in the previous section. In fact, the solution to the one-dimensional case provides an automatic way of thresholding the eigenvectors of a similarity matrix. The generalization to higher-dimensions is covered in Sections 2.3 and 2.4 and is a straightforward extension of the one-dimensional case. Such an extension indeed provides a solution to the problem of simultaneously thresholding multiple eigenvectors, which is the basis for spectral clustering techniques.

- **Chapter 3, Generalized Principal Component Analysis (GPCA),** covers the segmentation of piecewise linear data, i.e., data lying on a collection of subspaces. Section 3.2 gives the basic formulation of the problem. Section 3.3 covers the case of subspaces of co-dimension one (hyperplanes), including the polynomial factorization (Section 3.3.2) and polynomial differentiation (Section 3.3.3) algorithms. Section 3.4 covers the case of subspaces of equal dimension, which is reduced to the case of hyperplanes via a projection. Section 3.5 covers the case of subspaces of arbitrary dimensions via polynomial differentiation and division. Sections 3.6 derives an optimal function for obtaining the subspaces from noisy data. Section 3.7 shows how to use GPCA to initialize iterative algorithms such as K-subspace and EM.

- **Chapters 4 and 5** extend the theory of Chapter 3 to the case of piecewise bilinear data. Although the segmentation of piecewise bilinear data can always be reduced to the segmentation of piecewise bilinear, the last step of the reduction is combinatorial. Therefore, we have chosen to concentrate on the problem of segmenting dynamic scenes from 2-D imagery, because in this case the combinatorial part can be bypassed by exploiting the geometric structure of the problem. Chapter 4 covers the segmentation of static and dynamic scenes from image intensities, and is a natural generalization of the so-called direct methods to the case of multiple motion models. Chapter 5 covers the segmentation of dynamic scenes from feature points, and is a natural generalization of the eight-point algorithm to multiple rigidly moving objects.

# Chapter 2

# Polynomial Segmentation (Polysegment)

## 2.1 Introduction

Eigenvector segmentation is one of the simplest and most widely used global approaches to segmentation and clustering [42, 11, 40, 45, 68]. The basic algorithm is based on thresholding the eigenvectors of the so-called *similarity matrix* and can be summarized as having the following steps [40]:

1. Associate to each data point a feature vector. Typical feature vectors in image segmentation are the pixel's coordinates, intensity, optical flow, output of a bank of filters, etc.

2. Form a *similarity* matrix $\mathcal{S} \in \mathbb{R}^{N \times N}$ corresponding to $N$ data points. Ideally $\mathcal{S}_{ij} = 1$ if points $i$ and $j$ belong to the same group and $\mathcal{S}_{ij} = 0$ otherwise. A typical choice for $\mathcal{S}_{ij}$ is $\exp(-d_{ij}^2/2\sigma^2)$, where $d_{ij}$ is *a* distance between the features associated to points $i$ and $j$ and $\sigma$ is a free parameter. $d_{ij}$ is chosen so that the intragroup distance is small and the intergroup distance is large. When the points are ordered according to which group they belong, the similarity matrix should be block diagonal as illustrated in Figure 2.1.

3. Group the points by thresholding an eigenvector $\boldsymbol{x} \in \mathbb{R}^N$ of the similarity matrix $\mathcal{S} \in \mathbb{R}^{N \times N}$. Ideally, if two points $i$ and $j$ belong to the same group, then $x_i = x_j$. Thus if the points are reordered according to which group they belong, the eigenvector should be a piecewise constant function of the points as illustrated in Figure 2.1.

In practice, the data points are corrupted with noise, the intragroup distance is nonzero and the intergroup distance is not infinity. This means that, in general, $x_i \neq x_j$ even if points $i$

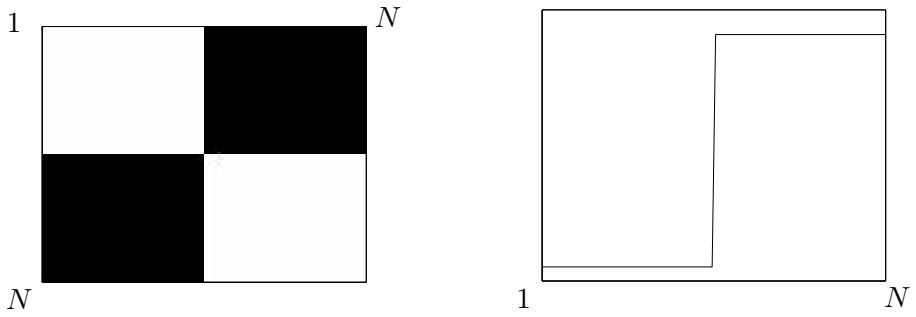



Figure 2.1: A similarity matrix for two groups (left) and its leading eigenvector (right), after segmentation. Dark regions represent $S_{ij} = 0$ and light regions represent $S_{ij} = 1$.

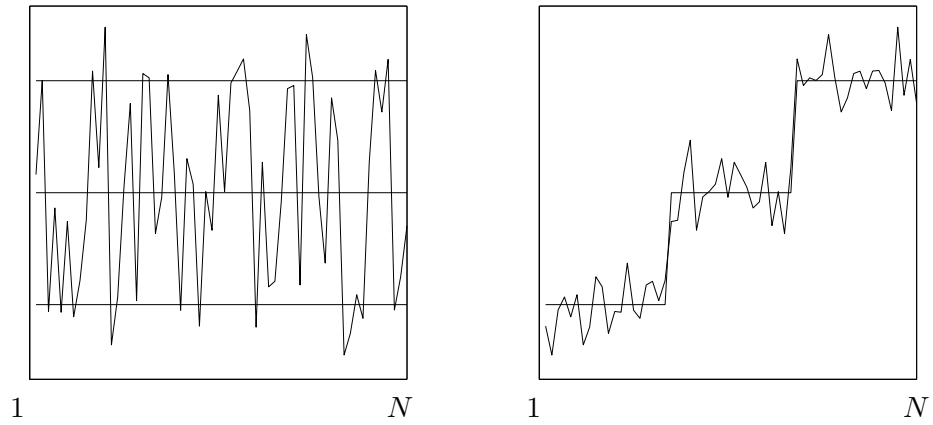



Figure 2.2: Eigenvector obtained from noisy image measurements and the "ideal" eigenvector. Before segmentation (left) and after segmentation (right).

and $j$ belong to the same group. We illustrate this phenomenon in Figure 2.2, where the leading eigenvector of $\mathcal{S}$ is not piecewise constant, yet there is an *unknown* underlying piecewise constant eigenvector: the "ideal" eigenvector. The question is

> *How does one recover the "ideal" eigenvector from the "noisy" one? Is there an analytic way of doing so?*

Furthermore, since information about the number of groups is also contained in the noisy eigenvector

> *How does one obtain an estimate of the number of groups from the noisy eigenvector?*

To our surprise, these questions have never been addressed in an analytic fashion. Most of the existing work (See Section 2.1.2 for a review) uses heuristics to threshold one or more eigenvectors of the similarity matrix and then extract the segmentation.

### 2.1.1 Contributions

In this chapter, we address the eigenvector segmentation problem in a simple algebraic geometric framework. We assume that the number of groups is unknown and that there exists a set of underlying "ideal" eigenvectors which are permutations of piecewise constant vectors. The problem then becomes one of estimating the number of groups, the "ideal" eigenvectors and the corresponding permutation from a set of "noisy" eigenvectors of $\mathcal{S}$. We propose to solve this problem using *polynomial segmentation (Polysegment)*, a simple technique that transforms each eigenvector into a univariate polynomial. The number of groups $n$ becomes the degree of the polynomial and the finite values that the "ideal" eigenvectors can take become the roots of the polynomial.

In Section 2.2 we consider the case of a *single* eigenvector. In Section 2.2.1 we derive a rank condition on the entries of the "ideal" eigenvector from which we determine the number of groups $n$. Once the number of groups has been determined, the segmentation of the data points can be obtained from the roots of a polynomial of degree $n$ in one variable, whose coefficients can be computed by solving a linear system. This shows that there is a unique global solution to the eigenvector segmentation problem, which can be obtained in closed form if and only if $n \leq 4$. In Section 2.2.2 we show that this purely algebraic solution is robust in the presence of noise since it corresponds to the least squares solution to the algebraic error derived in the ideal case. In the case of zero-mean Gaussian noise on the entries of the eigenvector, we show that such a sub-optimal objective function can be easily modified to obtain an optimal function for the chosen noise model.

In Section 2.3 we consider the problem of segmenting the data from two eigenvectors and show that Polysegment can be directly applied after transforming the two (real) eigenvectors into a complex one, and then working with complex polynomials. In Section 2.4 we study the case of multiple eigenvectors and show that it can be reduced to the case of one or two eigenvectors after a suitable projection. We show how to use Polysegment to initialize K-means and EM in Section 2.5.

In Section 2.6 we present experimental results on intensity-based image segmentation that show that Polysegment performs similarly or better than K-means and EM, but is computationally less costly, because it only needs to solve one linear system in $n$ variables plus one polynomial of degree $n$ in one variable. We also present experimental results on texture-based image segmentation that show that Polysegment is very efficient at computing and segmenting textures and produces a visually appealing segmentation of natural scenes from the Berkeley segmentation dataset. We then apply Polysegment to 2-D and 3-D motion segmentation using either point features or optical flow. Finally, we present experimental results on face clustering with varying expressions.

### 2.1.2 Previous work

Spectral clustering techniques were first applied to motion segmentation by Boult and Brown [7]. The authors propose a rank constraint to estimate the number of independent motions and obtain the segmentation of the image data from the leading singular vectors of the matrix of feature points in multiple frames. A similar technique was earlier proposed by Scott and Longuet-Higgins [42] in the context of feature segmentation. The authors assume that the number of groups $n$ is given and use the first $n$ eigenvectors of the similarity matrix $\mathcal{S}$ to build a segmentation matrix $\mathcal{Q}$ such that $\mathcal{Q}_{ij} = 1$ if pixels belong to the same group and zero otherwise. In the presence of noise, the segmentation of the data is obtained by thresholding $\mathcal{Q}$, which is sensitive to noise. The same technique was later applied by Costeira and Kanade [11] to orthographic motion segmentation. In this case the similarity matrix is obtained as the outer product of a matrix formed from a collection of feature points in multiple frames. Instead of thresholding $\mathcal{Q}$, the authors obtain the segmentation by partitioning a graph that is formed from the entries of $\mathcal{Q}$. An alternative approach to thresholding $\mathcal{Q}$ based on model selection techniques was proposed by Kanatani [31]. Shi and Malik [45] demonstrated that segmentation based on a single eigenvector can be interpreted as a sub-optimal solution of a two-way graph partitioning problem. They explored three algorithms for image segmentation. In the *two-way Ncut* they threshold the second eigenvector of a normalized similarity matrix into two groups. The choice of two groups is arbitrary, and can produce the wrong segmentation for eigenvectors such as the one in Figure 2.2. In the *recursive two-way Ncut* each one of the two groups is further segmented into two sub-groups by applying the two-way Ncut to the eigenvectors associated to the similarity matrices of the previously computed groups. In this case it is unclear when to stop subdividing currently computed groups. The authors also explore a *K-way Ncut* that uses $K$ eigenvectors. The $K$ entries corresponding to each pixel are used as feature vectors that are clustered using the K-means algorithm with random initialization. They do not provide an analytic way of initializing K-means. Weiss [68] showed that the eigenvector segmentation algorithms in [11, 40, 42, 45] are very much equivalent to each other. In some special cases, he also analyzed the conditions under which they should give a good segmentation. For example, the algorithm in [42] gives a good segmentation when the intergroup similarities are zero, the intragroup similarities are positive and the first eigenvalue of each intragroup similarity matrix is bigger than the second eigenvalue of any other. Similar conditions were derived in [39]. Unfortunately, these conditions depend on the spectral properties of the segmented data and hence they cannot be checked when the true segmentation is unknown.

## 2.2 One-dimensional clustering: the case of one eigenvector

Assume that we are given an eigenvector $\boldsymbol{x} \in \mathbb{R}^N$ of a similarity matrix $\mathcal{S} \in \mathbb{R}^{N \times N}$, where $N$ is the number of data points, and that we would like to segment the entries of $\boldsymbol{x}$ into an *unknown* number of groups $n$. We assume that there exists an (unknown) *ideal* eigenvector $\tilde{\boldsymbol{x}}$ that takes on a finite number of values, i.e., $\tilde{x}_j \in \{\mu_1, \mu_2, \ldots, \mu_n\}$, with $\mu_1 \neq \cdots \neq \mu_n$, for $j = 1, \ldots, N$. We define the *eigenvector segmentation problem* as follows.

**Problem 1 (Eigenvector segmentation problem)**

Given an eigenvector $\boldsymbol{x} \in \mathbb{R}^N$ of a similarity matrix $\mathcal{S} \in \mathbb{R}^{N \times N}$, estimate the number of groups $n$, the constants $\{\mu_i\}_{i=1}^n$, and the segmentation of the data, i.e., the group to which each point belongs.

### 2.2.1 The ideal case

Imagine for the time being that we had access to the ideal eigenvector $\tilde{\boldsymbol{x}}$. In this case, the segmentation problem can be trivially solved by sorting the entries of $\boldsymbol{x} = \tilde{\boldsymbol{x}}$. However, we will pretend as if we did not know the sorting-based solution so that we can derive the equations that $\boldsymbol{x}$ has to satisfy. It turns out that those equations are precisely the ones that will enable us to recover $\tilde{\boldsymbol{x}}$ from $\boldsymbol{x}$, when $\tilde{\boldsymbol{x}}$ is unknown.

Let $x \in \mathbb{R}$ be an indefinite variable representing say the $j^{th}$ entry of $\boldsymbol{x} \in \mathbb{R}^N$. Then, there exists a constant $\mu_i$ such that $x = \mu_i$. This means that

$$(x = \mu_1) \vee (x = \mu_2) \vee \cdots \vee (x = \mu_n), \tag{2.1}$$

which can be compactly written as the following polynomial of degree $n$ in $x$:

$$p_n(x) = \prod_{i=1}^{n} (x - \mu_i) = \sum_{k=0}^{n} c_k x^k = 0. \tag{2.2}$$

Since the above equation is valid for every entry of $\boldsymbol{x}$, we have that

$$L_n \, \boldsymbol{c} \doteq \begin{bmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^n \\ 1 & x_2 & x_2^2 & \cdots & x_2^n \\ \vdots & & & & \vdots \\ 1 & x_N & x_N^2 & \cdots & x_N^n \end{bmatrix} \begin{bmatrix} c_0 \\ \vdots \\ c_{n-1} \\ 1 \end{bmatrix} = 0. \tag{2.3}$$

where $L_n \in \mathbb{R}^{N \times (n+1)}$ is the data matrix and $\boldsymbol{c} \in \mathbb{R}^{n+1}$ is the vector of coefficients of $p_n(x)$.

In order for the linear system of equation (2.3) to have a unique solution for the vector of coefficients $\boldsymbol{c} \in \mathbb{R}^{n+1}$, we must have that rank$(L_n) = n$. This rank constraint on $L_n \in \mathbb{R}^{N\times(n+1)}$ provides a criterion to determine the number of groups $n$ from the eigenvector $\boldsymbol{x}$, as follows.

**Theorem 1 (Number of groups)** *Let $L_i \in \mathbb{R}^{N\times(i+1)}$ be the matrix formed from the first $i+1$ columns of $L_n$. If $N \geq n$, then*

$$\text{rank}(L_i) \begin{cases} > i, & \textit{if } i < n, \\ = i, & \textit{if } i = n, \\ < i, & \textit{if } i > n. \end{cases} \tag{2.4}$$

*Therefore, the number of groups $n$ is given by*

$$\boxed{n \doteq \min\{i : \text{rank}(L_i) = i\}.} \tag{2.5}$$

*Proof.* Consider the polynomial $p_n(x)$ as a polynomial over the algebraically closed field $\mathbb{C}$ and assume that $\mu_1 \neq \mu_2 \neq \cdots \neq \mu_n$. Then the ideal $I$ generated by $p_n(x)$ is a *radical ideal* with $p_n(x)$ as its only generator. According to Hilbert's Nullstellensatz (see page 380, [34]), there is a one-to-one correspondence between such an ideal $I$ and the algebraic set

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

associated to it. Hence its generator $p_n(x)$ is uniquely determined by points in this algebraic set. By definition, $p_n(x)$ has the lowest degree among all the elements in the ideal $I$. Hence no polynomial with lower degree would vanish on all points in $\{\mu_1, \mu_2, \ldots, \mu_n\}$. Furthermore, since all the constants $\mu_i$ are real, if $x + \sqrt{-1}y \in \mathbb{C}$ is in $Z(I)$, then $(x + \sqrt{-1}y) = \mu_i \Leftrightarrow (x = \mu_i) \ \wedge \ (y = 0)$. Hence all points on the (real) line determine the polynomial $p_n(x)$ uniquely and vice-versa. Since the coefficients of the polynomial $p_n(x)$ lie in the null space of $L_n$, and the rank of $L_n$ determines the number of solutions, it follows that the null space of $L_i$ is trivial if $i < n$, one-dimensional if $i = n$ and at least two-dimensional if $i > n$. This completes the proof. ∎

The intuition behind Theorem 1 can be explained as follows. Consider for simplicity the case of $n = 2$ groups, so that $p_n(x) = p_2(x) = (x - \mu_1)(x - \mu_2)$, with $\mu_1 \neq \mu_2$. Then it is clear that there is no polynomial of degree one, $p_1(x) = x - \mu$, that is satisfied by *all* the data. Similarly, there are infinitely many polynomials of degree 3 or more that are satisfied by all the data, namely any multiple of $p_2(x)$. Thus the degree $n = 2$ is the only one for which there is a unique polynomial representing all the data. Since the vector of coefficients $\boldsymbol{c} \in \mathbb{R}^{n+1}$ of the polynomial

$p_n(x)$ lies in the null space of $L_n$, and the rank of $L_n$ determines the number of solutions of the linear system in (2.3), the number of groups is determined as the degree for which the null space of $L_n$ is one-dimensional.

We can therefore use Theorem 1 to estimate the number of groups incrementally from equation (2.5), starting with $i = 1, 2, \ldots$, etc. Notice that the minimum number of points needed is $N \geq n$, which is *linear* on the number of groups.[1]

Once the number of groups $n$ has been computed, we can linearly solve for the vector of coefficients $\boldsymbol{c}$ from equation (2.3). In fact, after rewriting (2.3) as a (non-homogeneous) linear system with unknowns $[c_0, c_1, \ldots, c_{n-1}]^T$, the least squares solution for $[c_0, c_1, \ldots, c_{n-1}]^T$ can be obtained by solving the linear system

$$\begin{bmatrix} 1 & E[\boldsymbol{x}] & \cdots & E[\boldsymbol{x}^{n-1}] \\ E[\boldsymbol{x}] & E[\boldsymbol{x}^2] & \cdots & E[\boldsymbol{x}^{n-1}] \\ \vdots & & & \vdots \\ E[\boldsymbol{x}^{n-1}] & E[\boldsymbol{x}^n] & \cdots & E[\boldsymbol{x}^{2n-2}] \end{bmatrix} \begin{bmatrix} c_0 \\ \vdots \\ c_{n-1} \end{bmatrix} = - \begin{bmatrix} E[\boldsymbol{x}^n] \\ E[\boldsymbol{x}^{n+1}] \\ \vdots \\ E[\boldsymbol{x}^{2n-1}] \end{bmatrix}, \tag{2.6}$$

where $E[\boldsymbol{x}^k] \doteq \frac{1}{N}\sum_{j=1}^{N} x_j^k$ is the $k^{th}$ moment of the data. This shows that for a mixture of $n$ groups, it is enough to consider all the moments of the data up to degree $2n-1$.

Finally, since

$$p_n(x) = \prod_{i=1}^{n}(x - \mu_i) = \sum_{k=0}^{n} c_k x^k = 0, \tag{2.7}$$

given $n$ and $\boldsymbol{c}$ we can obtain $\{\mu_i\}_{i=1}^n$ as the $n$ roots of the polynomial $p_n(x)$. Given $\{\mu_i\}_{i=1}^n$, the segmentation is obtained by assigning point $j$ to group $i$ whenever $\mu_j = x_i$.

**Remark 1 (Solvability of roots of univariate polynomial)** *It is well-known from abstract algebra [34] that there is a closed form solution for the roots of univariate polynomials of degree $n \leq 4$. Hence, there is a closed form solution to the eigenvector segmentation problem for $n \leq 4$ as well.*

### 2.2.2 The general case

Let us now consider the case in which we are given a noisy eigenvector $\boldsymbol{x}$ whose ideal eigenvector $\tilde{\boldsymbol{x}}$ is unknown. As before, let $x$ be an indefinite variable representing say the $j^{th}$ entry of $\boldsymbol{x}$. Then, there exists a constant $\mu_i$ such that $x \approx \mu_i$, hence we must have

$$p_n(x) = (x - \mu_1)(x - \mu_2)\cdots(x - \mu_n) = \sum_{k=0}^{n} c_k x^k \approx 0. \tag{2.8}$$

[1] We will see in future chapters that this is no longer the case for more general segmentation problems.

By applying the above equation to each entry of $\boldsymbol{x}$, we obtain the system of linear equations

$$L_n \boldsymbol{c} \approx 0, \tag{2.9}$$

where $L_n \in \mathbb{R}^{N \times (n+1)}$ is defined in (2.3). We can solve this equation in a least-squares sense by minimizing the algebraic error

$$E_A(\boldsymbol{c}) \doteq \sum_{j=1}^{N} \left(p_n(x_j)\right)^2 = \sum_{j=1}^{N} \left( \sum_{k=0}^{n} c_k x_j^k \right)^2 = ||L_n \boldsymbol{c}||^2. \tag{2.10}$$

The solution $\boldsymbol{c}$ to the above problem is simply the singular vector of $L_n$ corresponding to the smallest singular value. Given $\boldsymbol{c}$, the cluster centers $\{\mu_i\}_{i=1}^n$ can be obtained as the $n$ roots of $p_n(x)$. Finally, given $\{\mu_i\}_{i=1}^n$, the segmentation of the data is obtained by assigning point $j$ to the group $i$ that minimizes the distance between $x_j$ and $\mu_i$, i.e., point $j$ is assigned to the group

$$i = \arg \min_{\ell=1,\ldots,n} (x_j - \mu_\ell)^2. \tag{2.11}$$

In summary, if the number of groups $n$ is given, then *the same* algorithm that we derived in the ideal case can be directly applied to compute the vector of coefficients $\boldsymbol{c}$, the cluster centers $\{\mu_i\}_{i=1}^n$ and the segmentation of the data. Now if the number of groups $n$ is unknown, we cannot directly compute it from the rank condition in (2.5), because the matrix $L_i$ may be full rank for any $i \geq 1$. Therefore, we determine the number of groups by thresholding the singular values of the data matrix. That is, we estimate the number of groups as

$$n = \min\{i : \sigma_{i+1}/\sigma_i < \epsilon\}, \tag{2.12}$$

where $\sigma_i$ is the $i^{th}$ singular value of $L_i$ and $\epsilon$ is a pre-specified threshold that depends on the noise level. One can also use the geometric information criterion to estimate the rank as shown in [32].

Even though we have derived the polynomial segmentation algorithm Polysegment in a purely algebraic setting, it turns out that it also has a probabilistic interpretation. Let $\{x_j\}_{j=1}^N$ be a noise corrupted version of the ideal data $\{\tilde{x}_j\}_{j=1}^N$ drawn from a mixture model with means $\{\mu_i\}_{i=1}^n$. The problem is then to estimate the means of the mixture model $\{\mu_i\}_{i=1}^n$ from the noisy sample data $\{x_j\}_{j=1}^N$. The following lemma [54] shows that the algebraic solution described above is exactly the moment estimator for certain types of distributions, e.g., Exponential and Gamma.

**Lemma 1 (Moment estimator for mixtures of scalar random variables)** *Given a collection of points $\{x_j\}_{j=1}^N$ drawn from a mixture model with means $\{\mu_i\}_{i=1}^n$, if the probability distribution for group $i$ is such that $E(x^k) = \mu_i^k$ for all $i = 1, \ldots, n$ and for all $k \geq 1$, then the solution for $\{\mu_i\}_{i=1}^n$ given by (2.6) and (2.7) corresponds to the moment estimator for the means of the mixture model.*

Consider now the case in which the data $\{x_j\}_{j=1}^N$ is corrupted with i.i.d. zero-mean Gaussian noise. Since this case does not satisfy the conditions of Lemma 1, the algebraic solution is not necessarily optimal in a maximum likelihood sense. We therefore seek an optimal solution by solving the following constrained optimization problem

$$\min_{c_0,\ldots,c_{n-1}} \quad \sum_{j=1}^{N} (\tilde{x}_j - x_j)^2 \tag{2.13}$$

$$\text{subject to} \quad \sum_{k=0}^{n} c_k \tilde{x}_j^k = 0, \quad j = 1, ..., N. \tag{2.14}$$

Since $p_n(\tilde{x}) = p_n(x) + p_n'(x)(\tilde{x} - x) + O((\tilde{x} - x)^2)$ and $(x - \tilde{x})$ is assumed to be zero-mean and Gaussian, after neglecting higher order terms an optimal objective function can be obtained by minimizing

$$E_O(\boldsymbol{c}) \doteq \sum_{j=1}^{N} \left( \frac{p_n(x_j)}{p_n'(x_j)} \right)^2 = \sum_{j=1}^{N} \left( \frac{\sum_{k=0}^{n} c_k x_j^k}{\sum_{k=1}^{n} k c_k x_j^{k-1}} \right)^2. \tag{2.15}$$

Minimizing $E_O(\boldsymbol{c})$ is an unconstrained optimization problem in $n$ variables, which can be solved with standard optimization techniques. Notice that the optimal error $E_O(\boldsymbol{c})$ is just a normalized version of the algebraic error $E_A(\boldsymbol{c})$. Given $n$ and $\boldsymbol{c}$ we obtain the constants $\{\mu_i\}_{i=1}^n$ as before, i.e., they are the $n$ roots of the polynomial $p_n(x)$. Given the constants $\{\mu_i\}_{i=1}^n$, the segmentation of the data is obtained as in (2.11).

**Remark 2 (Solving for $\{\mu_i\}_{i=1}^n$ directly)** *Notice that in the nonlinear case it is not necessary to solve for $\boldsymbol{c}$ first. Instead one can define the optimal error $E_O$ as a function of $\{\mu_i\}_{i=1}^n$ directly, because $p_n(x_j) = (x_j - \mu_1) \cdots (x_j - \mu_n)$. The error becomes*

$$\sum_{j=1}^{N} \left( \frac{p_n(x_j)}{p_n'(x_j)} \right)^2 = \sum_{j=1}^{N} \left( \frac{\prod_{i=1}^{n} (x_j - \mu_i)}{\sum_{i=1}^{n} \prod_{\ell \neq i} (x_j - \mu_\ell)} \right)^2. \tag{2.16}$$

*In the presence of noise is better to search for $\{\mu_i\}_{i=1}^n$ directly, without computing $\boldsymbol{c}$ first. This is because the unconstrained minimization of $E_O(\boldsymbol{c})$ does not consider the constraints on the entries of $\boldsymbol{c}$ associated to the fact that $p_n(x)$ should have real roots.*

**Remark 3 (Approximate distance from a point to its cluster center)** *Notice from (2.16) that if a point $x_j$ is close to cluster center $\mu_i$, then the denominator is approximately equal to $\prod_{\ell \neq i} (x_j - \mu_\ell)$. After dividing the numerator by the denominator, we notice that the contribution of point $j$ to the error $E_O(\boldsymbol{c})$ is equal to $(x_j - \mu_i)^2$. Therefore, the error function $E_O(\boldsymbol{c})$ is a clever way of writing the sum of the square distances from each point to its own cluster center, modulo higher order terms.*

## 2.3 Two-dimensional clustering: the case of two eigenvectors

Consider now the case in which we are given eigenvectors $\boldsymbol{x}_1 \in \mathbb{R}^N$ and $\boldsymbol{x}_2 \in \mathbb{R}^N$ of a similarity matrix $\mathcal{S} \in \mathbb{R}^{N \times N}$. As before, the objective is to find two ideal eigenvectors $\tilde{\boldsymbol{x}}_1 \in \mathbb{R}^N$ and $\tilde{\boldsymbol{x}}_2 \in \mathbb{R}^N$ such that the rows of the matrix $\tilde{X} = [\tilde{\boldsymbol{x}}_1 \quad \tilde{\boldsymbol{x}}_2] \in \mathbb{R}^{N \times 2}$ take on finitely many values $\{\boldsymbol{\mu}_i \in \mathbb{R}^2\}_{i=1}^n$. Alternatively, we can interpret the above problem as a clustering problem in $\mathbb{R}^2$. We could imagine that each row of the data matrix $X = [\boldsymbol{x}_1 \quad \boldsymbol{x}_2] \in \mathbb{R}^{N \times 2}$ is a data point to be clustered and that $\{\boldsymbol{\mu}_i\}_{i=1}^n$ are the (unknown) *cluster centers*.

We now show that the two-eigenvector problem can be solved using the *same* technique we used in the single-eigenvector case, i.e., polynomial segmentation, except that we need to use complex coordinates. To this end, let us interpret the cluster centers as a collection of complex numbers $\{\boldsymbol{\mu}_i \in \mathbb{C}\}_{i=1}^n$ and let $\boldsymbol{z} = \boldsymbol{x}_1 + \sqrt{-1}\boldsymbol{x}_2 \in \mathbb{C}^N$ be a new (complex) eigenvector. Then each coordinate $z \in \mathbb{C}$ of the (noisy) eigenvector $\boldsymbol{z} \in \mathbb{C}^N$ must approximately satisfy the polynomial

$$p_n(z) = \prod_{i=1}^{n}(z - \boldsymbol{\mu_i}) = \sum_{k=0}^{n} c_k z^k = 0 \tag{2.17}$$

As before, by applying the above equation to each one of the $N$ entries of $\boldsymbol{z}$ we obtain the following linear system on the vector of (complex) coefficients $\boldsymbol{c} \in \mathbb{C}^{n+1}$

$$L_n \, \boldsymbol{c} = 0, \tag{2.18}$$

where $L_n \in \mathbb{C}^{N \times (n+1)}$ is defined similarly to (2.3), but computed from the complex eigenvector $\boldsymbol{z}$. We can now solve for $\boldsymbol{c}$ in a least-squares sense from the SVD of $L_n$. Given $\boldsymbol{c}$, we compute the $n$ roots of $p_n(z)$, which correspond to the $n$ cluster centers in $\mathbb{R}^2$ $\{\boldsymbol{\mu}_i\}_{i=1}^n$. The clustering of the data is then obtained by assigning each row of $X$ to the closest cluster center, similarly to (2.11).

**Remark 4 (A difference between one-dimensional and two-dimensional cases)** *Although the one-dimensional and two-dimensional cases are conceptually identical, in the noisy case there is a potential difference that is worth mentioning. In the one-dimensional case we are dealing with polynomials in* $\mathbb{R}$*, and* $\mathbb{R}$ *is not an algebraically closed field. Therefore the roots of* $p_n(x)$ *may not be all* real*, because we never enforced that when solving for the vector of coefficients* $\boldsymbol{c}$ *from* $L_n\boldsymbol{c} = 0$*. In the two-dimensional case, on the other hand, we are working in* $\mathbb{C}$ *which is an algebraically closed field, hence all the roots are complex and there is no need to constraint the roots of* $p_n(z)$ *when solving for* $\boldsymbol{c}$*. However this difference is only conceptual. In practice one always gets real solutions in the one-dimensional case. For example, if* $n = 2$ *one can solve for* $\boldsymbol{c}$ *from (2.6) and show that* $c_2 = Var[\boldsymbol{x}]$*,* $c_1 = E[\boldsymbol{x}^2]E[\boldsymbol{x}] - E[\boldsymbol{x}^3]$ *and* $c_0 = E[\boldsymbol{x}^3]E[\boldsymbol{x}] - E[\boldsymbol{x}^2]^2 \leq 0$*, hence* $c_1^2 - 4c_0c_2 \geq 0$*.*

## 2.4 K-dimensional clustering: the case of multiple eigenvectors

In many segmentation problems a single eigenvector will not be enough for obtaining the correct segmentation. We illustrate this in Figure 2.3, where individual eigenvectors contain two groups, while all three eigenvectors contain three groups.

In this section, we generalize Polysegment to deal simultaneously with multiple (noisy) eigenvectors[2] $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_K \in \mathbb{R}^N$ of a similarity matrix $\mathcal{S} \in \mathbb{R}^{N \times N}$. The objective is to find a collection of ideal eigenvectors $\tilde{\boldsymbol{x}}_1, \tilde{\boldsymbol{x}}_2, \ldots, \tilde{\boldsymbol{x}}_K \in \mathbb{R}^N$ such that the rows of the matrix of ideal eigenvectors $\tilde{X} = [\tilde{\boldsymbol{x}}_1, \tilde{\boldsymbol{x}}_2, \ldots, \tilde{\boldsymbol{x}}_K] \in \mathbb{R}^{N \times K}$ take on finitely many values $\{\boldsymbol{\mu}_i \in \mathbb{R}^K\}_{i=1}^n$. As before, we can interpret the multiple eigenvector segmentation problem as a clustering problem in $\mathbb{R}^K$ in which we would like to cluster the rows of $X = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_K] \in \mathbb{R}^{N \times K}$ around the cluster centers $\{\boldsymbol{\mu}_i \in \mathbb{R}^K\}_{i=1}^n$.

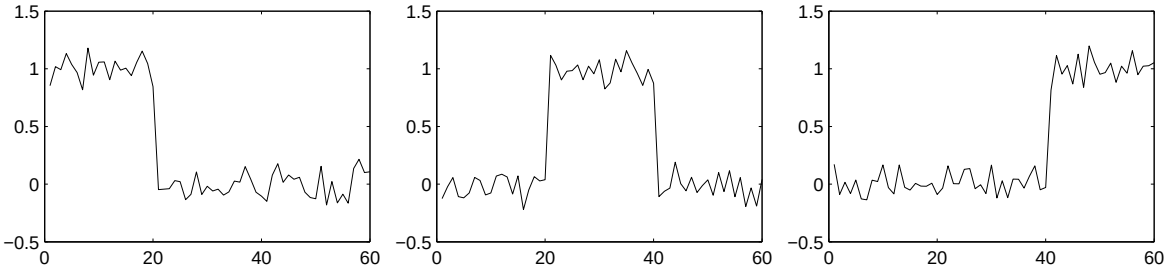



Figure 2.3: A case in which individual eigenvectors contain two groups, while all three eigenvectors contain three groups.

In principle, one may wonder if it is possible to solve the case of $K > 2$ eigenvectors by applying the same trick of the $K = 2$ case, namely to identify $\mathbb{R}^2$ with the complex plane. Unfortunately, one cannot directly generalize the properties of complex numbers to higher dimensions. In the case $K = 4$, for example, one could think that working with quaternions could be the solution. However, unlike the multiplication of complex numbers, the multiplication of quaternions is not commutative. Furthermore, it is unclear how to solve for the roots of a polynomial on quaternions.

Therefore, in solving the case $K > 2$ we will look for an alternative solution based on reducing the problem to the cases $K = 1$ and/or $K = 2$. To this end, notice that the case $K = 2$ can be reduced to the case $K = 1$ by projecting the rows of $X \in \mathbb{R}^{N \times 2}$ onto a one-dimensional subspace. We illustrate this in Figure 2.4, where three clusters are projected onto the horizontal axis,

[2] In general, it is enough to use $K = \text{rank}(\mathcal{S})$ eigenvectors. Thus $K$ can be obtained by choosing the eigenvectors of $\mathcal{S}$ that are such that the corresponding eigenvalues are above a certain threshold.

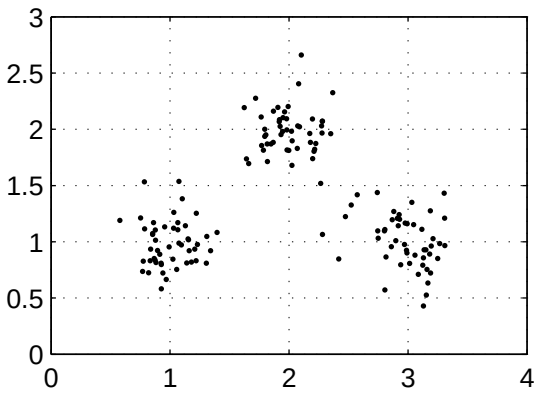



Figure 2.4: Projecting the rows of $X \in \mathbb{R}^{N\times 2}$ onto any line not perpendicular to the lines passing through the cluster centers preserves the segmentation. In this example, one can project onto the horizontal axis, but not onto the vertical axis.

and the segmentation of the data is preserved. More generally, given a matrix with $K$ eigenvectors $X \in \mathbb{R}^{N\times K}$, we can project its rows onto almost every[3] one-dimensional subspace of $\mathbb{R}^K$ and the segmentation into $n$ clusters is preserved. Since choosing a particular projection is equivalent to choosing a vector $\lambda \in \mathbb{R}^K$ and defining a new (projected) data set $X\lambda \in \mathbb{R}^N$, one can now apply Polysegment with $K = 1$ to the new *single* eigenvector $\boldsymbol{x} = X\lambda$ and obtain the segmentation of the data. Similarly, we can choose a matrix $\Lambda \in \mathbb{R}^{K\times 2}$ and project the rows of $X$ onto a 2-dimensional subspace to obtain two eigenvectors $X\Lambda \in \mathbb{R}^{N\times 2}$. We can then apply the Polysegment with $K = 2$ to the data, by embedding it into the complex plane.

In order to make the segmentation less dependent on a particular choice of the projection, we choose a collection of projections. For example, we can choose to project along each one of the axis in $\mathbb{R}^K$, which gives the original eigenvectors $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_K$.[4] Then one can apply polynomial segmentation with $K = 1$ to each one of them to obtain their corresponding "ideal" eigenvectors $\tilde{\boldsymbol{x}}_1, \tilde{\boldsymbol{x}}_2, \ldots, \tilde{\boldsymbol{x}}_K \in \mathbb{R}^N$. Since the entries of each $\tilde{\boldsymbol{x}}_j \in \mathbb{R}^N$ take on at most $n$ different values, many of the rows of $\tilde{X} = [\tilde{\boldsymbol{x}}_1, \tilde{\boldsymbol{x}}_2, \ldots, \tilde{\boldsymbol{x}}_K] \in \mathbb{R}^{N\times K}$ will be repeated. In fact, the $n$ different rows of $\tilde{X}$ should correspond to the cluster centers $\{\boldsymbol{\mu}_i\}_{i=1}^n$. Therefore, the segmentation of the data can be achieved by sorting the rows of $\tilde{X}$.

Algorithm 1 summarizes the overall algorithm.

[3] Except when the one-dimensional subspace is perpendicular to the line connecting any pair of cluster centers.

[4] Notice that projecting along one of the axis may not preserve the segmentation of the data, as illustrated in Figure 2.4 However, at least one of the $r$ projections has to preserve the segmentation of the data into $n$ groups, otherwise the number of groups is less than $n$.

**Algorithm 1 (Polysegment: Polynomial segmentation algorithm)**

Let $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_K \in \mathbb{R}^N$ be a given collection of eigenvectors of a similarity matrix $\mathcal{S} \in \mathbb{R}^{N \times N}$. Alternatively, let the $N$ rows of $X = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_K] \in \mathbb{R}^{N \times K}$ be a set of points in $\mathbb{R}^K$ to be clustered.

1. For $\ell = 1, \ldots, K$
   (a) Compute the number of groups $n_\ell \leq n$ for eigenvector $\boldsymbol{x}_\ell$ from (2.12).
   (b) Given $n_\ell$, compute the vector of coefficients $\boldsymbol{c} \in \mathbb{R}^{n_\ell + 1}$ from the linear system (2.3).
   (c) Given $\boldsymbol{c}$, compute the roots $\mu_1, \ldots, \mu_{n_\ell}$ of the polynomial $\sum_{k=0}^{n_\ell} c_k x^k$.
   (d) Compute the $j^{th}$ entry of the "ideal" eigenvector $(\tilde{\boldsymbol{x}}_\ell)_j$ as $\arg\min_{\{\mu_i\}}((\boldsymbol{x}_\ell)_j - \mu_i)^2$.
2. Set the number of groups $n$ as the number of distinct rows in the matrix of ideal eigenvectors $\tilde{X} = [\tilde{\boldsymbol{x}}_1, \ldots, \tilde{\boldsymbol{x}}_r] \in \mathbb{R}^{N \times K}$.
3. Set $\{\boldsymbol{\mu}_i \in \mathbb{R}^K\}_{i=1}^n$ to be the $n$ different rows in $\tilde{X}$.
4. Sort the rows of $\tilde{X}$ according to $\{\boldsymbol{\mu}_i\}_{i=1}^n$ to obtain the segmentation of the $N$ data points.

**Remark 5 (Multiple cues)** *Notice that the multiple eigenvector algorithm can be naturally used to simultaneously incorporate many cues. In image segmentation, for example, one could have one similarity matrix for each cue (motion, intensity, color, texture, etc.) and obtain their eigenvectors. Then Polysegment can be applied to the collection of all eigenvectors obtained from all cues. Alternatively, one can combine the individual similarity matrices into a single matrix, as proposed in [45], and then apply Polysegment to the eigenvectors of the combined similarity matrix.*

**Remark 6 (Number of groups)** *According to Algorithm 1, in the absence of noise the number of groups $n$ contained in all the eigenvectors will be given by the number of distinct rows in the matrix $\tilde{X}$. In fact, except for the degenerate case mentioned in footnote 3, each column of $\tilde{X}$ should give the correct number of groups. In the presence of noise, however, each individual eigenvector will provide a possibly different segmentation of the data, hence the number of different rows in $\tilde{X}$ will be much larger than the number of different entries in each column of $\tilde{X}$. Therefore, Algorithm 1 will tend to overestimate the number of groups and some post-processing will be needed to reduce the number of groups. In Section 2.6.2 we will discuss a particular strategy for reducing the number of groups in the case of texture-based image segmentation.*

## 2.5 Initialization of iterative algorithms in the presence of noise

In this section, we briefly describe two iterative algorithms for piecewise constant data segmentation and model estimation,[5] K-means and Expectation Maximization (EM), and show how to use Polysegment to initialize them. Since both K-means and EM can be applied to the segmentation of multiple eigenvectors, as in the previous section, we assume that we are given a matrix $X = [\boldsymbol{x}_1, \dots, \boldsymbol{x}_K] \in \mathbb{R}^{N \times K}$ containing $K$ eigenvectors. We will denote the $j^{th}$ row of $X$ as $\boldsymbol{y}_j \in \mathbb{R}^K$ and consider it as one of the data points to be clustered. Also we let $\{\boldsymbol{\mu}_i \in \mathbb{R}^K\}_{i=1}^n$ be the collection of cluster centers.

### 2.5.1 The K-means algorithm

The K-means algorithm minimizes a weighted square distance from point $\boldsymbol{y}_j$ to the cluster center $\boldsymbol{\mu}_i$ which is defined as

$$\sum_{j=1}^{N} \sum_{i=1}^{n} w_{ij} \|\boldsymbol{y}_j - \boldsymbol{\mu}_i\|^2, \tag{2.19}$$

where the weights $w_{ij}$ represent the membership of the data to each one of the clusters. The K-means algorithm starts by initializing the cluster centers, which can be done randomly or by choosing a subset of the data points $\{\boldsymbol{y}_j\}_{j=1}^N$. Then, the algorithm minimizes the error function (2.19) using a coordinate descent algorithm that iterates between two steps. In the first step it minimizes over $\{w_{ij}\}$ with $\{\boldsymbol{\mu}_i\}$ held constant, which gives the following formula for the weights

$$w_{ij} = \begin{cases} 1 & i = \arg\min_{\ell=1,\dots,n} \|\boldsymbol{y}_j - \boldsymbol{\mu}_\ell\|^2 \\ 0 & \text{otherwise} \end{cases}. \tag{2.20}$$

In the second step it minimizes over the cluster centers $\{\boldsymbol{\mu}_i\}_{i=1}^n$ with the weights $\{w_{ij}\}$ held constant, which leads to the following formula for the cluster centers

$$\boldsymbol{\mu}_i = \frac{\sum_{j=1}^N w_{ij} \boldsymbol{y}_j}{\sum_{i=1}^N w_{ij}}. \tag{2.21}$$

Notice that the Polysegment algorithm also gives an estimate of the cluster centers $\{\boldsymbol{\mu}_i\}_{i=1}^n$, but it does not require initialization. One can therefore use the solution of Polysegment to initialize K-means, thus replacing the random initialization of the cluster centers.

[5] We refer the readers to [30] for more details.

### 2.5.2 The Expectation Maximization algorithm

For the EM algorithm, we assume that the data points are generated by firstly choosing one of the clusters according to a multinomial distribution with parameters $\{0 \leq \pi_i \leq 1\}_{i=1}^n$, $\sum_{i=1}^n \pi_i = 1$, and secondly choosing a point $\boldsymbol{y}_j$ from one of the clusters, say cluster $i$, according to a Gaussian distribution with mean $\boldsymbol{\mu}_i \in \mathbb{R}^K$ and covariance $\sigma_i^2 I \in \mathbb{R}^{K \times K}$. Let $z_{ij} = 1$ denote the event that point $j$ corresponds to cluster $i$. Then the complete log-likelihood (neglecting constant factors) on both the data $\boldsymbol{y}_j$ and the latent variables $z_{ij}$ is given by

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

The EM algorithm maximizes the complete log-likelihood using a coordinate ascent algorithm that iterates between the following two steps, starting from an initial estimate for the model parameters $\{(\boldsymbol{\mu}_i, \sigma_i, \pi_i)\}_{i=1}^n$.

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

$$w_{ij} \doteq E[z_{ij}|\boldsymbol{y}_j, \theta] = P(z_{ij} = 1|\boldsymbol{y}_j, \theta) = \frac{\frac{\pi_i}{\sigma_i} \exp(-\frac{\|\boldsymbol{y}_j - \boldsymbol{\mu}_i\|^2}{2\sigma_i^2})}{\sum_{i=1}^n \frac{\pi_i}{\sigma_i} \exp(-\frac{\|\boldsymbol{y}_j - \boldsymbol{\mu}_i\|^2}{2\sigma_i^2})}.$$

Then the expected complete log-likelihood is given by

$$\sum_{j=1}^{N} \sum_{i=1}^{n} w_{ij}(\log(\pi_i) - \log(\sigma_i)) - w_{ij} \frac{\|\boldsymbol{y}_j - \boldsymbol{\mu}_i\|^2}{2\sigma_i^2}.$$

**M-step: Maximizing the expected log-likelihood.** The Lagrangian for $\pi_i$ 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 first order condition for $\boldsymbol{\mu}_i$ is

$$\sum_{j=1}^{N} w_{ij}(\boldsymbol{y}_j - \boldsymbol{\mu}_i) = 0 \quad \Rightarrow \quad \boldsymbol{\mu}_i = \frac{\sum_{j=1}^N w_{ij} \boldsymbol{y}_j}{\sum_{i=1}^N w_{ij}}.$$

Finally, after taking derivatives of the expected log-likelihood with respect to $\sigma_i$ one obtains

$$\sigma_i^2 = \frac{\sum_{j=1}^N w_{ij} \|\boldsymbol{y}_j - \boldsymbol{\mu}_i\|^2}{\sum_{j=1}^N w_{ij}}.$$

If all clusters have the same covariance, i.e., if for all $i = 1, \ldots, n$ we have $\sigma_i = \sigma$, then we have

$$\sigma^2 = \frac{\sum_{i=1}^{n} \sum_{j=1}^{N} w_{ij} \|\boldsymbol{y}_j - \boldsymbol{\mu}_i\|^2}{N}.$$

Therefore, from an optimization point of view, the K-means and EM algorithms are very similar (at least in the case of Gaussian noise for EM). They both minimize a weighted distance function, and the only difference is in the computation of the weights. While the K-means algorithm uses a "hard" assignment $w_{ij} \in \{0, 1\}$, the EM algorithm uses a "soft" assignment $w_{ij} \in [0, 1]$. However, from an statistical point of view the K-means algorithm does not have any probabilistic model in mind, while the EM algorithm can be shown to converge to a local maxima of the log-likelihood. However, convergence to the global maximum is not guaranteed and depends on initialization. Since the Polynomial Segmentation (Polysegment) algorithm proposed in the previous section does *not* need initialization, it can be naturally used to to initialize the cluster centers $\{\boldsymbol{\mu}_i\}_{i=1}^{n}$ in K-means, EM, or any other iterative algorithm.

## 2.6 Applications of Polysegment in computer vision

In this section, we present examples of the application of Polysegment to various problems in computer vision, such as image segmentation based on intensity and texture, 2-D and 3-D motion segmentation, and face clustering with varying expressions.

### 2.6.1 Image segmentation based on intensity

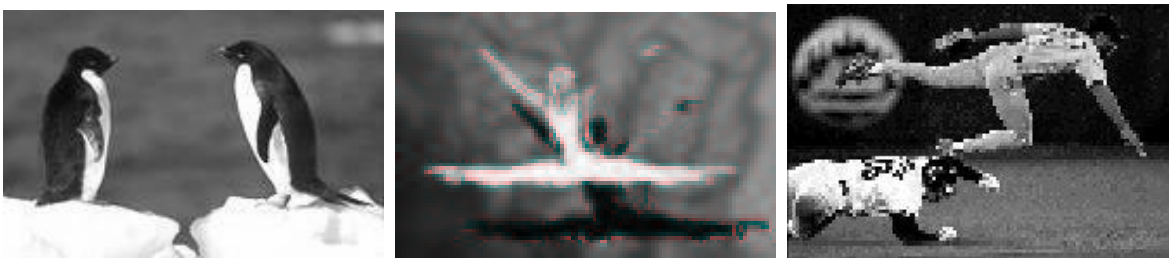
Image segmentation refers to the problem of separating the pixels of an image (or those of an image sequence) into a collection of groups, where each group is defined by similarity of one or more of a collection of features such as intensity, motion, texture, color, etc.

In this section, we apply Polysegment to the problem of segmenting an image based on intensity. Instead of computing the eigenvectors of the standard similarity matrix defined by

$$\mathcal{S}_{ij} = \exp\left(-(I_i - I_j)^2 / 2\sigma^2\right), \tag{2.22}$$

where $I_i$ is the intensity of pixel $i$, we apply Polysegment directly to the image intensities. That is, we form a *single* vector $\boldsymbol{x} \in \mathbb{R}^N$, where $N$ is the number of pixels, with its $j^{th}$ entry defined as $x_j = I_j$, for $j = 1, \ldots, N$. This choice of $\boldsymbol{x}$ has the advantage of avoiding the computation of the eigenvectors of a large $N \times N$ matrix and, as we will see in short, it produces a visually appealing segmentation of the image intensities.

We applied the K-means, Expectation Maximization (EM) and Polynomial Segmentation (Polysegment) algorithms to the penguin, dancer and baseball images shown in Figure 2.5. In all cases the number of groups is obtained from the Polysegment algorithm as $n = 3$. This number is given as an input to K-means and EM, because they need to know the number of groups in advance. The K-means algorithm is randomly initialized with 3 intensity values uniformly chosen on the interval $[0, 1]$. The EM algorithm is initialized with K-means, unless otherwise stated.



(a) Penguin (b) Dancer (c) Baseball

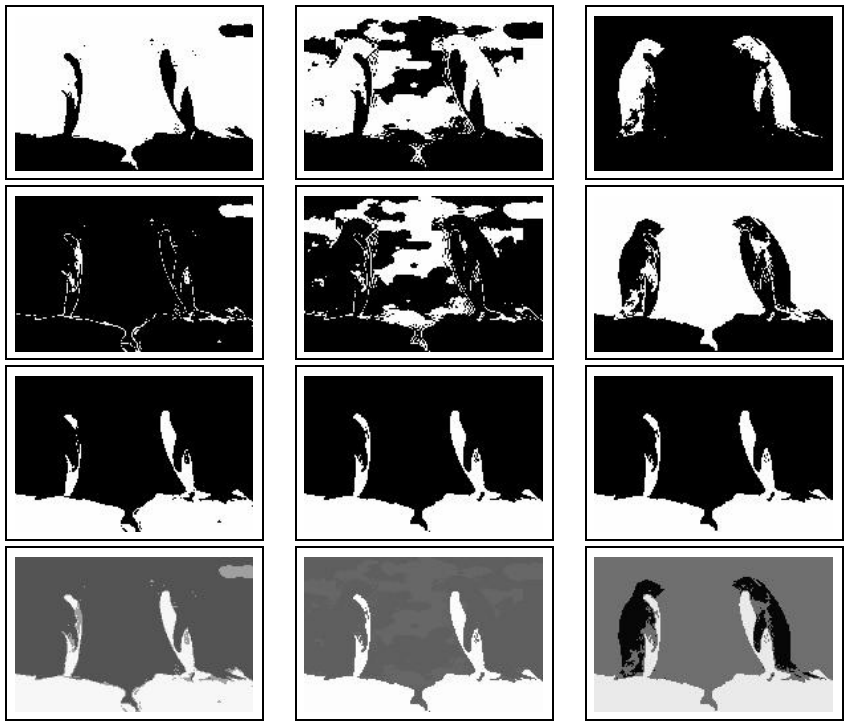
Figure 2.5: Input images for intensity-based image segmentation.

Figures 2.6(a)-(c) plot the segmentation results given by each algorithm for the *penguin* image. Each one of the three groups is plotted in white. We observe that K-means and EM converge to a local minima, while Polysegment gives a good segmentation and is about 5 times faster than K-means and 35 times faster than EM, as shown in Table 2.1. In Figures 2.6(d)-(e) the solution of Polysegment is used to re-initialize K-means and EM. They now give a good segmentation, although the solution of Polysegment is still slightly better. Notice that the execution time of K-means and EM reduces around 40% and 20%, respectively, when initialized with Polysegment.

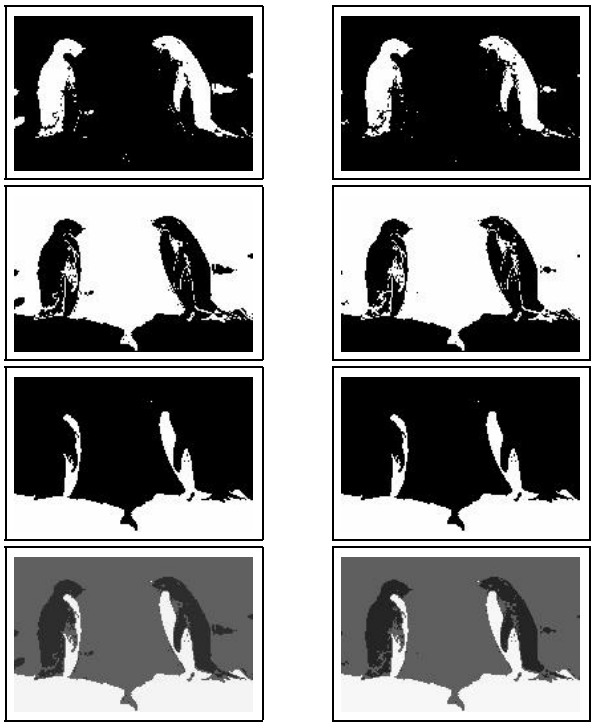
Figure 2.7 plots the segmentation results for the *dancer* image. Notice that all the algorithms give a very similar segmentation of the image. However, Polysegment is approximately 5 times faster than K-means and 20 times faster than EM. When re-initialized with Polysegment, the execution time of K-means reduces by about 40%, while the execution time of EM does not change.

Figure 2.8 plots the segmentation results for the *baseball* image. As before, all algorithms give a similar segmentation, but Polysegment is at approximately 5 times faster.

In summary, these examples show that Polysegment produces a segmentation of 1-D data that is similar to the ones given by K-means and EM, though in about $20\%$ of the execution time. This is because for $N$ pixels and $n$ groups, Polysegment only needs to solve one $N \times (n+1)$ linear system, and then find the roots of a polynomial of degre $n$.



(a) K-means (b) EM (c) Polysegment

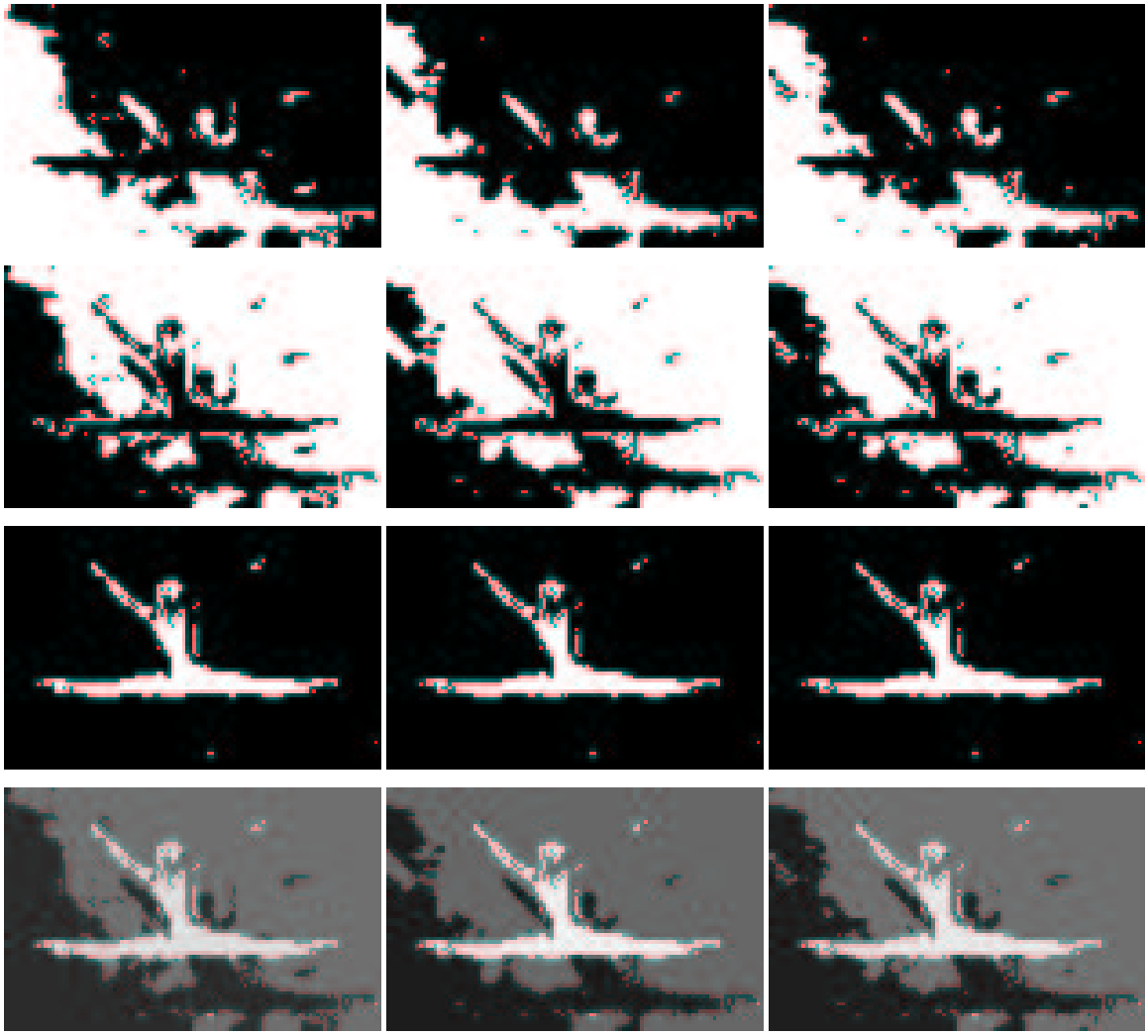


(d) Polysegment + K-means (e) Polysegment + EM

Figure 2.6: Intensity-based segmentation of the penguin image. From top to down: group 1, group 2, group 3 and overall segmentation computed by assigning each pixel to the closest gray level.

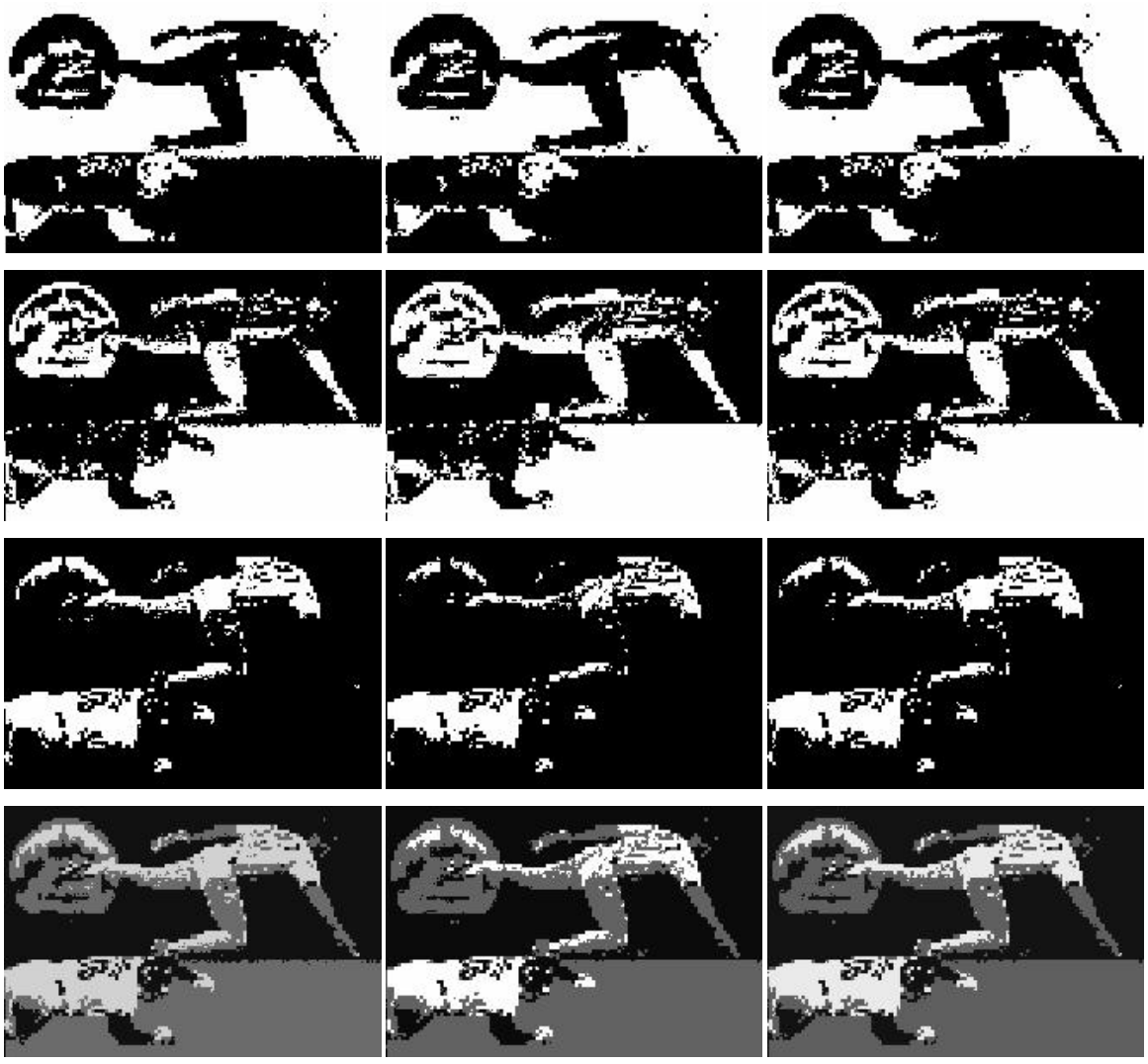
Table 2.1: Execution time (seconds) of each algorithm for a MATLAB implementation, running on a 400 MHz Pentium II PC.

| Image | Size | K-means | EM | Polysegment | Polysegment + K-means | Polysegment K-means + EM |
|---|---|---|---|---|---|---|
| Penguin | $117 \times 180$ | 1.74 | 10.9 | 0.31 | 1.10 | 8.0 |
| Dancer | $67 \times 104$ | 0.53 | 2.37 | 0.12 | 0.37 | 2.37 |
| Baseball | $147 \times 221$ | 2.11 | 9.72 | 0.47 | 1.50 | 9.45 |



(a) K-means (b) EM (c) Polysegment

Figure 2.7: Intensity-based segmentation of the dancer image. From top to down: group 1, group 2, group 3 and overall segmentation computed by assigning each pixel to the closest gray level.



(a) K-means (b) EM (c) Polysegment

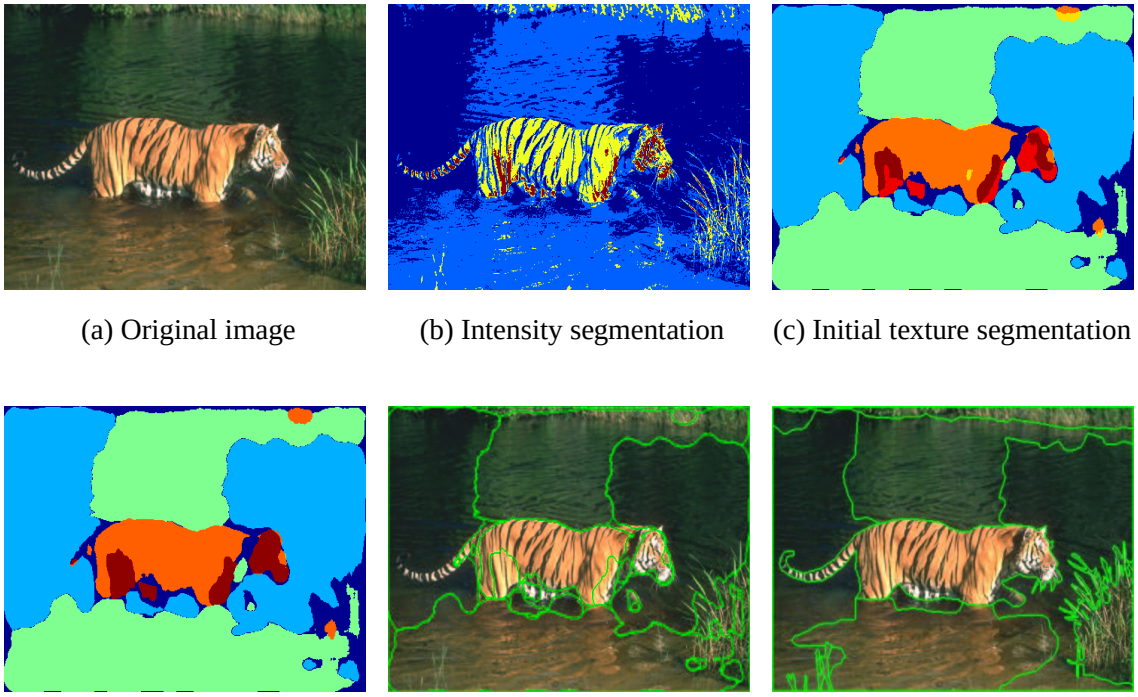
Figure 2.8: Intensity-based segmentation of the baseball image. From top to down: group 1, group 2, group 3 and overall segmentation computed by assigning each pixel to the closest gray level.

### 2.6.2 Image segmentation based on texture

In this section, we apply Polysegment to the problem of segmenting an image based on texture. We propose a simple algorithm that uses Polysegment to compute *textons* based on quantized image intensities (no color information is used), and then segments the image by applying Polysegment in each dimension of the texton space. The algorithm proceeds as follows.

1. **Intensity-based segmentation.** The original image is segmented into $n_t$ groups by applying Polysegment to the vector of image intensities $\boldsymbol{x} \in \mathbb{R}^N$. Figures 2.9(b), 2.10(b) and 2.11(b) show examples of applying Polysegment based on intensity to images with texture.
2. **Texton computation.** As expected, the "quantized" image obtained in the previous step is approximately constant in regions with little texture and has a larger variability in regions with a lot of texture. Therefore, we can use the distribution of the quantized image intensities in a neighborhood of each pixel as a measure of the texturedness of a region. More specifically, we compute a histogram of quantized intensity values in a $w \times w$ neighborhood of each pixel. We interpret such a histogram as a vector in $\mathbb{R}^{n_t}$ and call it *texton*, since it defines a measure of the texturedness around each pixel. We then form a matrix of textons $X \in \mathbb{R}^{N \times n_t}$ whose $j^{th}$ row is equal to the texton associated with pixel $j$. We interpret each column of $X$ as an eigenvector of some similarity matrix.[6]
3. **Texton segmentation.** The textons computed in the previous step are segmented into groups by applying Polysegment to the matrix of textons $X \in \mathbb{R}^{N \times n_t}$, as described in Algorithm 1.
4. **Reducing the number of groups.** As discussed at the end of Section 2.4, Polysegment can produce a large number of groups when applied to multiple eigenvectors. In order to reduce the number of groups, we associate a *new* image to the output of the previous step and attempt to segment it into a smaller number of groups. More specifically, for each one of the group of pixels obtained in the previous step, we compute the average value of their intensity in the original image and generate a new quantized image containing those values. Figures 2.9(c), 2.10(c) and 2.11(c) show some examples of these new quantized images.
5. **Final segmentation.** Apply Polysegment based on intensity to the image obtained in the previous step. The result corresponds to the final texture-based segmentation of the original image. Figures 2.9(d), 2.10(d) and 2.11(d) show some examples of applying Polysegment based on intensity to the images in Figures 2.9(c), 2.10(c) and 2.11(c), respectively.

[6] In principle one could consider each texton as a feature vector associated with each pixel, form a similarity matrix from the distance between pairs of features, and compute a set of eigenvectors of such a similarity matrix. This second approach is however very computationally intensive when $N$ is large.



(a) Original image (b) Intensity segmentation (c) Initial texture segmentation

(d) Final texture segmentation (e) Final segmentation overlay (f) Human segmentation

Figure 2.9: Texture-based segmentation results for the *tiger* image. (a) Original $321 \times 481$ image. (b) The original image is segmented into 4 groups by applying Polysegment to the image intensities. (c) A 4-dimensional texton is associated with each pixel by computing a histogram of quantized intensities in a $23 \times 23$ window around each pixel. Polysegment is applied in each dimension in texton space to separate the textons into 10 groups. The image in (c) is generated with the average intensity of the pixels belonging to each group of textons. (d) Polysegment is applied to the intensity of the image in (c) to obtain the final segmentation into 6 groups. (e) The overall texture-based segmentation is overlayed onto the original image. (f) Human segmentation results from the Berkeley segmentation dataset [38]. The overall execution time is 24 seconds.

The above algorithm for texture computation and segmentation was implemented in MATLAB and runs in approximately 20-30 seconds for $321 \times 481$ images, and 5-10 seconds for $128 \times 192$ images[7]. We tested the algorithm on some images from the Corel database and the Berkeley segmentation dataset [38]. Figure 2.9 shows segmentation results for the image of a *tiger*. The water is separated into two groups due to variations of intensity. The tiger is segmented into four groups. The largest group corresponds to the body of the tiger and the others correspond to smaller parts in the body that have a different texture. The tail and the bushes are not segmented properly because they are averaged out when computing textons in a $31 \times 31$ window. Notice that the segmentation

[7] Computation times are for a MATLAB implementation running on a 400 MHz Pentium II PC.

results are similar to those obtained by human subjects.

Figure 2.10 shows segmentation results for the image of a *marmot* lying on a rock with some other rocks on the background. The algorithm gives a nice segmentation of the image, especially for the rocks in the front and in the upper right corner. The front of the marmot is correctly segmented, but its back is not separated from the big rock in the center, because there is no clear texture boundary. The algorithm could not segment the two large rocks on the top left of the image.

Figure 2.11 shows segmentation results for the image of a *zebra* with grass on the background. Our algorithm gives a nice segmentation of the image into two groups: the zebra and the grass. These results outperform those reported in [45] that apply the normalized cuts algorithm to the eigenvectors of a similarity matrix computed from the output of a bank of filters.

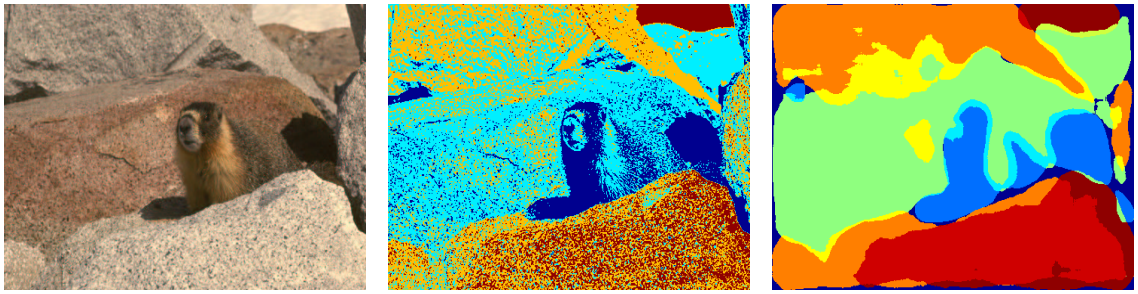
### 2.6.3 Segmentation of 2-D translational motions from feature points or optical flow

A classic problem in visual motion analysis is to estimate a motion model for a set of 2-D feature points as they move in a video sequence. Ideally, one would like to fit a single model that describes the motion of all the feature points in the image. In practice, however, different regions of the image will obey different motion models due to depth discontinuities, perspective effects, multiple moving objects, etc. Therefore, one is faced with the problem of fitting multiple motion models to the image, without knowing which pixels are moving according to the same model.

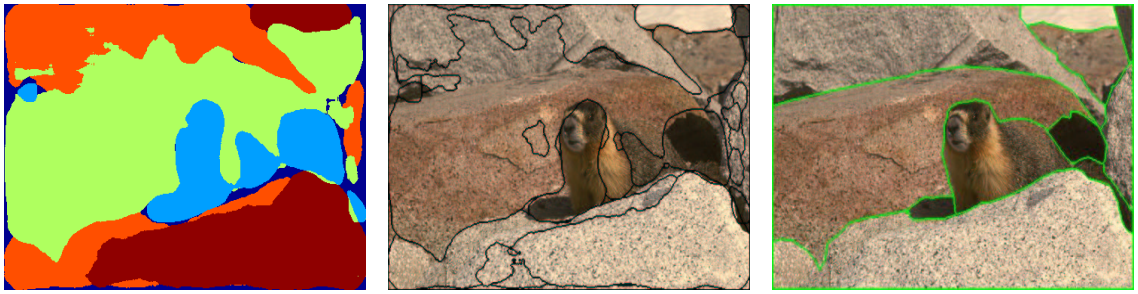
The typical solution to the above problem is problem is to consider a local approach in which one considers a window around each pixel (or the K-nearest neighbors of each feature point) and assumes that within each window there is a single motion model. Choosing a small window ensures that there is a single model in the window, though in the presence of noise the estimation of the model is poor. Choosing a large window does improve the estimates of the motion model. However, it is more likely that the window will contain more than one motion model.

In this section, we consider the 2-D motion segmentation problem in the case of 2-D translational motions and show that it is a direct application of Polysegment with $K = 2$.[8] We demonstrate that one can *globally* fit multiple motion models by applying Polysegment to all the features, without having to choose a window (neighborhood) around each pixel (feature point). Alternatively, one can also apply Polysegment within a window. However, since Polysegment is not restricted to estimating a single motion model, one can choose a large window to obtain a robust estimate of the models, without having to worry about crossing motion boundaries.

[8] We will discuss more complex motion models later in the thesis. For example, see Section 3.9.3 for the segmentation of affine motion models also from feature points.

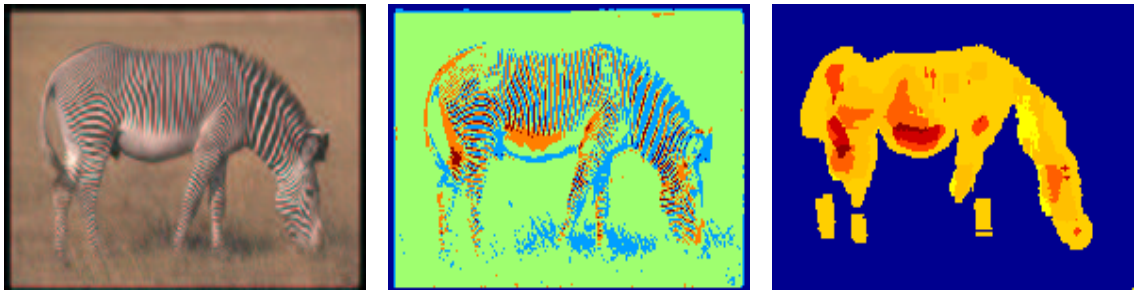


(a) Original image (b) Intensity segmentation (c) Initial texture segmentation

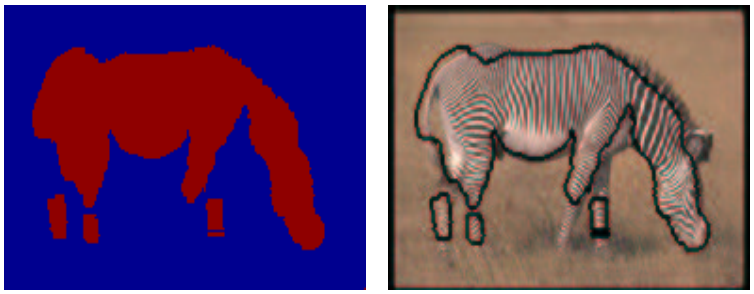


(d) Final texture segmentation (e) Final segmentation overlay (f) Human segmentation

Figure 2.10: Texture-based segmentation results for the $321 \times 481$ *marmot* image. Five groups are obtained by segmenting 4-D textons computed in a $31 \times 31$ window. The execution time is 25 sec.



(a) Original image (b) Intensity segmentation (c) Initial texture segmentation



(d) Final texture segmentation (e) Final segmentation overlay

Figure 2.11: Texture-based segmentation results for the $128 \times 192$ *zebra* image. Two groups are obtained from 5-D textons computed in a $11 \times 11$ window. The execution time is 25 sec.

**2-D Motion segmentation from feature points**

We let $\{\boldsymbol{x}_1^j \in \mathbb{R}^2\}_{j=1}^N$ and $\{\boldsymbol{x}_2^j \in \mathbb{R}^2\}_{j=1}^N$ be a collection of $N$ feature points in two frames from a sequence. Under the 2-D translational motion model each feature moves according to one out of $n$ possible 2-D displacement vectors $\{\boldsymbol{d}_i \in \mathbb{R}^2\}_{i=1}^n$. That is, for each feature pair $(\boldsymbol{x}_1^j, \boldsymbol{x}_2^j)$ there exist a 2-D displacement $\boldsymbol{d}_i$ such that

$$\boldsymbol{x}_2^j = \boldsymbol{x}_1^j + \boldsymbol{d}_i. \tag{2.23}$$

The problem is now to estimate the $n$ motion models $\{\boldsymbol{d}_i \in \mathbb{R}^2\}_{i=1}^n$ from the collection of $N$ feature pairs $\{(\boldsymbol{x}_1^j, \boldsymbol{x}_2^j)\}_{j=1}^N$. To this end, we define a set of complex data points $\{\boldsymbol{z}_j \in \mathbb{C}\}_{j=1}^N$ from the displacement of the features in the image plane between the two views $\{\boldsymbol{x}_2^j - \boldsymbol{x}_1^j \in \mathbb{R}^2\}_{j=1}^N$. Then, the motion models $\{\boldsymbol{d}_i \in \mathbb{R}^2\}_{i=1}^n$ can be immediately obtained by applying Polysegment to the complex data $\{\boldsymbol{z}_j \in \mathbb{C}\}_{j=1}^N$.

**2-D Motion segmentation from optical flow**

Imagine now that rather than a collection of feature points we are given the optical flow $\{\boldsymbol{u}_j \in \mathbb{R}^2\}_{j=1}^N$ between two consecutive views of a video sequence. If we assume that the optical flow is piecewise constant, i.e., the optical flow of every pixel in the image takes only $n$ possible values $\{\boldsymbol{d}_i \in \mathbb{R}^2\}_{i=1}^n$, then at each pixel $j$ we have that there exists a motion $\boldsymbol{d}_i$ such that

$$\boldsymbol{u}_j = \boldsymbol{d}_i. \tag{2.24}$$

The problem is now to estimate the $n$ motion models $\{\boldsymbol{d}_i \in \mathbb{R}^2\}_{i=1}^n$ from the optical flow $\{\boldsymbol{u}_j\}_{j=1}^N$. Notice that this problem is equivalent to the problem of segmenting the image into $n$ regions with constant flow within the region. We solve this problem by applying Polysegment with $K = 2$ to the optical flow data $\{\boldsymbol{u}_j\}_{j=1}^N$ interpreted as a collection of points in the complex plane $\mathbb{C}$.

We tested the proposed approach by segmenting 12 frames of a real sequence consisting of an aerial view of two robots moving on the ground. At each frame, we apply the Polysegment algorithm with $K = 2$ to the optical flow[9] of all $N = 240 \times 352$ pixels in the image and segment the image measurements according to the $n = 3$ estimated translational motion models corresponding to the two robots and the background. Figure 2.12 shows the results of applying our algorithm to segmenting the pixels in frames 1, 4, 7, and 10 of the sequence. Notice that the three different motion models are correctly estimated, and the two moving robots are correctly segmented.

[9] We compute optical flow using Black's code at `http://www.cs.brown.edu/people/black/ignc.html`.

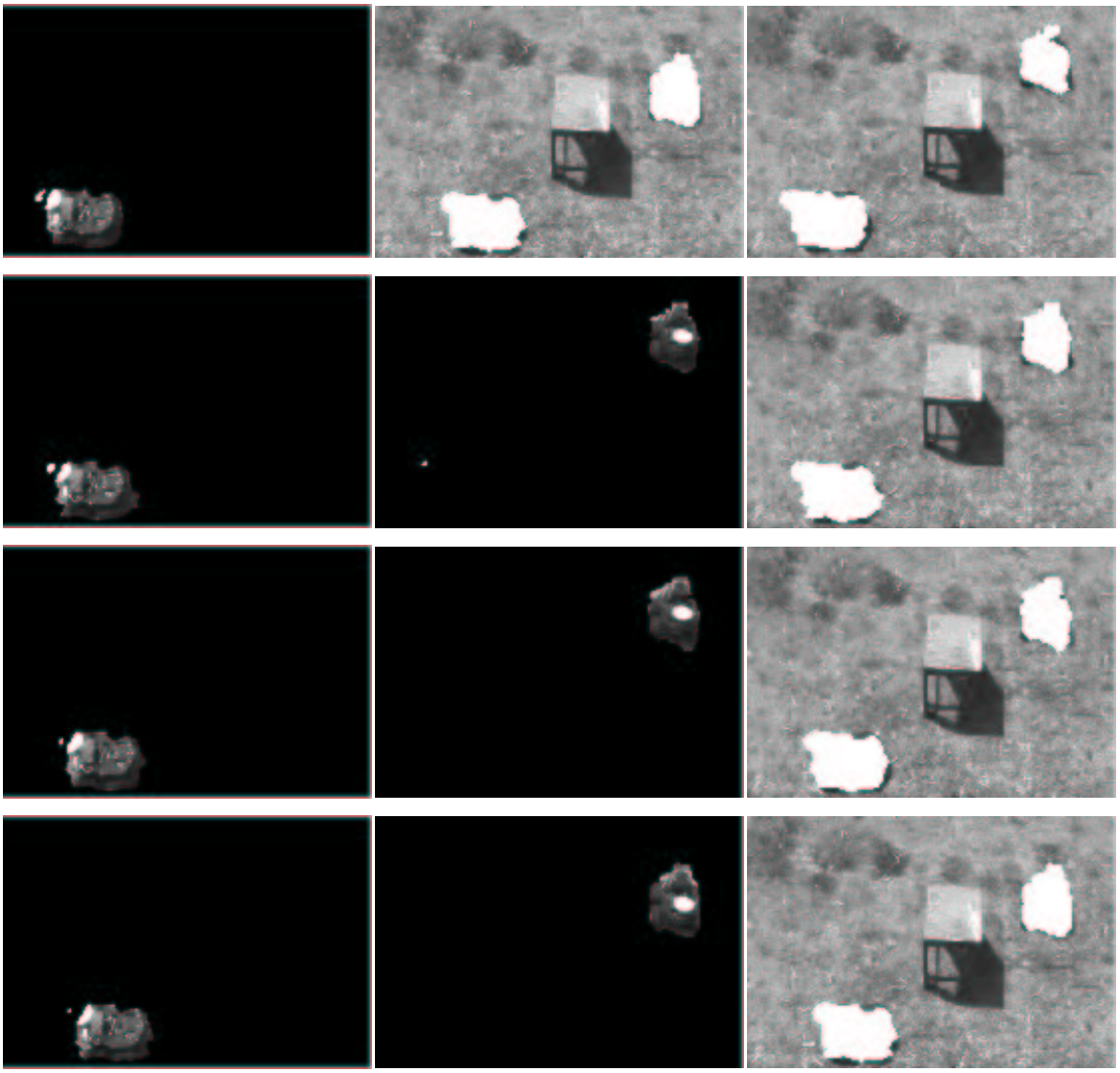


Figure 2.12: Segmenting the optical flow of a video sequence using Polysegment with $K = 2$. At each frame, we use the optical flow of all $N = 240 \times 352$ pixels to build the data matrix $L_n \in \mathbb{C}^{N \times (n+1)}$ corresponding to $n = 3$ motions: the two robots and the background. We then obtain a vector $\boldsymbol{c} \in \mathbb{C}^{n+1}$ such that $L_n \boldsymbol{c} = 0$, and compute $\{\boldsymbol{d}_i \in \mathbb{C}^2\}_{i=1}^n$ as the roots of the polynomial $\sum_{k=0}^n c_k z^k$. We then assign each pixel $j$ to motion model $\boldsymbol{d}_i \in \mathbb{R}^2$ if $i = \arg\min_\ell \|\boldsymbol{u}_j - \boldsymbol{d}_\ell\|$.

### 2.6.4 Segmentation of 3-D infinitesimal motions from optical flow in multiple views

In this section, we apply Polysegment to the problem of segmenting the 3-D infinitesimal motion of $n$ independently and rigidly moving objects observed by a moving perspective camera in multiple frames. We let $(\mathrm{u}_j^i, \mathrm{v}_j^i)$ be the optical flow of pixel $i$ in frame $j$ relative to frame $0$, with $i = 1, \ldots, N$ and $j = 1, \ldots, f$. Let $U$ and $V$ be the multi-frame optical flow matrices

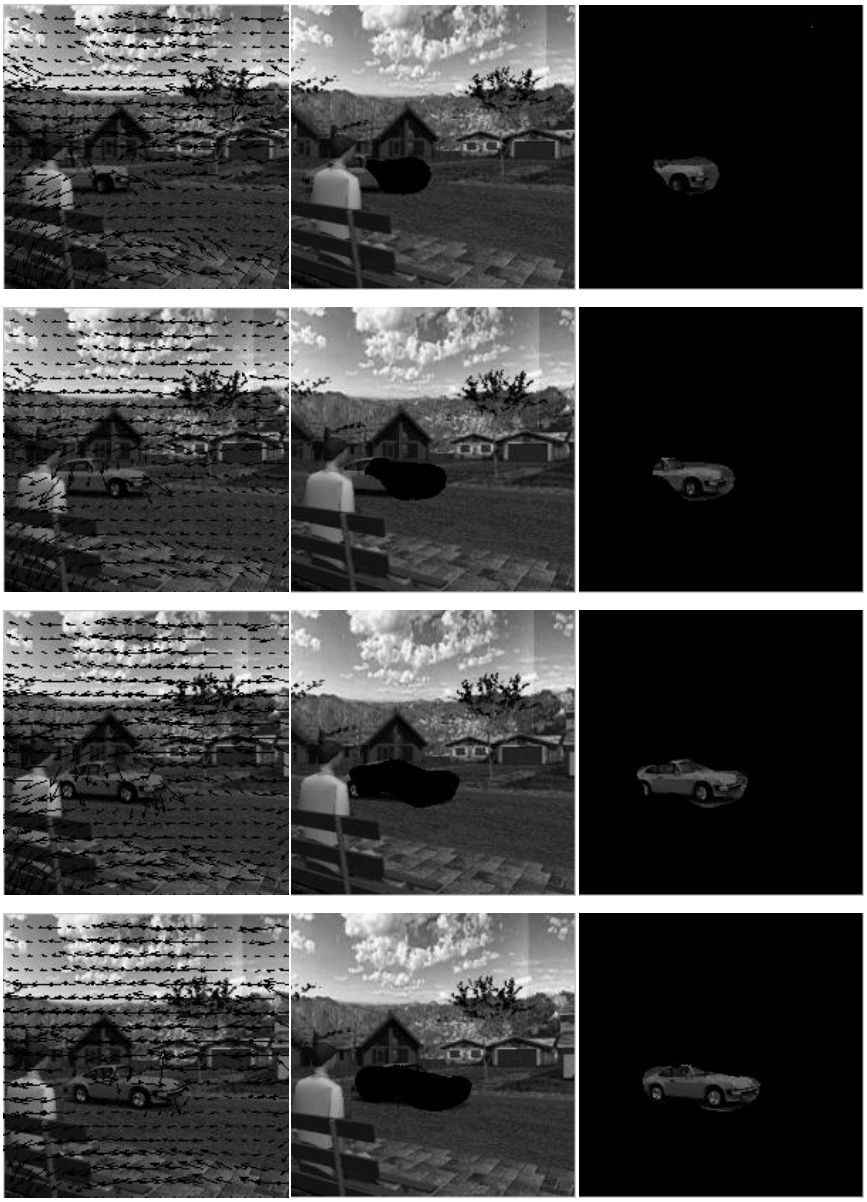
$$U = \begin{bmatrix} \mathrm{u}_1^1 & \cdots & \mathrm{u}_f^1 \\ \vdots & & \vdots \\ \mathrm{u}_1^N & \cdots & \mathrm{u}_f^N \end{bmatrix} \quad \text{and} \quad V = \begin{bmatrix} \mathrm{v}_1^1 & \cdots & \mathrm{v}_f^1 \\ \vdots & & \vdots \\ \mathrm{v}_1^N & \cdots & \mathrm{v}_f^N \end{bmatrix}.$$

We showed in [63] (see also [37]) that the matrix $\mathcal{S} = [U, V][U, V]^T \in \mathbb{R}^{N \times N}$ defines a similarity matrix for the 3-D motion of the objects. Since the eigenvectors of $\mathcal{S}$ are the singular vectors of $W = [U, V] \in \mathbb{R}^{N \times 2f}$, we will apply our segmentation algorithm to the singular vectors of $W$, since it is computationally more efficient when $2f << N$.

Figure 2.13 shows the *street* sequence[10], which contains two independent motions: the car translating to the right and the camera panning to the right. Figure 2.13(a) shows frames 3, 8, 12 and 16 of the sequence with the corresponding optical flow superimposed. The optical flow is computed using Black's algorithm[11]. Figures 2.13(b)-(c) show the segmentation results. In frame 3 the car is partially occluded, thus only the frontal part of the car is segmented from the background. The door is incorrectly segmented because it is in a region with low texture. As time proceeds, motion information is integrated over time by incorporating optical flow from many frames in the optical flow matrix, thus the door is correctly segmented. In frame 16 the car is fully visible and correctly segmented from the moving background.

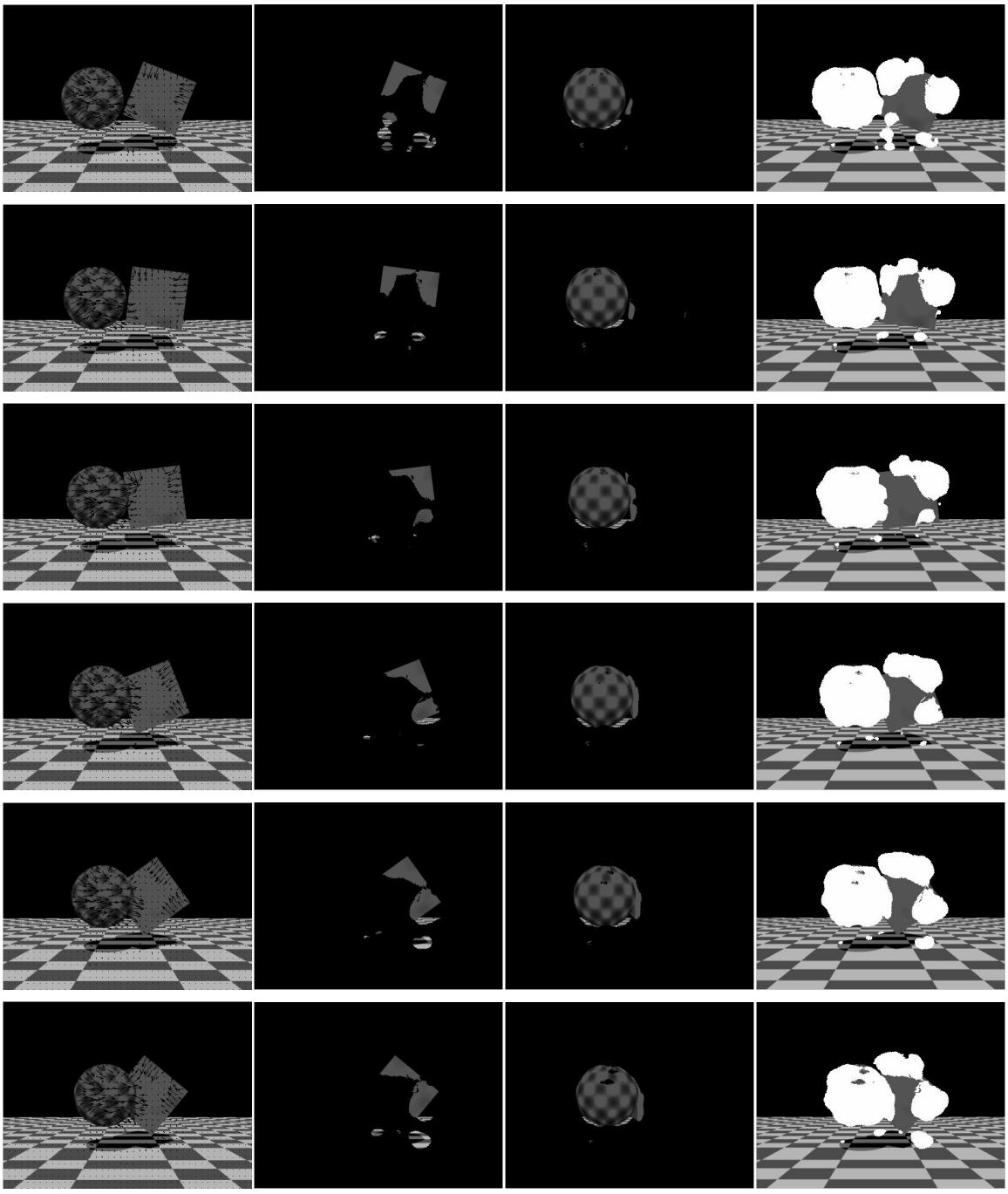
Figure 2.14 shows the *sphere-cube* sequence, which contains a sphere rotating along a vertical axis and translating to the right, a cube rotating counter clock-wise and translating to the left, and a static background. Even though the optical flow of the sphere appears to be noisy, its motion is correctly segmented. The top left (when visible), top and right sides of the square are also correctly segmented in spite of the fact that only normal flow is available. The left bottom side of the cube is merged with the background, because its optical flow is small, since the translational motion of the cube cancels its rotational motion. The center of the cube is never segmented correctly since it corresponds to a region with low texture. Integrating motion information over many frames does not help here since those pixels are in a region with low texture during the whole sequence.

[10] http://www.cs.otago.ac.nz/research/vision/Research/OpticalFlow/opticalflow.html#Sequences
[11] http://www.cs.brown.edu/people/black/ignc.html



(a) Optical flow (b) Group 1 (c) Group 2

Figure 2.13: Motion-based segmentation results for the *street* sequence. The sequence has 18 frames and $200 \times 200$ pixels. The camera is panning to the right while the car is also moving to the right. (a) Frames 3, 8, 12 and 16 of the sequence with their optical flow superimposed. (b) Group 1: motion of the camera. (c) Group 2: motion of the car.



(a) Optical flow (b) Group 1 (c) Group 2 (d) Group 3

Figure 2.14: Motion-based segmentation results for the *sphere-cube* sequence. The sequence contains 10 frames and $400 \times 300$ pixels. The sphere is rotating along a vertical axis and translating to the right. The cube is rotating counter clock-wise and translating to the left. The background is static. (a) Frames 2-7 with their optical flow superimposed. (b) Group 1: cube motion. (c) Group 2: sphere motion. (d) Group 3: static background.

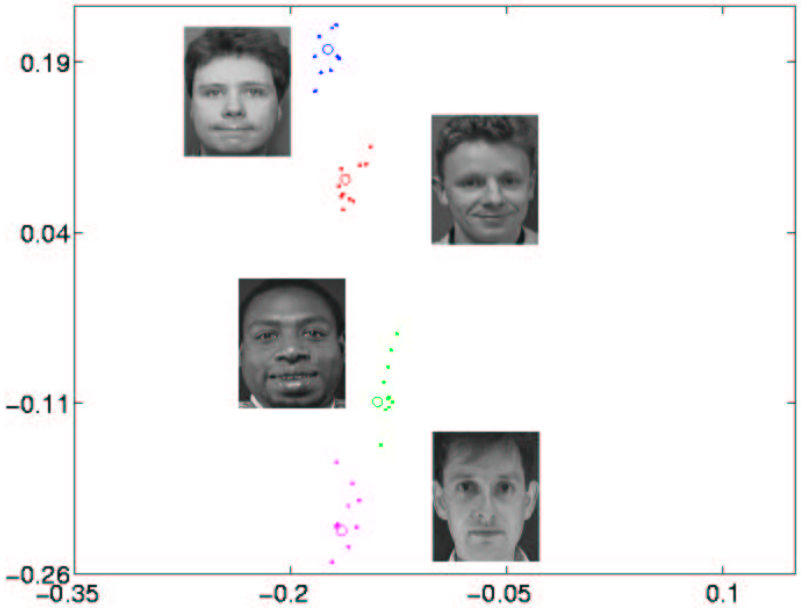
### 2.6.5 Face clustering with varying expressions

A fundamental problem in face recognition is to cluster a collection of images of faces taken under varying illumination, pose, expression, etc. This is a very challenging problem, because images of the same face may differ significantly under varying conditions. Conversely, it is easy to build examples in which the images of two different faces appear similar from image data only.

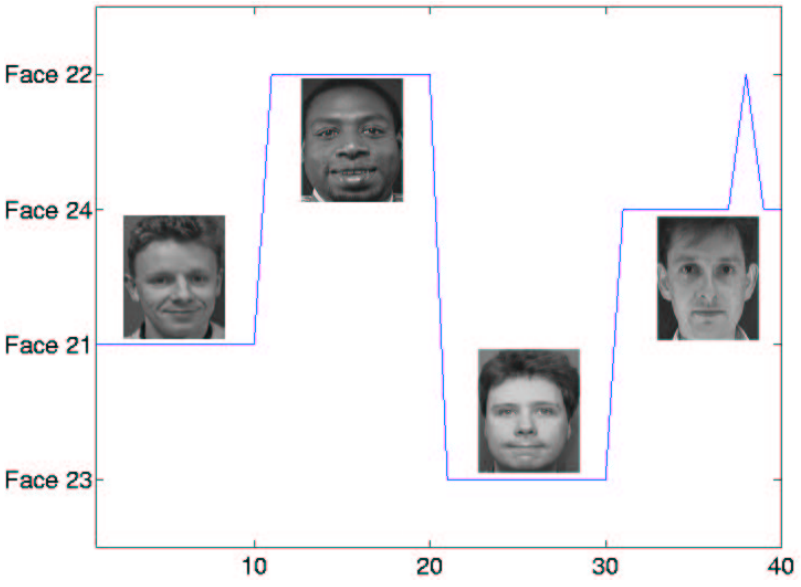
In this section, we apply Polysegment to the problem of clustering faces with varying expressions. We assume that the images $\{I_j \in \mathbb{R}^K\}_{j=1}^N$ cluster around $n$ cluster centers in the image space $\mathbb{R}^K$, with each cluster center corresponding to a different individual. Since in practice the number of pixels $K$ is large, we first apply PCA to project the images onto $\mathbb{R}^{K'}$ with $K' << K$. More specifically, we compute the SVD of the data $[I_1,\ I_2,\ \ldots,\ I_N]_{K\times N} = U\Sigma V^T$ and generate a new data matrix $X \in \mathbb{R}^{N\times K'}$ consisting of the first $K'$ columns of $V$. As before, we interpret the rows of $X$ as a new set of data points in $\mathbb{R}^{K'}$, so that we can avoid building a similarity matrix.

We apply Polysegment to the subset of the ORL Database of Faces (AT&T Cambridge) shown in Figure 2.15 which consists of $N = 10n$ images of $n = 4$ faces (subjects 21-24). Each individual has $10$ different facial expressions, and in some cases there is also a small change in pose. All the images are taken with the same illumination. For computational efficiency, we first project each image from $\mathbb{R}^K$, where $K = 92 \times 112 = 10,304$ pixels, to the first $K' = 2$ principal components using PCA. Figure 2.16(a) shows the $40$ images as data points in $\mathbb{R}^2$. Notice that the faces of different individuals indeed cluster around $4$ cluster centers. We then apply Polysegment to the 2-dimensional data and obtain the corresponding $4$ cluster centers shown in Figure 2.16(a) with a "$\circ$". Figure 2.16(b) shows the clustering of the faces. Notice that there is only one mismatch.

Figure 2.15: A subset of the ORL Database of Faces (AT&T Cambridge) consisting of $N = 40$ images of $n = 4$ faces (subjects 21-24) with varying expressions.




(a) Images of the $N = 40$ faces projected onto the two principal components, and the cluster centers ("$\circ$") estimated by Polysegment.




(b) Segmentation results obtained by assigning each image to the closest cluster center. There is only one mismatch.

Figure 2.16: Clustering faces with varying expressions using Polysegment with $K = 2$.

## 2.7 Conclusions, discussions and future work

We have proposed a simple analytic solution to the problem of segmenting piecewise constant data from the eigenvectors of a similarity matrix.

In the absence of noise, we derived a rank constraint on the entries of each eigenvector, from which one can determine the number of groups $n$ contained in the data. Given $n$, the segmentation of a single eigenvector is equivalent to solving a linear system in $n$ variables plus computing the roots of a polynomial of degree $n$ in one variable. In the presence of noise, we showed that the purely algebraic solution is robust since it minimizes the algebraic error obtained in the noise free case. Furthermore, we derived the optimal error function for the case of zero-mean Gaussian noise in the entries of the eigenvector. We also generalized our polynomial segmentation technique to the case of multiple eigenvectors by reducing it to the single eigenvector case. We then showed how our technique can be naturally used to initialize iterative algorithms such as K-means and Expectation Maximization (EM).

We applied our algebraic algorithm (Polysegment) to the problem of segmenting an image based on different cues such as intensity, texture or motion. Our experiments on intensity-based segmentation showed that Polysegment performs similarly to K-means and EM, but is computationally lest costly. Our experiments on texture-based image segmentation showed that Polysegment is very efficient at computing textures and gives a visually appealing segmentation of natural scenes. Our experiments on motion segmentation showed that Polysegment gives a good segmentation of both static and dynamic scenes. We also applied Polysegment to the problem of clustering faces with varying expressions. Our experiments showed the possibility of applying Polysegment to high-dimensional data after a suitable projection. It is important to notice that none of our experiments required the use of any of the nonlinear optimization algorithms. In all cases, a simple linear algebraic technique was enough to segment real noisy image data. We therefore believe that the results presented in this chapter are quite encouraging and we look forward to applying Polysegment to a wider variety of segmentation problems involving piecewise constant data. We are particularly interested in image segmentation from multiple cues.

Future work will hence concentrate on improving the simultaneous segmentation of multiple eigenvectors. Our current algorithm obtains the overall segmentation by combining individual segmentations given by each eigenvector. This usually produces a segmentation of the scene containing too many groups. We showed how to reduce the number of groups in the case of texture-based image segmentation and expect to generalize that technique to arbitrary data in the near future.

# Chapter 3

# Generalized Principal Component Analysis (GPCA)

## 3.1 Introduction

Principal Component Analysis (PCA) [29] refers to the problem of identifying a linear subspace $S \subset \mathbb{R}^K$ of unknown dimension $k < K$ from $N$ sample points $\boldsymbol{x}^j \in S$, $j = 1, 2, \ldots, N$. 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 $[\boldsymbol{x}^1, \boldsymbol{x}^2, \ldots, \boldsymbol{x}^N] \in \mathbb{R}^{K \times N}$. 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 $\boldsymbol{x}^j$ to their projections $\tilde{\boldsymbol{x}}^j$ in $S$.

In addition to this algebraic-geometric interpretation, PCA can also be understood in a probabilistic manner. In Probabilistic PCA [53] (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 [9]. However, when the underlying distribution is non Gaussian the solution to PPCA is no longer linear. For example, in [9] 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 [41] is based on first embedding the data into a higher-dimensional *feature* space $\boldsymbol{F}$ 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 $\boldsymbol{F}$ 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 chapter, we consider the following (alternative) extension of PCA to the case of mixtures of subspaces, which we call Generalized Principal Component Analysis (GPCA):

---

**Problem 2 (Generalized Principal Component Analysis (GPCA))**

Given a set of sample points $\boldsymbol{X} = \{\boldsymbol{x}^j \in \mathbb{R}^K\}_{j=1}^N$ 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 3.1 illustrates the case of $n = 3$ subspaces of $\mathbb{R}^3$ of dimensions $k_1 = k_2 = k_3 = 2$.

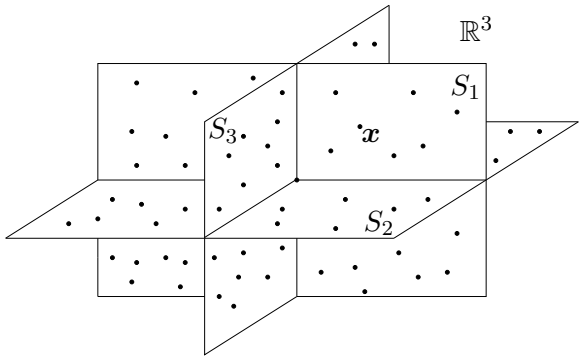



Figure 3.1: Three ($n = 3$) 2-dimensional subspaces $S_1, S_2, S_3$ in $\mathbb{R}^3$. The objective of GPCA is to identify all three subspaces from samples $\{\boldsymbol{x}\}$ drawn from these subspaces.

### 3.1.1 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 [31] (see also [7, 11]) 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 [11]. 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 [46] 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 [52] 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.

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

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?*

### 3.1.2 Our approach to mixtures of principal components: GPCA

In this chapter, 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.[2] 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. *Algebraic sets and varieties*: We show in Section 3.2 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] Part of the results presented in this chapter were published in [58].

2. *Mixtures of $(K-1)$-dimensional subspaces*: We show in Section 3.3 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 3.4 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 3.5 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 3.6 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 3.7 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 3.8 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 3.9 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 7 (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 multi-linear counterpart, the so-called higher order singular value decomposition (HOSVD) [13], such a generalization is not unique. Furthermore, while the SVD of a matrix $A = U\Sigma V^T$ produces a diagonal matrix $\Sigma$, 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.*

## 3.2 Representing mixtures of subspaces as algebraic sets and varieties

We represent each subspace $S_i$ by choosing a basis

$$B_i \doteq [\boldsymbol{b}_{i1}, \ldots, \boldsymbol{b}_{i(K-k_i)}] \in \mathbb{R}^{K \times (K-k_i)} \tag{3.1}$$

for its orthogonal complement[3] $S_i^{\perp}$. With this representation, each subspace is described as

$$S_i = \{\boldsymbol{x} \in \mathbb{R}^K : B_i^T \boldsymbol{x} = 0\} = \{\boldsymbol{x} \in \mathbb{R}^K : \bigwedge_{j=1}^{K-k_i} (\boldsymbol{b}_{ij}^T \boldsymbol{x} = 0)\}. \tag{3.2}$$

Therefore, an arbitrary point $\boldsymbol{x}$ lies on one of the subspaces if and only if

$$(\boldsymbol{x} \in S_1) \vee \cdots \vee (\boldsymbol{x} \in S_n) \equiv \bigvee_{i=1}^{n} (\boldsymbol{x} \in S_i) \equiv \bigvee_{i=1}^{n} \bigwedge_{j=1}^{K-k_i} (\boldsymbol{b}_{ij}^T \boldsymbol{x} = 0) \equiv \bigwedge_{\sigma} \bigvee_{i=1}^{n} (\boldsymbol{b}_{i\sigma(i)}^T \boldsymbol{x} = 0), \tag{3.3}$$

where the right hand side (RHS) of (3.3) is obtained by exchanging ands and ors using De Morgan's laws, and $\sigma$ represents a particular choice of one normal vector $\boldsymbol{b}_{i\sigma(i)}$ 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

$$\bigvee_{i=1}^{n} (\boldsymbol{b}_{i\,\sigma(i)}^T \boldsymbol{x} = 0) \equiv \left( p_{n\sigma}(\boldsymbol{x}) = \prod_{i=1}^{n} (\boldsymbol{b}_{i\,\sigma(i)}^T \boldsymbol{x}) = 0 \right), \tag{3.4}$$

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*[4] as a product of $n$ linear expressions in $\boldsymbol{x}$, 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 (3.4).

**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:*

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

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

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

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

$$p_{21}(\boldsymbol{x}) = x_1 x_3 \qquad \textit{and} \qquad p_{22}(\boldsymbol{x}) = x_2 x_3.$$

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

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

**Remark 8** *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 $Z$ [23]. 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 $n$, $\{p_{n\sigma}(\boldsymbol{x})\}$. Furthermore, if we let $I'$ be the sub-ideal of $I$ generated by the polynomials $\{p_{n\sigma}(\boldsymbol{x})\}$, then $I$ is exactly the radical ideal*[5] *of the ideal $I'$, i.e., $I = rad[I']$.*

The problem of identifying each subspace $S_i$ is then equivalent to one of solving for the normal bases $\{B_i\}_{1=1}^n$ from the set of *nonlinear* equations in (3.4). 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

$$M_n(K) = \begin{pmatrix} n+K-1 \\ K-1 \end{pmatrix} = \begin{pmatrix} n+K-1 \\ n \end{pmatrix} \tag{3.5}$$

different monomials, thus the dimension of $R_n(K)$ as a vector space is $M_n(K)$.[6] 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:*

$$\nu_n : \ [x_1, \ldots, x_K]^T \mapsto [\ldots, \boldsymbol{x}^{\boldsymbol{n}}, \ldots]^T, \tag{3.6}$$

*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 9 (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:*

$$\nu_n(x_1, x_2) = [x_1^n, x_1^{n-1} x_2, x_1^{n-2} x_2^2, \ldots, x_2^n]^T. \tag{3.7}$$

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

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

Thanks to the Veronese map, each polynomial in (3.4) becomes the following linear expression in the vector coefficients $\boldsymbol{c}_n \in \mathbb{R}^{M_n}$

$$p_n(\boldsymbol{x}) = \nu_n(\boldsymbol{x})^T \boldsymbol{c}_n = \sum c_{n_1,\ldots,n_K} x_1^{n_1} \cdots x_K^{n_K} = 0, \tag{3.8}$$

where $c_{n_1,\ldots,n_K} \in \mathbb{R}$ represents the coefficient of monomial $\boldsymbol{x}^{\boldsymbol{n}}$. Therefore, if we apply (3.8) 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 $\boldsymbol{c}_n \in \mathbb{R}^{M_n}$

$$L_n(K)\ \boldsymbol{c}_n \doteq \begin{bmatrix} \nu_n(\boldsymbol{x}^1)^T \\ \nu_n(\boldsymbol{x}^2)^T \\ \vdots \\ \nu_n(\boldsymbol{x}^N)^T \end{bmatrix} \boldsymbol{c}_n = 0 \in \mathbb{R}^N, \tag{3.9}$$

where $L_n(K) \in \mathbb{R}^{N \times M_n}$ is the matrix of *embedded* data points.[7]

**Remark 10 (Kernel Matrix)** *In the context of Kernel PCA, if the polynomial embedding is used, then $\mathcal{C} = L_n^T L_n \in \mathbb{R}^{M_n \times M_n}$ 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 11 (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 (3.9) shows analytically that the commonly used polynomial embedding $\nu_n$ 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 (3.9) 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*[8] in $Z = \cup_{i=1}^n S_i$, then

$$M_n - \prod_{i=1}^n (K - k_i) \leq \text{rank}(L_n) = M_n - m \leq M_n - 1. \tag{3.10}$$

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

[8] 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 $x_1^2 + x_1 x_2$ and $x_2^2 - x_1 x_2$ 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.[9]

In the following sections, we propose an alternative solution to the above problem. In Section 3.3, 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 3.4, 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 3.5, 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.

## 3.3 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 3.3.1, we show that the collection of hyperplanes can be represented with a unique (factorable) polynomial $p_n(\boldsymbol{x})$ whose degree $n$, the number of hyperplanes, can be recovered from a rank constraint on the embedded data matrix $L_n$ and whose coefficients $\boldsymbol{c}_n$ can be recovered by solving a linear system. In Section 3.3.2, 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 3.3.3, 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 $p_n(\boldsymbol{x})$ evaluated at $n$ points each one lying on each one of the hyperplanes.

[9] To the best of our knowledge, although it has been shown that a polynomial-time algorithm exists, the algorithm is not yet known [47].

### 3.3.1 Estimating the number of hyperplanes $n$ and the vector of coefficients $\boldsymbol{c}_n$

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 $\boldsymbol{b}_i \in \mathbb{R}^K$ as follows:[10]

$$S_i = \{\boldsymbol{x} \in \mathbb{R}^K : \boldsymbol{b}_i^T \boldsymbol{x} = b_{i1}x_1 + b_{i2}x_2 + \ldots + b_{iK}x_K = 0\}. \tag{3.11}$$

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

$$(\boldsymbol{b}_1^T \boldsymbol{x} = 0) \ \vee \ (\boldsymbol{b}_2^T \boldsymbol{x} = 0) \ \vee \cdots \vee \ (\boldsymbol{b}_n^T \boldsymbol{x} = 0), \tag{3.12}$$

which is equivalent to the following homogeneous polynomial of degree $n$ in $\boldsymbol{x}$ with real coefficients:

$$\boxed{p_n(\boldsymbol{x}) = \prod_{i=1}^{n} (\boldsymbol{b}_i^T \boldsymbol{x}) = 0.} \tag{3.13}$$

The problem of identifying each subspace $S_i$ is then equivalent to one of solving for the vectors $\{\boldsymbol{b}_i\}_{i=1}^n$ from the *nonlinear* equation (3.13). 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 3.2, we can use the Veronese map of degree, $\nu_n$, to convert equation (3.13) into the following linear expression in the vector of coefficients $\boldsymbol{c}_n \in \mathbb{R}^{M_n}$:

$$p_n(\boldsymbol{x}) = \nu_n(\boldsymbol{x})^T \boldsymbol{c}_n = \sum c_{n_1,n_2,\ldots,n_K} x_1^{n_1} x_2^{n_2} \cdots x_K^{n_K} = 0, \tag{3.14}$$

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

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

$$\begin{aligned}
p_2(\boldsymbol{x}) &= (b_{11}x_1 + b_{12}x_2 + b_{13}x_3)(b_{21}x_1 + b_{22}x_2 + b_{23}x_3) \\
\nu_2(\boldsymbol{x}) &= [x_1^2, x_1x_2, x_1x_3, x_2^2, x_2x_3, x_3^2]^T \\
\boldsymbol{c}_2 &= [\underbrace{b_{11}b_{21}}_{c_{2,0,0}}, \underbrace{b_{11}b_{22}+b_{12}b_{21}}_{c_{1,1,0}}, \underbrace{b_{11}b_{23}+b_{13}b_{21}}_{c_{1,0,1}}, \underbrace{b_{12}b_{22}}_{c_{0,2,0}}, \underbrace{b_{12}b_{23}+b_{13}b_{22}}_{c_{0,1,1}}, \underbrace{b_{13}b_{23}}_{c_{0,0,2}}]^T.
\end{aligned}$$

**Remark 12** *Notice that each $c_{n_1,\ldots,n_K}$ is a symmetric multilinear function of $(\boldsymbol{b}_1, \boldsymbol{b}_2, \ldots, \boldsymbol{b}_n)$, that is $c_{n_1,\ldots,n_K}$ is linear in each $\boldsymbol{b}_i$ and:*

$$c_{n_1,\ldots,n_K}(\boldsymbol{b}_1, \boldsymbol{b}_2, \ldots, \boldsymbol{b}_n) = c_{n_1,\ldots,n_K}(\boldsymbol{b}_{\sigma(1)}, \boldsymbol{b}_{\sigma(2)}, \ldots, \boldsymbol{b}_{\sigma(n)}) \text{ for all } \sigma \in \mathfrak{S}_n, \tag{3.15}$$

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

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

**Remark 13 (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 $\boldsymbol{c}_n$ of the polynomial $p_n(\boldsymbol{x})$ can be interpreted as the symmetric tensor product of the coefficients $\boldsymbol{b}_i$'s of each polynomial of degree $1$, that is:*

$$\boldsymbol{c}_n \simeq \mathit{Sym}(\boldsymbol{b}_1 \otimes \boldsymbol{b}_2 \otimes \ldots \otimes \boldsymbol{b}_n) = \sum_{\sigma \in \mathfrak{S}_n} \boldsymbol{b}_{\sigma(1)} \otimes \boldsymbol{b}_{\sigma(2)} \otimes \ldots \otimes \boldsymbol{b}_{\sigma(n)}$$

*where $\otimes$ represents the tensor or Kronecker product and $\simeq$ represents the homeomorphism between the symmetric tensor Sym$(\boldsymbol{b}_1 \otimes \boldsymbol{b}_2 \otimes \ldots \otimes \boldsymbol{b}_n)$ in $Sym^n(\mathbb{R}^K)$ and its symmetric part written as a vector $\boldsymbol{c}_n$ in $\mathbb{R}^{M_n}$.*

As demonstrated in Section 3.2 (see equation (3.9)), after applying (3.14) to the given collection of $N$ sample points $\{\boldsymbol{x}^j\}_{j=1}^N$, we obtain the following system of linear equations on the vector of coefficients $\boldsymbol{c}_n$

$$L_n \, \boldsymbol{c}_n \doteq \begin{bmatrix} \nu_n(\boldsymbol{x}^1)^T \\ \nu_n(\boldsymbol{x}^2)^T \\ \vdots \\ \nu_n(\boldsymbol{x}^N)^T \end{bmatrix} \boldsymbol{c}_n = 0 \in \mathbb{R}^N. \tag{3.16}$$

**Remark 14 (Brill's equations on the entries of $\boldsymbol{c}_n$)** *Given $n$, one can solve for $\boldsymbol{c}_n$ from (3.16) in a linear fashion. However, notice that the entries of $\boldsymbol{c}_n$ cannot be independent from each other, because the polynomial $p_n(\boldsymbol{x})$ must be factorable as a product of linear forms. The factorability of $p_n(\boldsymbol{x})$ enforces constraints on the entries of $\boldsymbol{c}_n$, which are polynomials of degree $(n+1)$ on $M_n$ variables, the so-called Brill's equations [19]. 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 (3.16) will automatically satisfy Brill's equations. Understanding how to use Brill's equations to improve the estimation of $\boldsymbol{c}_n$ 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 $\boldsymbol{c}_n$ from equation (3.16). To this end, notice that if the number of hyperplanes $n$ was known, we could immediately recover $\boldsymbol{c}_n$ as the eigenvector of $L_n^T L_n$ associated with its smallest eigenvalue. However, since the above linear system (3.16) depends explicitly on the number of hyperplanes $n$, we cannot estimate $\boldsymbol{c}_n$ 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 $\boldsymbol{c}_n$ is unique (up to a scale factor), as stated by the following theorem.

**Theorem 2 (Number of hyperplanes)** *Assume that a collection of $N \geq M_n - 1$ sample points $\{\boldsymbol{x}^j\}_{j=1}^N$ 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 (3.16), but computed with the Veronese map $\nu_i(\boldsymbol{x})$ of degree $i$. If the sample points are in general position and at least $K-1$ points correspond to each hyperplane, then:*

$$\text{rank}(L_i) \begin{cases} > M_i - 1, & i < n, \\ = M_i - 1, & i = n, \\ < M_i - 1, & i > n. \end{cases} \tag{3.17}$$

*Therefore, the number $n$ of hyperplanes is given by:*

$$\boxed{n = \min\{i : \text{rank}(L_i) = M_i - 1\}.} \tag{3.18}$$

*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, [34]), 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 (3.16) 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 15** *In the presence of noise, one cannot directly estimate $n$ from (3.18), 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 $\sigma_k$ 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 2 and the linear system in equation (3.16) allow us to determine the number of hyperplanes $n$ and the vector of coefficients $\boldsymbol{c}_n$, respectively, from sample points $\{\boldsymbol{x}^j\}_{i=1}^N$. The rest of the problem becomes now how to recover the normal vectors $\{\boldsymbol{b}_i\}_{i=1}^n$ from $\boldsymbol{c}_n$. Sections 3.3.2 and 3.3.3 present two algorithms for recovering the normal vectors based on polynomial factorization and polynomial differentiation and division, respectively.

### 3.3.2 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 3 (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 $n \leq 4$.*

#### GPCA as a polynomial factorization problem

From equations (3.13) and (3.14) 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 3 (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 16 (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*

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

*Notice that*

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

*maps a $K \times n$-dimensional space to an $M_n$-dimensional space. In general $M_n$ is much larger than $(K \times n - n + 1)$.*[11] *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 $p_n(\boldsymbol{x})$ is guaranteed by its definition as a product of linear functionals. In relation to the uniqueness of the factorization, it is clear that each $\boldsymbol{b}_i$ can be multiplied by an arbitrary scale to obtain the same $\boldsymbol{c}_n$ up to scale. Since we can fix the norm of $\boldsymbol{c}_n$ to be 1 when solving (3.16), we are actually free to choose the scale of $n-1$ of the $\boldsymbol{b}_i$'s only. The following proposition is a consequence of the well-known Gauss Lemma in Algebra (see page 181, [34]) and guarantees the uniqueness of the factorization of $p_n(\boldsymbol{x})$ 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.*

**Solving the polynomial factorization problem**

Knowing the existence and uniqueness of a solution to the polynomial factorization problem (Problem 3), we are now interested in finding an algorithm that recovers the $\boldsymbol{b}_i$'s from $\boldsymbol{c}_n$. 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*

$$\begin{aligned} p_2(\boldsymbol{x}) &= (\boldsymbol{b}_1^T\boldsymbol{x})(\boldsymbol{b}_2^T\boldsymbol{x}) = (b_{11}x_1 + b_{12}x_2 + b_{13}x_3)(b_{21}x_1 + b_{22}x_2 + b_{23}x_3) \\ &= \underbrace{(b_{11}b_{21})}_{c_{2,0,0}}x_1^2 + \underbrace{(b_{11}b_{22}+b_{12}b_{21})}_{c_{1,1,0}}x_1x_2 + \underbrace{(b_{11}b_{23}+b_{13}b_{21})}_{c_{1,0,1}}x_1x_3 + \\ &\quad \underbrace{(b_{12}b_{22})}_{c_{0,2,0}}x_2^2 + \underbrace{(b_{12}b_{23}+b_{13}b_{22})}_{c_{0,1,1}}x_2x_3 + \underbrace{(b_{13}b_{23})}_{c_{0,0,2}}x_3^2. \end{aligned}$$

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

*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 $\boldsymbol{c}_2 \in \mathbb{R}^6$ 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 $\boldsymbol{b}_1$ and $\boldsymbol{b}_2$ are only computable up to scale, we can actually divide $\boldsymbol{c}_2$ by $c_{0,2,0}$ (if nonzero) and set the last two entries of $\boldsymbol{b}_1$ and $\boldsymbol{b}_2$ 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 $\boldsymbol{b}_1$ and $\boldsymbol{b}_2$. 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*

$$\begin{bmatrix} b_{22} & b_{12} \\ b_{23} & b_{13} \end{bmatrix} \begin{bmatrix} b_{11} \\ b_{21} \end{bmatrix} = \begin{bmatrix} c_{1,1,0} \\ c_{1,0,1} \end{bmatrix} \tag{3.19}$$

*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 $\{\boldsymbol{b}_i\}_{i=1}^n$ from the roots of a polynomial $q_n(t)$ associated with the last $n+1 = 3$ coefficients of $p_n(\boldsymbol{x})$; and (2) solve for the first $K-2$ entries of $\{\boldsymbol{b}_i\}_{i=1}^n$ 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 $\boldsymbol{b}_i$:** Consider the last $n+1$ coefficients of $p_n(\boldsymbol{x})$:

$$[c_{0,\ldots,0,n,0}\ ,\ c_{0,\ldots,0,n-1,1}\ ,\ \cdots\ ,\ c_{0,\ldots,0,0,n}]^T \in \mathbb{R}^{n+1}, \tag{3.20}$$

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

$$\sum c_{0,\ldots,0,n_{K-1},n_K} x_{K-1}^{n_{K-1}} x_K^{n_K} = \prod_{i=1}^{n} \left(b_{iK-1}x_{K-1} + b_{iK}x_K\right). \tag{3.21}$$

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

$$q_n(t) = c_{0,\ldots,0,n,0}t^n + c_{0,\ldots,0,n-1,1}t^{n-1} + \cdots + c_{0,\ldots,0,0,n} \tag{3.22}$$

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

$$(b_{iK-1}\ ,\ b_{iK}) = (1\ ,\ -t_i) \qquad i = 1, \ldots, n. \tag{3.23}$$

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 $\boldsymbol{c}_n$ 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 (3.23). 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 $\boldsymbol{b}_i$:** 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

$$\frac{\partial p_n(\boldsymbol{x})}{\partial x_J} = \frac{\partial}{\partial x_J}\left(\prod_{i=1}^{n}(\boldsymbol{b}_i^T\boldsymbol{x})\right) = \sum_{i=1}^{n} b_{iJ}\left(\prod_{\ell=1}^{i-1}(\boldsymbol{b}_i^T\boldsymbol{x}) \prod_{\ell=i+1}^{n}(\boldsymbol{b}_i^T\boldsymbol{x})\right), \tag{3.24}$$

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

$$\sum c_{0,\ldots,0,1,n_{J+1},\ldots,n_K} x_{J+1}^{n_{J+1}} \cdots x_K^{n_K} = \sum_{i=1}^{n} b_{iJ} g_i^J(\boldsymbol{x}), \tag{3.25}$$

where

$$g_i^J(\boldsymbol{x}) = \prod_{\ell=1}^{i-1} \left( \sum_{j=J+1}^{K} b_{\ell j} x_j \right) \prod_{\ell=i+1}^{n} \left( \sum_{j=J+1}^{K} b_{\ell j} x_j \right) \tag{3.26}$$

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 (3.25) we get

$$\left[ \begin{array}{cccc} \mathcal{V}_1^J & \mathcal{V}_2^J & \cdots & \mathcal{V}_n^J \end{array} \right] \left[ \begin{array}{c} b_{1J} \\ b_{2J} \\ \vdots \\ b_{nJ} \end{array} \right] = \left[ \begin{array}{c} c_{0,\ldots,0,1,n-1,0,\ldots,0} \\ c_{0,\ldots,0,1,n-2,1,\ldots,0} \\ \vdots \\ c_{0,\ldots,0,1,0,0,\ldots,n-1} \end{array} \right] \tag{3.27}$$

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 $j = J+1, \ldots K$.

**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 $p_2(\boldsymbol{x}) = x_2 x_3 \in R_2(3)$. This is because it does not use *all* the entries of $\boldsymbol{c}_n$ 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 (3.27), 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 $\{\boldsymbol{b}_i\}_{i=1}^n$:

**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

$$h^J \doteq \sum_{i=1}^{n} \alpha_i h_1^J \cdots h_{i-1}^J h_{i+1}^J \cdots h_n^J = 0 \tag{3.28}$$

if and only if $\alpha_i = 0, \alpha_i \in \mathbb{R}$, $i = 1, \ldots, n$. If $n = 2$, (3.28) reduces to:

$$\alpha_1 h_2^J + \alpha_2 h_1^J = 0. \tag{3.29}$$

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 (3.28) by $h_1^J$ we obtain:

$$\frac{h^J}{h_1^J} = \alpha_1 \frac{h_2^J \cdots h_n^J}{h_1^J} + \sum_{i=2}^{n} \alpha_i \underbrace{h_2^J \cdots h_{i-1}^J h_{i+1}^J \cdots h_n^J}_{\text{polynomial in } R_{n-1}(K-J)} = 0. \tag{3.30}$$

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. ■

**Obtaining a unique solution for the degenerate cases**

Proposition 2 states that in order for the $K - 2$ linear systems in (3.27) 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 $2 \times 2$ minors of the matrix $B = [\boldsymbol{b}_1\ \boldsymbol{b}_2]^T \in \mathbb{R}^{2 \times K}$ are equal to zero. This implies that $\boldsymbol{b}_1$ is parallel to $\boldsymbol{b}_2$, violating the assumption of different subspaces. ■

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

$$p_3(\boldsymbol{x}) = (x_1 + x_2 + x_3)(x_1 + 2x_2 + 2x_3)(x_1 + 2x_2 + x_3).$$

*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 $\boldsymbol{c}_n$, 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 $p_n(\boldsymbol{x})$ 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 $\boldsymbol{x}$, hence on the $b_{ij}$'s, that gives a new vector of coefficients $\boldsymbol{c}'_n$ whose associated polynomial $q'_n(t)$ is non-degenerate. It is clear that we only need to modify the entries of each $\boldsymbol{b}_i$ associated to the last two variables. Thus, we consider the following linear transformation $T : \mathbb{R}^K \to \mathbb{R}^K$:

$$\boldsymbol{x} = T\boldsymbol{y} = \begin{bmatrix} 1 & 0 & \cdots & 0 & t & t \\ 0 & 1 & & 0 & t & t \\ \vdots & & \ddots & & & \vdots \\ & & & & 1 & t \\ 0 & 0 & \cdots & & 0 & 1 \end{bmatrix} \boldsymbol{y}. \tag{3.31}$$

Under this transformation, the polynomial $p_n(\boldsymbol{x})$ becomes:

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

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:

$$b'_{rK-1}(t)b'_{sK}(t) = b'_{sK-1}(t)b'_{rK}(t) \tag{3.32}$$

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 $\boldsymbol{c}'_n$ of the new polynomial $p'_n(\boldsymbol{y})$. The following proposition establishes the relationship between $\boldsymbol{c}_n$ and $\boldsymbol{c}'_n$:

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

$$(\boldsymbol{\ell}_1^T\boldsymbol{y})^{n_1}\ (\boldsymbol{\ell}_2^T\boldsymbol{y})^{n_2}\ \cdots\ (\boldsymbol{\ell}_K^T\boldsymbol{y})^{n_K}, \tag{3.33}$$

*where $\boldsymbol{\ell}_j^T$ is the $j$-th row of $T$.*

*Proof.* Let $p_n^1(\boldsymbol{x}), p_n^2(\boldsymbol{x}) \in R_n^F(K)$ and $\alpha, \beta \in \mathbb{R}$. Then the polynomial $\alpha p_n^1(\boldsymbol{x}) + \beta p_n^2(\boldsymbol{x})$ is transformed by $T$ into $\alpha p_n^1(L\boldsymbol{y}) + \beta p_n^2(L\boldsymbol{y})$. 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 $x_1^{n_1}x_2^{n_2}\cdots x_K^{n_K} = (e_1^T\boldsymbol{x})^{n_1}(e_2^T\boldsymbol{x})^{n_2}\cdots(e_K^T\boldsymbol{x})^{n_K}$, where $\{e_j\}_{j=1}^K$ is the standard basis for $\mathbb{R}^K$. We obtain $(e_1^TT\boldsymbol{y})^{n_1}(e_2^TT\boldsymbol{y})^{n_2}\cdots(e_K^TT\boldsymbol{y})^{n_K}$, or equivalently $(\boldsymbol{\ell}_1^T\boldsymbol{y})^{n_1}\ (\boldsymbol{\ell}_2^T\boldsymbol{y})^{n_2}\ \cdots\ (\boldsymbol{\ell}_K^T\boldsymbol{y})^{n_K}$. ∎

**Remark 17** *Due to the upper triangular structure of $T$ in (3.31), 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 $p'_n(\boldsymbol{y})$ will have no repeated roots. Therefore, we can apply the previously described factorization algorithm to the coefficients $\boldsymbol{c}'_n$ of $p'_n(\boldsymbol{y})$ to obtain the set of transformed normal vectors $\{\boldsymbol{b}'_i\}_{i=1}^n$. Since by definition of $p'_n(\boldsymbol{y})$ we have $\boldsymbol{b}_i'^T = \boldsymbol{b}_i^TT$, the original normal vectors are given by $\boldsymbol{b}_i = T^{-T}\boldsymbol{b}'_i$. It turns out that, due to the particular structure of $T$, we do not actually need to compute $T^{-T}$. We can obtain $\{\boldsymbol{b}_i\}_{i=1}^n$ directly from $\{\boldsymbol{b}'_i\}_{i=1}^n$ and $t$ as follows:

$$\begin{array}{lcll} b_{ij} & = & b'_{ij}, & i = 1,\ldots,n,\ j = 1,\ldots,K-2 \\ b_{iK-1} & = & b'_{iK-1} - t\sum_{j=1}^{K-2} b_{ij}, & i = 1,\ldots,n \\ b_{iK} & = & b'_{iK} - t\sum_{j=1}^{K-1} b_{ij}, & i = 1,\ldots,n. \end{array} \tag{3.34}$$

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:*

$$T = \begin{bmatrix} 1 & t & t \\ 0 & 1 & t \\ 0 & 0 & 1 \end{bmatrix} \tag{3.35}$$

*and*

$$\tilde{T} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 3t & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 3t & t & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 3t^2 & 2t & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 6t^2 & 2t^2+2t & 2t & 2t & 1 & 0 & 0 & 0 & 0 & 0 \\ 3t^2 & 2t^2 & 2t & t^2 & t & 1 & 0 & 0 & 0 & 0 \\ t^3 & t^2 & 0 & t & 0 & 0 & 1 & 0 & 0 & 0 \\ 3t^3 & t^3+2t^2 & t^2 & 2t^2+t & t & 0 & 3t & 1 & 0 & 0 \\ 3t^3 & 2t^3+t^2 & 2t^2 & t^3+2t^2 & t^2+t & t & 3t^2 & 2t & 1 & 0 \\ t^3 & t^3 & t^2 & t^3 & t^2 & t & t^3 & t^2 & t & 1 \end{bmatrix}. \tag{3.36}$$

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 2 (Polynomial Factorization Algorithm (PFA) for Mixtures of Hyperplanes)**

Given sample points $\{\boldsymbol{x}^j\}_{j=1}^N$ 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 $\{\boldsymbol{b}_i \in \mathbb{R}^K\}_{i=1}^n$ as follows:

1. Apply the Veronese map of order $i$, for $i = 1, 2, \ldots$, to the vectors $\{\boldsymbol{x}^j\}_{j=1}^N$ and form the matrix $L_i$ in (3.16). Stop when $\text{rank}(L_i) = M_i - 1$ and set the number of hyperplanes $n$ to be the current $i$. Then solve for $\boldsymbol{c}_n$ from $L_n \boldsymbol{c}_n = 0$ and normalize so that $\|\boldsymbol{c}_n\| = 1$.

2. (a) Get the coefficients of the univariate polynomial $q_n(t)$ from the last $n+1$ entries of $\boldsymbol{c}_n$.

   (b) If the first $\ell$, $0 \leq \ell \leq n$, coefficients of $q_n(t)$ are equal to zero, set $(b_{iK-1}, b_{iK}) = (0, 1)$ for $i = 1, \ldots, \ell$. Then use (3.23) 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 $\boldsymbol{x} = T\boldsymbol{y}$ in (3.31) and repeat 2(a), 2(b) and 2(c) for the transformed polynomial $p'_n(\boldsymbol{y})$ 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 (3.27) for $J = K-2, \ldots, 1$. If a transformation $T$ was used in 2(d), then compute $\boldsymbol{b}_i$ from $\boldsymbol{b}'_i$ and $t$ using equation (3.34).

### 3.3.3 Estimating the hyperplanes: the polynomial differentiation algorithm (PDA)

The main attraction of the polynomial factorization algorithm (PFA) (Algorithm 2) 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 $\boldsymbol{c}_n$ 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 $\boldsymbol{b}_i$'s are unknown

In this section, we propose a new algorithm for estimating the normal vectors $\{\boldsymbol{b}_i\}_{i=1}^n$ which is based on *polynomial differentiation* rather than *polynomial factorization*. We show that given $\boldsymbol{c}_n$ one can recover $\{\boldsymbol{b}_i\}_{i=1}^n$ by evaluating the derivatives of $p_n(\boldsymbol{x})$ 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 $\boldsymbol{c}_n$ from the linear system $L_n\boldsymbol{c}_n = 0$, as described in Section 3.3.1.

2. Compute the normal vectors $\{\boldsymbol{b}_i\}_{i=1}^n$ as the derivative of $p_n(\boldsymbol{x})$ evaluated at the $n$ points $\{\boldsymbol{y}_i \in S_i\}_{i=1}^n$, 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 $p_n(\boldsymbol{x})$, as we describe below.

**Obtaining normal vectors by differentiation**

Imagine, for the time being, that we were given a set of $n$ points $\{\boldsymbol{y}_i\}_{i=1}^n$, each one lying on only one of the $n$ hyperplanes, that is $\boldsymbol{y}_i \in S_i$ 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 $p_n(\boldsymbol{x})$ evaluated at each $\boldsymbol{y}_i$. We have:

$$Dp_n(\boldsymbol{x}) = \frac{\partial p_n(\boldsymbol{x})}{\partial \boldsymbol{x}} = \frac{\partial}{\partial \boldsymbol{x}} \prod_{i=1}^{n} (\boldsymbol{b}_i^T \boldsymbol{x}) = \sum_{i=1}^{n} (\boldsymbol{b}_i) \prod_{\ell \neq i} (\boldsymbol{b}_\ell^T \boldsymbol{x}). \tag{3.37}$$

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

$$\boxed{\boldsymbol{b}_i = \frac{Dp_n(\boldsymbol{y}_i)}{\|Dp_n(\boldsymbol{y}_i)\|} \quad i = 1, \ldots, n.} \tag{3.38}$$

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 $p_n(\boldsymbol{x})$ 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.

**Obtaining one point per hyperplane: an algebraic solution**

Consider a random line $\mathcal{L} \doteq \{t\boldsymbol{v} + \boldsymbol{x}_0, \ t \in \mathbb{R}\}$ with direction $\boldsymbol{v}$ and base point $\boldsymbol{x}_0$. 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 $p_n(t\boldsymbol{v} + \boldsymbol{x}_0) = 0$, the $n$ points $\{\boldsymbol{y}_i\}_{i=1}^n$ can be obtained as

$$\boxed{\boldsymbol{y}_i = t_i \boldsymbol{v} + \boldsymbol{x}_0 \quad i = 1, \ldots, n,} \tag{3.39}$$

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

$$q_n(t) = p_n(t\boldsymbol{v} + \boldsymbol{x}_0) = \prod_{i=1}^{n} (t\boldsymbol{b}_i^T \boldsymbol{v} + \boldsymbol{b}_i^T \boldsymbol{x}_0). \tag{3.40}$$

The problem is now how to choose $\boldsymbol{v}$ and $\boldsymbol{x}_0$. 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 $\boldsymbol{x}_0 = 0$ or if $\boldsymbol{x}_0$ is parallel to $\boldsymbol{v}$, then the number of roots is either one or infinity. These two cases can be obviously avoided. Another degenerate configuration happens when the direction $\boldsymbol{v}$ 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 $\boldsymbol{v}$ is parallel to one of the hyperplanes if and only if $\boldsymbol{b}_i^T \boldsymbol{v} = 0$ for some $i = 1, \ldots, n$, this degenerate case can be avoided by choosing $\boldsymbol{v}$ such that $p_n(\boldsymbol{v}) \neq 0$. Therefore, in the absence of noise, we randomly choose $\boldsymbol{x}_0$ and $\boldsymbol{v}$ on the unit sphere and if the above conditions are met, we proceed with the computation of $t_i$, $\boldsymbol{y}_i$ and $\boldsymbol{b}_i$, 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 $\{\boldsymbol{b}_{i\ell}\}$ 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 $\boldsymbol{X}$ 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 18 (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 $\boldsymbol{x}_0 = [0, \ldots, 0, 0, 1]^T$ and $\boldsymbol{v} = [0, \ldots, 0, 1, 0]^T$. Therefore, we can summarize together the PFA discussed in the preceding section and the PDA described in this section in the following algorithm.*

**Algorithm 3 (PFA and Algebraic PDA for Mixtures of Hyperplanes)**

**solve** $L_n \boldsymbol{c}_n = 0$;

**set** $p_n(\boldsymbol{x}) = \boldsymbol{c}_n^T \nu_n(\boldsymbol{x})$;

**compute** the $n$ roots $t_1, \ldots, t_n$ of the univariate polynomial $q_n(t) = p_n(t\boldsymbol{v} + \boldsymbol{x}_0)$ with:

- PFA: $\boldsymbol{x}_0 = [0, \ldots, 0, 0, 1]^T$ and $\boldsymbol{v} = [0, \ldots, 0, 1, 0]^T$;
- PDA: $\boldsymbol{x}_0$ and $\boldsymbol{v}$ chosen randomly;

**obtain** the hyperplane normal vectors $\boldsymbol{b}_i$:

- PFA: solve $K - 2$ linear systems of equations to find the normal vectors $\boldsymbol{b}_i$;
- PDA: differentiate $p_n(\boldsymbol{x})$ to obtain $\boldsymbol{b}_i = \frac{Dp_n(\boldsymbol{y}_i)}{\|Dp_n(\boldsymbol{y}_i)\|}$ at $\boldsymbol{y}_i = \boldsymbol{x}_0 + \boldsymbol{v}t_i$.

**Obtaining one point per hyperplane: a recursive solution**

As we will see in Section 3.5, 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 $\boldsymbol{y}_n$ lying on one of the hyperplanes by checking that $p_n(\boldsymbol{y}_n) = 0$. Since we are given a set of data points $\boldsymbol{X} = \{\boldsymbol{x}^j\}_{j=1}^n$ lying on the hyperplanes, in principle we can choose $\boldsymbol{y}_n$ to be any of the data points. However, in the presence of noise and outliers a random choice of $\boldsymbol{y}_n$ may be far from the true hyperplanes. Another possibility is to choose $\boldsymbol{y}_n$ as the point in $\boldsymbol{X}$ that minimizes $|p_n(\boldsymbol{x})|$. However, the above choice has the following problems in the presence of noise:

1. The value $|p_n(\boldsymbol{x})|$ is merely an *algebraic* error, i.e., it does not really represent the *geometric* distance from $\boldsymbol{x}$ to 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 $\{\boldsymbol{b}_i\}_{i=1}^n$.

2. Points $\boldsymbol{x}$ lying close to the intersection of two or more subspaces are more likely to be chosen, because two or more factors in $p_n(\boldsymbol{x}) = (\boldsymbol{b}_1^T\boldsymbol{x})\cdots(\boldsymbol{b}_n^T\boldsymbol{x})$ are approximately zero, which yields a smaller value for $|p_n(\boldsymbol{x})|$. Furthermore, since $Dp_n(\boldsymbol{x}) = 0$ for $\boldsymbol{x}$ 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 (3.37) that for arbitrary $\boldsymbol{x}$ the vector $Dp_n(\boldsymbol{x})$ is a linear combination of the normal vectors $\{\boldsymbol{b}_i\}_{i=1}^n$. Thus if $\boldsymbol{x}$ is close to two subspaces 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 2** *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*

$$\|\boldsymbol{x} - \tilde{\boldsymbol{x}}\| = n\frac{|p_n(\boldsymbol{x})|}{\|Dp_n(\boldsymbol{x})\|} + O\big(\|\boldsymbol{x} - \tilde{\boldsymbol{x}}\|^2\big). \tag{3.41}$$

*Proof.* Replace $m = 1$ in the proof of Lemma 3. ∎

The importance of Lemma 2 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 (3.41) 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:

$$\boldsymbol{y}_n = \underset{\boldsymbol{x}\in\boldsymbol{X}:Dp_n(\boldsymbol{x})\neq 0}{\arg\min} \frac{|p_n(\boldsymbol{x})|}{\|Dp_n(\boldsymbol{x})\|}, \tag{3.42}$$

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 (3.42), 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

$$p_{n-1}(\boldsymbol{x}) \doteq p_n(\boldsymbol{x})/(\boldsymbol{b}_n^T\boldsymbol{x}) = (\boldsymbol{b}_1^T\boldsymbol{x})\cdots(\boldsymbol{b}_{n-1}^T\boldsymbol{x}) = 0. \tag{3.43}$$

Therefore, similarly to (3.42), 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 4 (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** $i = n : 1$,

$$\boldsymbol{y}_i = \underset{\boldsymbol{x}\in\boldsymbol{X}:Dp_i(\boldsymbol{x})\neq 0}{\arg\min} \frac{|p_i(\boldsymbol{x})|}{\|Dp_i(\boldsymbol{x})\|}, \tag{3.44}$$

$$\boldsymbol{b}_i = \frac{Dp_i(\boldsymbol{y}_i)}{\|Dp_i(\boldsymbol{y}_i)\|}, \tag{3.45}$$

$$p_{i-1}(\boldsymbol{x}) = \frac{p_i(\boldsymbol{x})}{\boldsymbol{b}_i^T\boldsymbol{x}}, \tag{3.46}$$

**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 19 (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 20** *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*

$$\boldsymbol{y}_{i-1} = \underset{\boldsymbol{x} \in \boldsymbol{X}: Dp_n(\boldsymbol{x}) \neq 0}{arg\,\min} \frac{\frac{|p_n(\boldsymbol{x})|}{\|Dp_n(\boldsymbol{x})\|} + \delta}{|(\boldsymbol{b}_i^T \boldsymbol{x}) \cdots (\boldsymbol{b}_n^T \boldsymbol{x})| + \delta}, \tag{3.47}$$

*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 21 (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 $\boldsymbol{y}_i$ and obtain $\boldsymbol{b}_i$ from $Dp_i(\boldsymbol{y}_i)$ according to PDA-rec; then set $\boldsymbol{x}_0 = \boldsymbol{y}_i$ and $\boldsymbol{v} = \boldsymbol{b}_i$ and solve for the roots of $q_n(t) = p_n(t\boldsymbol{v} + \boldsymbol{x}_0)$. Choose the root $t^* = \min(|t_i|)$ and obtain a new point lying on one of the hyperplanes as $\boldsymbol{y}_i \leftarrow t^* \boldsymbol{v} + \boldsymbol{x}_0$.*

## 3.4 Estimating a mixture of subspaces of equal dimension $k < K$

We showed in Section 3.2 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 3.3 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 3.3 to the case of subspaces of equal dimension $0 < k_1 = \cdots = k_n = k < K$. In Section 3.4.1 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 3.4.2 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 3.4.3 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.

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

In this section, we show that the segmentation of a sample set $\boldsymbol{X}$ drawn from $n$ $k$-dimensional subspaces of a space of dimension $K > k$ is preserved after projecting the sample set $\boldsymbol{X}$ onto a *generic* subspace $Q$ of dimension $k+1$ ($\leq K$). An example is shown in Figure 3.2, 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.

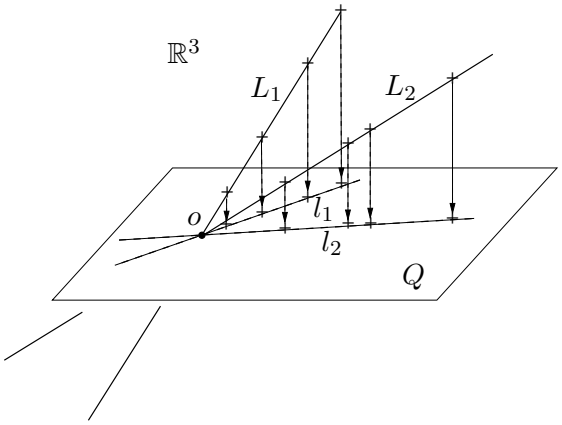



Figure 3.2: Two 1-dimensional subspaces $L_1, L_2$ in $\mathbb{R}^3$ projected onto a 2-dimensional plane $Q$. Clearly, the membership of each sample (labeled as "+"on the lines) is preserved through the projection.

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

$$\pi_Q : \mathbb{R}^K \to Q \qquad \boldsymbol{x} \mapsto \boldsymbol{x}', \tag{3.48}$$

and the projection of $S_i$ as $S_i' \doteq \pi_Q(S_i)$. Also let $\boldsymbol{X}' \doteq \pi_Q(\boldsymbol{X})$ 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*[12], then $\dim(S_i')$ remains to be $k$ – no reduction in the dimension of each subspace[13], and there is a one-to-one correspondence between $S_i'$ and $S_i$ – no reduction in the number of subspaces[14] $n$. 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 $Z' = \cup_{i=1}^n S_i'$.

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

[13] This requires that $Q$ be transversal to each $S_i^\perp$, i.e., 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 [25].

[14] 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.

The effect of such a projection can also be explained from an algebraic viewpoint. As we discussed in Sections 3.2, 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 3.3. 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 3.3 (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 $\iota_Q: Q\rightarrow\mathbb{R}^K$ 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:

$$\{\boldsymbol{x}\in\cup S_i\}\xrightarrow{\pi_Q}\{\boldsymbol{x}'\in\cup S'_i\}\xrightarrow[PDA]{PFA}\{\boldsymbol{b}'\in\cup {S'_i}^\perp\}\xrightarrow{\iota_P}\{\boldsymbol{b}\in\cup {S_i}^\perp\}. \tag{3.49}$$

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.

### 3.4.2 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 [29]. 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 3.3, in which a collection of lines can also be interpreted as a collection of planes.

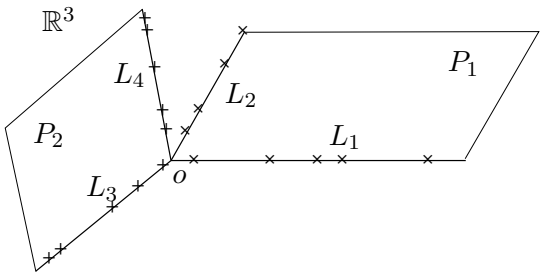



Figure 3.3: A set of samples that can be interpreted as coming either from four 1-dimensional subspaces $L_1, L_2, L_3, L_4$ in $\mathbb{R}^3$, or from two 2-dimensional subspaces $P_1, P_2$ in $\mathbb{R}^3$.

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 [23]. Therefore, the $k = \dim(S_i)$ and the number of subspaces $n$ are always uniquely defined in a purely algebraic fashion. In this sense, for the case shown in Figure 3.3, 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.

**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 (3.9) by applying the Veronese map of degree $i = 1, 2, \ldots$ to the projected data. From our analysis in Sections 3.2 and 3.4.1, there is a unique polynomial of degree $n$ generating $I(Z')$ whose coefficients are in the null space of $L_n(k+1)$. Thus 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 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 $p_n(\boldsymbol{x})$. Therefore, 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

$$\boxed{n = \min\{i : \text{rank}(L_i(k+1)) = M_i(k+1) - 1\}.} \tag{3.50}$$

**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 (3.9), 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 (3.54) we have that rank$(L_n(\ell+1)) = M_n(\ell+1) - 1$. When $\ell > k$, then equation (3.9) has more than one solution, thus rank$(L_n(\ell+1)) < M_n(\ell+1) - 1$. Therefore, if $n$ is known, we can compute $k$ as

$$\boxed{k = \min\{\ell : \text{rank}(L_n(\ell+1)) = M_n(\ell+1) - 1\}.} \tag{3.51}$$

**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 (3.9), 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 $\ell$ for which there exist an $i$ such that $L_i(\ell+1)$ drops rank, that is

$$\boxed{k = \min\{\ell : \exists i \geq 1 \text{ such that } \text{rank}(L_i(\ell+1)) < M_i(\ell+1)\}.} \tag{3.52}$$

Given $k$ one can compute $n$ as in equation (3.54).

More formally, we have shown the following.

**Theorem 4 (Number of subspaces $n$ and their dimension $k$)** *Assume that a collection of $N \geq M_n(k+1) - 1$ sample points $\{\boldsymbol{x}^j\}_{j=1}^N$ 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 (3.9), but computed with the Veronese map $\nu_i$ 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:*

$$\boxed{k = \min\{\ell : \exists i \geq 1 \text{ such that } \text{rank}(L_i(\ell+1)) < M_i(\ell+1)\},} \tag{3.53}$$

*and the number $n$ of subspaces is given by:*

$$\boxed{n = \min\{i : \text{rank}(L_i(k+1)) = M_i(k+1) - 1\}.} \tag{3.54}$$

*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 (3.54) is a direct consequence of Theorem 2. The rest of the theorem follows from the analysis given in this section. ∎

**Corollary 1** *The vector of coefficients $\boldsymbol{c}_n \in \mathbb{R}^{M_n(k+1)}$ of the homogeneous polynomial $p_n(\boldsymbol{x})$ 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 22** *The above statement indirectly claims that in order to* linearly *estimate the polynomial $p_n(\boldsymbol{x})$, 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 23** *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 15). 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.*

### 3.4.3 Estimating the subspaces: the polynomial differentiation algorithm (PDA)

As we discussed in Section 3.4.1, when $k < K-1$, there are both geometric and algebraic reasons that suggest that we should first project the sample set $\boldsymbol{X}$ 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 3.3 (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 5 (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 4.
2. Apply PCA with $k+1$ principal components to the original data points $[\boldsymbol{x}^1, \ldots, \boldsymbol{x}^N] \in \mathbb{R}^{K \times N}$ to obtain the projected data points $[\boldsymbol{x}'^1, \ldots, \boldsymbol{x}'^N] \in \mathbb{R}^{(k+1)\times N}$.
3. Apply GPCA for hyperplanes (PFA or PDA) to the projected data $[\boldsymbol{x}'_1, \ldots, \boldsymbol{x}'_N] \in \mathbb{R}^{k+1 \times N}$ to obtain a collection of normal vectors $\{\boldsymbol{b}'_i \in \mathbb{R}^{k+1}\}_{i=1}^n$.
4. Cluster the original data by assigning point $\boldsymbol{x}^j$ to subspace $S_i$ if
$$i = \arg \min_{\ell=1,\ldots,n} |\boldsymbol{b}'^T_\ell \boldsymbol{x}'^j|. \tag{3.55}$$
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 5 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,[15] 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 $\ell$, let $p'_{n\ell}(\boldsymbol{x}')$ be the polynomial representing the collection of projected subspaces $\{S'_i = \pi_{Q^\ell}(S_i)\}_{i=1}^n$, and let $p_{n\ell}(\boldsymbol{x}) = p'_{n\ell}(\pi_{Q^\ell}(\boldsymbol{x}))$ be its corresponding polynomial in $I(Z)$. Then, from the analysis of Section 3.3.3 we have that if $\boldsymbol{y}_i \in S_i$ then

$$\left.\frac{\partial p_{n\ell}(\boldsymbol{x})}{\partial \boldsymbol{x}}\right|_{\boldsymbol{x}=\boldsymbol{y}_i} \in S_i^\perp \text{ for all } \ell = 1,\ldots,m. \tag{3.56}$$

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

**Lemma 3** *The Euclidean distance from point $\boldsymbol{x}$ to its closest subspace is given by*

$$\|\boldsymbol{x} - \tilde{\boldsymbol{x}}\| = n\sqrt{P_n(\boldsymbol{x})\big(DP_n(\boldsymbol{x})^T DP_n(\boldsymbol{x})\big)^\dagger P_n(\boldsymbol{x})^T} + O\big(\|\boldsymbol{x} - \tilde{\boldsymbol{x}}\|^2\big), \tag{3.57}$$

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

*Proof.* The projection $\tilde{\boldsymbol{x}}$ of a point $\boldsymbol{x}$ 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

$$\begin{aligned} \min \quad & \|\tilde{\boldsymbol{x}} - \boldsymbol{x}\|^2 \\ \text{subject to} \quad & p_{n\ell}(\tilde{\boldsymbol{x}}) = 0 \quad \ell = 1,\ldots,m. \end{aligned} \tag{3.58}$$

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

$$\min_{\tilde{\boldsymbol{x}},\lambda} \|\tilde{\boldsymbol{x}} - \boldsymbol{x}\|^2 + P_n(\tilde{\boldsymbol{x}})\lambda. \tag{3.59}$$

[15] Recall from Section 3.2 that the set of subspaces $Q$ that fail to preserve the segmentation is a zero-measure set.

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

$$2(\tilde{\boldsymbol{x}} - \boldsymbol{x}) + DP_n(\tilde{\boldsymbol{x}})\lambda. \tag{3.60}$$

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

$$\begin{aligned} \lambda &= 2\big(DP_n(\tilde{\boldsymbol{x}})^T DP_n(\tilde{\boldsymbol{x}})\big)^\dagger DP_n(\tilde{\boldsymbol{x}})^T \boldsymbol{x} && (3.61) \\ \|\tilde{\boldsymbol{x}} - \boldsymbol{x}\|^2 &= \frac{1}{2}\boldsymbol{x}^T DP_n(\tilde{\boldsymbol{x}})\lambda, && (3.62) \end{aligned}$$

where we have used the fact that $(DP_n(\tilde{\boldsymbol{x}}))^T\tilde{\boldsymbol{x}} = nP_n(\tilde{\boldsymbol{x}}) = 0$. After replacing (3.61) on (3.62) we obtain that the squared distance from $\boldsymbol{x}$ to its closest subspace can be expressed as

$$\|\tilde{\boldsymbol{x}} - \boldsymbol{x}\|^2 = \boldsymbol{x}^T DP_n(\tilde{\boldsymbol{x}})\big(DP_n(\tilde{\boldsymbol{x}})^T DP_n(\tilde{\boldsymbol{x}})\big)^\dagger DP_n(\tilde{\boldsymbol{x}})^T \boldsymbol{x}. \tag{3.63}$$

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

$$\|\tilde{\boldsymbol{x}} - \boldsymbol{x}\|^2 \approx n^2 P_n(\boldsymbol{x})\big(DP_n(\boldsymbol{x})^T DP_n(\boldsymbol{x})\big)^\dagger P_n(\boldsymbol{x})^T, \tag{3.64}$$

which completes the proof. ∎

Thanks to Lemma 3, we can choose a point $\boldsymbol{y}_n$ in the dataset $\boldsymbol{X}$ that lies on (close to) one of the subspaces as:

$$\boldsymbol{y}_n = \underset{\boldsymbol{x}\in\boldsymbol{X}: DP_n(\boldsymbol{x})\neq 0}{\arg\min} P_n(\boldsymbol{x})\big(DP_n(\boldsymbol{x})^T DP_n(\boldsymbol{x})\big)^\dagger P_n(\boldsymbol{x})^T. \tag{3.65}$$

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

$$p_{n,\ell-1}(\boldsymbol{x}) = \frac{p_{n\ell}(\boldsymbol{x})}{\boldsymbol{b}_{n\ell}^T\boldsymbol{x}}. \tag{3.66}$$

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 $Q^\ell$, $\ell = 1, \ldots, m$. 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 6 (PDA for Mixtures of Subspaces of Equal Dimension $k < K-1$)**

**obtain** the number of subspaces $n$ and their dimension $k$ as in Theorem 4.

**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**

$$P_i(\boldsymbol{x}) = [p_{i1}(\boldsymbol{x}), \ldots, p_{im}(\boldsymbol{x})] \in \mathbb{R}^{1\times m}, \tag{3.67}$$

$$\boldsymbol{y}_i = \underset{\boldsymbol{x}\in\boldsymbol{X}: DP_i(\boldsymbol{x})\neq 0}{\arg\min} P_i(\boldsymbol{x})\big(DP_i(\boldsymbol{x})^T DP_i(\boldsymbol{x})\big)^\dagger P_i(\boldsymbol{x})^T, \tag{3.68}$$

$$\boldsymbol{b}_{i\ell} = \frac{Dp_{i\ell}(\boldsymbol{x})}{\|Dp_{i\ell}(\boldsymbol{x})\|}, \text{ for } \ell = 1, \ldots, m, \tag{3.69}$$

$$p_{i-1,\ell} = \frac{p_{i\ell}(\boldsymbol{x})}{\boldsymbol{b}_{i\ell}^T\boldsymbol{x}}, \text{ for } \ell = 1, \ldots, m, \tag{3.70}$$

$$B_i = PCA([\boldsymbol{b}_{i1}, \ldots, \boldsymbol{b}_{im}]) \tag{3.71}$$

**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.**

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

Our analysis in Sections 3.3 and 3.4 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 3.2, we can only obtain a basis $\{p_{n\ell}(\boldsymbol{x})\}$ for those factorable polynomials and each element in the basis may not be factorable. In Section 3.5.1 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 $\{p_{n\ell}(\boldsymbol{x})\}$, even when they are not factorable. Given the normals to that subspace $\{\boldsymbol{b}_{n\ell} \in S_n^\perp\}$ 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 $p_{n-1,\ell}(\boldsymbol{x})$ defining the remaining $n-1$ subspaces. However, in this case we cannot perform polynomial division, because the given polynomials $\{p_{n\ell}(\boldsymbol{x})\}$ may not be factorable. In Section 3.5.2, we derive an algorithm that uses the data and the estimated normals to estimate the polynomials $\{p_{n-1,\ell}(\boldsymbol{x})\}$, without performing polynomial division.

### 3.5.1 Obtaining subspace bases by polynomial differentiation

Recall from Section 3.2 that given a set of points $\boldsymbol{X} = \{\boldsymbol{x}^j\}_{j=1}^N$ 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 $\{p_{n\ell}(\boldsymbol{x}) = \boldsymbol{c}_{n\ell}^T \nu_n(\boldsymbol{x})\}$ whose coefficients lie in the $\big(m = \dim(I(Z))\big)$-dimensional null space of the embedded data matrix $L_n \in \mathbb{R}^{N \times M_n}$, i.e., $L_n \boldsymbol{c}_{n\ell} = 0$ 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 $\{p_{n\ell}(\boldsymbol{x})\}_{\ell=1}^m$.

As we also hinted in Section 3.2, 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 4** *Let $p$ be any polynomial in the ideal $I$ of an algebraic set $Z$, i.e., $p(\boldsymbol{x}) = 0, \forall \boldsymbol{x} \in Z$. If $T_{\boldsymbol{x}_0} Z$ is the (Zariski) tangent space to $Z$ at a smooth point $\boldsymbol{x}_0$, then the derivative of $p(\boldsymbol{x})$ at $\boldsymbol{x}_0$ satisfies:*

$$\boldsymbol{t}^T \frac{\partial p(\boldsymbol{x})}{\partial \boldsymbol{x}}\Big|_{\boldsymbol{x}_0} = 0, \quad \forall \boldsymbol{t} \in T_{\boldsymbol{x}_0} Z. \tag{3.72}$$

*Furthermore, $(T_{\boldsymbol{x}_0} Z)^\perp = \text{span}\Big\{ \frac{\partial p(\boldsymbol{x})}{\partial \boldsymbol{x}}\Big|_{\boldsymbol{x}_0}, \forall p \in I \Big\}$.*

*Proof.* Equation (3.72) is obvious since $p(\boldsymbol{x}) \equiv 0$ on $Z$ and the left hand side is merely a directional derivative along $\boldsymbol{t}$ at the regular point $\boldsymbol{x}_0$. The fact that the derivatives span the entire normal space is the consequence of the general dimension theory for algebraic varieties [6, 22, 15]. ∎

Notice that, for a particular $p(\boldsymbol{x})$ in one of the homogeneous components, say $I_{n'}$ ($n' \geq n$), of the ideal $I$, its derivative could be zero at $\boldsymbol{x}_0$.[16] 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.[17] To see this more clearly, let $\{\boldsymbol{y}_i \in S_i\}_{i=1}^n$ be a set of $n$ points each one lying on only one of the subspaces. Also let $\boldsymbol{c}_n$ be a vector in the null space of $L_n$. By construction, even though $\boldsymbol{c}_n$ may not correspond to a factorable polynomial, it can be written as a linear combination of vectors $\boldsymbol{c}_{n\ell}$ which correspond to factorable polynomials, i.e., $\boldsymbol{c}_n = \sum \alpha_\ell \boldsymbol{c}_{n\ell}$. Then

$$\frac{\partial}{\partial \boldsymbol{x}} \boldsymbol{c}_n^T \nu_n(\boldsymbol{x})\Big|_{\boldsymbol{x}=\boldsymbol{y}_i} = \frac{\partial}{\partial \boldsymbol{x}} \sum_\ell \alpha_\ell \boldsymbol{c}_{n\ell}^T \nu_n(\boldsymbol{x})\Big|_{\boldsymbol{x}=\boldsymbol{y}_i} = \sum_\ell \alpha_\ell \boldsymbol{b}_{i\ell}, \tag{3.73}$$

where $\boldsymbol{b}_{i\ell} \in S_i^\perp$ is a normal vector to subspace $S_i$. Therefore, although $\boldsymbol{c}_n^T \nu_n(\boldsymbol{x})$ may not be factorable, its derivative at $\boldsymbol{y}_i$ 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 5 (Polynomial differentiation)** *For the GPCA problem, if the given sample set $\boldsymbol{X}$ is such that $\dim(\text{null}(L_n)) = \dim(I_n)$ and one generic point $\boldsymbol{y}_i$ is given for each subspace $S_i$, then we have*

$$S_i^\perp = \text{span}\Big\{ \frac{\partial}{\partial \boldsymbol{x}} \boldsymbol{c}_n^T \nu_n(\boldsymbol{x})\Big|_{\boldsymbol{x}=\boldsymbol{y}_i}, \ \forall \boldsymbol{c}_n \in \text{null}(L_n) \Big\}. \tag{3.74}$$

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

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

Theorem 5 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 $\{\boldsymbol{y}_i \in S_i\}$, 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 $\{\boldsymbol{y}_i\}$:

- Compute a basis for the null space null$(L_n)$ using, for example, SVD.
- Evaluate the derivative of the (possibly nonfactorable) polynomial $\boldsymbol{c}_n^T \nu_n(\boldsymbol{x})$ at $\boldsymbol{y}_i$ for each $\boldsymbol{c}_n$ in the basis of null$(L_n)$ to obtain a set of normal vectors in $S_i^\perp$.
- 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:

$$\boxed{k_i = K - \text{rank}\big(DP_n(\boldsymbol{y}_i)\big), \quad i = 1, \ldots, n.} \tag{3.75}$$

**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:*

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

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

$$p_{21}(\boldsymbol{x}) = x_1 x_3 \qquad \textit{and} \qquad p_{22}(\boldsymbol{x}) = x_2 x_3.$$

*The derivatives of these two polynomials are:*

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

*which evaluated at $\boldsymbol{y}_1 = (1, 1, 0)^T \in S_1$ and $\boldsymbol{y}_2 = (0, 0, 1)^T \in S_2$ yield*

$$DP_2(\boldsymbol{y}_1) = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 1 \end{bmatrix} \qquad \textit{and} \qquad DP_2(\boldsymbol{y}_2) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}.$$

*By applying PCA to $DP_2(\boldsymbol{y}_1)$ and $DP_2(\boldsymbol{y}_2)$ we obtain a basis for $S_1^\perp$ and $S_2^\perp$ as*

$$B_1 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \qquad \textit{and} \qquad B_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}.$$

**Remark 24 (Overestimating the number of subspaces)** *Notice that one may replace $n$ with any $n' > n$ in Theorem 5 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 25 (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$(L_1)$, $p_1(\boldsymbol{x}) = x_3$, that also vanishes on $S_1 \cup S_2$. Furthermore, the derivative of $p_1(\boldsymbol{x})$ 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 8, 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 26 (Duality)** *Recall from our analysis in Section 3.4.1 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 5 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 $\boldsymbol{x}$ on a subspace $S_i$ gives a basis $B_i = \{\boldsymbol{b}_{i\ell}\}$ for its orthogonal complement $S_i^\perp$:*

$$DP_n : \ \boldsymbol{x} \in S_i \quad \mapsto \quad B_i \subset S_i^\perp. \tag{3.76}$$

*Each vector $\boldsymbol{b} \in B_i$ 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 $\boldsymbol{X} = \{\boldsymbol{x}\}$, we obtain a set of co-samples $\boldsymbol{B} = \{\boldsymbol{b}\}$ for the union of all the complement subspaces $\cup S_i^\perp$. Obviously, identifying $S_i^\perp$ from $\boldsymbol{B}$ is exactly a GPCA problem that is* dual *to the original problem. If we apply again the PDA algorithm to $\boldsymbol{B}$, then the output of the algorithm will be exactly the bases for the subspaces $(S_i^\perp)^\perp = S_i$.*[18]

[18] 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 [15], Chapter 16, their derivatives are exactly tangent vectors on these subspaces.

**Remark 27 (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 $\boldsymbol{X}$ clearly does not have this property.*[19]

*However, thanks to the duality equation (3.76), one can compute a basis for $S_i^\perp$ at every point $\boldsymbol{x}_i$ in $S_i$. A distance function between a point $\boldsymbol{x}_i$ in $S_i$ and $\boldsymbol{x}_j$ in $S_j$ can be defined between the two bases:*

$$\mathcal{D}_{ij} = \langle S_i^\perp, S_j^\perp \rangle, \tag{3.77}$$

*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., $\mathcal{S}_{ij} = \exp(-\mathcal{D}_{ij}^2)$, for the $N$ samples in $\boldsymbol{X}$. 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.*

### 3.5.2 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 $\{\boldsymbol{y}_i \in S_i\}_{i=1}^n$. 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 $\{p_{n\ell}(\boldsymbol{x})\}_{\ell=1}^m$ be the set of $m$ polynomials whose coefficients are in the null space of the data matrix $L_n$. Also, let $\tilde{\boldsymbol{x}}$ be the projection of a point

[19] 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.

$\boldsymbol{x} \in \mathbb{R}^K$ onto its closest subspace. From Lemma 3 we have that the Euclidean distance from point $\boldsymbol{x}$ to its closest subspace is given by

$$\|\boldsymbol{x} - \tilde{\boldsymbol{x}}\| = n\sqrt{P_n(\boldsymbol{x})\big(DP_n(\boldsymbol{x})^T DP_n(\boldsymbol{x})\big)^\dagger P_n(\boldsymbol{x})^T} + O\big(\|\boldsymbol{x} - \tilde{\boldsymbol{x}}\|^2\big), \tag{3.78}$$

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

$$\boldsymbol{y}_n = \underset{\boldsymbol{x}\in\boldsymbol{X}: DP_n(\boldsymbol{x})\neq 0}{\arg\min} P_n(\boldsymbol{x})\big(DP_n(\boldsymbol{x})^T DP_n(\boldsymbol{x})\big)^\dagger P_n(\boldsymbol{x})^T, \tag{3.79}$$

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

In order to find a point $\boldsymbol{y}_{n-1}$ in one of the remaining $(n-1)$ subspaces, but not in $S_n$, we need to find a new set of polynomials $\{p_{(n-1)\ell}(\boldsymbol{x})\}$ defining the algebraic set $\cup_{i=1}^{n-1} S_i$. In the case of hyperplanes this was done by polynomial division, which is equivalent to solving for a vector $\boldsymbol{c}_{n-1} \in \mathbb{R}^{M_{n-1}}$ from a linear system of the form $\mathcal{D}_n(\boldsymbol{b}_n)\boldsymbol{c}_{n-1} = \boldsymbol{c}_n$, where $\boldsymbol{b}_n \in S_I^\perp$ (see Remark 19). In the case of subspaces of arbitrary dimensions, we cannot simply divide $p_{n\ell}(\boldsymbol{x})$ by $\boldsymbol{b}_n^T\boldsymbol{x}$ for $\boldsymbol{b}_n \in S_i^\perp$, because the polynomials $\{p_{n\ell}(\boldsymbol{x})\}$ may not be factorable. Furthermore, they do not necessarily have $\boldsymbol{b}_n^T\boldsymbol{x}$ 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 6 (Polynomial division)** *For the GPCA problem, if the given sample set $\boldsymbol{X}$ 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 $\{\boldsymbol{c}_{n-1}^T\nu_{n-1}(\boldsymbol{x})\}$ for all $\boldsymbol{c}_{n-1} \in \mathbb{R}^{M_{n-1}}$ that satisfy*

$$L_n\mathcal{D}_n(\boldsymbol{b}_n)\boldsymbol{c}_{n-1} = 0, \tag{3.80}$$

*where $\boldsymbol{b}_n$ can be any vector in $S_n^\perp$.*

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

$$L_n\mathcal{D}_n(\boldsymbol{b}_n)\boldsymbol{c}_{n-1} = 0. \tag{3.81}$$

We now show the sufficiency. That is, if $\boldsymbol{c}_{n-1}$ is a solution to (3.80), then $\boldsymbol{c}_n = \mathcal{D}_n(\boldsymbol{b}_n)\boldsymbol{c}_{n-1}$ is in the null space of $L_n$. From the construction of $\mathcal{D}_n$, we have $\boldsymbol{c}_n^T \nu_n(\boldsymbol{x}) = (\boldsymbol{c}_{n-1}^T \nu_{n-1}(\boldsymbol{x}))(\boldsymbol{b}_n^T \boldsymbol{x})$. Then for every $\boldsymbol{x} \in \cup_{i=1}^{n-1} S_i$ not in $S_n$, we have $\boldsymbol{c}_{n-1}^T \nu_{n-1}(\boldsymbol{x}) = 0$, because $\boldsymbol{b}_n^T \boldsymbol{x} \neq 0$. Therefore, $\boldsymbol{c}_{n-1}^T \nu_{n-1}(\boldsymbol{x})$ is a homogeneous polynomial of degree $(n-1)$ that vanishes on $\cup_{i=1}^{n-1} S_i$. ∎

Thus a collection of polynomials $\{p_{(n-1)\ell}(\boldsymbol{x})\}$ for $\cup_{i=1}^{n-1} S_i$ can be obtained from the null space of $L_n \mathcal{D}_n(\boldsymbol{b}_n) \in \mathbb{R}^{N \times M_{n-1}}$. 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.

**Algorithm 7 (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** $L_i \boldsymbol{c} = 0$ to obtain a basis $\{\boldsymbol{c}_{i\ell}\}_{\ell=1}^{r_i}$ of *null*$(L_i)$;

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

**do**

$$\boldsymbol{y}_i = \underset{\boldsymbol{x} \in \boldsymbol{X} : DP_i(\boldsymbol{x}) \neq 0}{\arg\min} P_i(\boldsymbol{x}) \big(DP_i(\boldsymbol{x})^T DP_i(\boldsymbol{x})\big)^{\dagger} P_i(\boldsymbol{x})^T, \tag{3.82}$$

$$B_i = PCA\big(DP_i(\boldsymbol{y}_i)\big), \tag{3.83}$$

$$L_{i-1} = L_i \mathcal{D}_i(\boldsymbol{b}_i), \qquad \text{with } \boldsymbol{b}_i \text{ the first column of } B_i, \tag{3.84}$$

**end;**

**end;**

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

In the case in which all the subspaces are hyperplanes, Algorithm 7 reduces exactly to Algorithm 4.

**Remark 28 (Avoiding polynomial division)** *Similarly to the case of hyperplanes (see Remark 6), one may avoid computing $P_i$ by choosing the points $\boldsymbol{y}_i$ with a heuristic function. Since a point in $\cup_{\ell=i}^{n} S_\ell$ must satisfy $\|B_i^T \boldsymbol{x}\| \cdots \|B_n^T \boldsymbol{x}\| = 0$, we can choose a point $\boldsymbol{y}_{i-1}$ on $\cup_{\ell=1}^{i-1} S_\ell$ as:*

$$\boldsymbol{y}_{i-1} = \underset{\boldsymbol{x} \in \boldsymbol{X} : DP_n(\boldsymbol{x}) \neq 0}{arg\,\min} \frac{\sqrt{P_n(\boldsymbol{x})(DP_n(\boldsymbol{x})^T DP_n(\boldsymbol{x}))^{\dagger} P_n(\boldsymbol{x})^T} + \delta}{\|B_i^T \boldsymbol{x}\| \cdots \|B_n^T \boldsymbol{x}\| + \delta}, \tag{3.85}$$

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

## 3.6 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 $B_i = [\boldsymbol{b}_{i1}, \ldots, \boldsymbol{b}_{i(K-k_i)}]$ of $S_i^{\perp}$, where $i = 1, \ldots, n$, from a set of nonlinear equations of the form (see equation (3.4))

$$p_{n\sigma}(\boldsymbol{x}^j) = \prod_{i=1}^{n} (\boldsymbol{b}_{i\sigma(i)}^T \boldsymbol{x}^j) = 0 \qquad \text{for} \quad j = 1, \ldots, N, \tag{3.86}$$

with $\sigma$ 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 14). 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

$$E_A(B_1, \ldots, B_n) = \sum_{j=1}^{N} \prod_{i=1}^{n} \|B_i^T \boldsymbol{x}^j\|^2, \tag{3.87}$$

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 (3.87) 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 (3.87), 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

$$\sum_{j=1}^{N} \|\tilde{\boldsymbol{x}}^j - \boldsymbol{x}^j\|^2 \tag{3.88}$$

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

$$p_n(\tilde{\boldsymbol{x}}^j) = \prod_{i=1}^{n} \|B_i^T \tilde{\boldsymbol{x}}^j\| = 0. \tag{3.89}$$

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

$$\begin{array}{lll} \min & \sum_{j=1}^N \|\tilde{\boldsymbol{x}}^j - \boldsymbol{x}^j\|^2 & \\ \text{subject to} & \prod_{i=1}^n \|B_i^T \tilde{\boldsymbol{x}}^j\| = 0 & j = 1, \ldots, N \\ & B_i^T B_i = I \in \mathbb{R}^{(K-k_i)\times(K-k_i)} & i = 1, \ldots, n, \end{array} \tag{3.90}$$

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

By using Lagrangian multipliers $\lambda^j$ for each constraint on $\tilde{\boldsymbol{x}}^j$ 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

$$\sum_{j=1}^{N} \|\tilde{\boldsymbol{x}}^j - \boldsymbol{x}^j\|^2 + \sum_{j=1}^{N} \lambda^j \prod_{i=1}^{n} \|B_i^T \tilde{\boldsymbol{x}}^j\| + \sum_{i=1}^{n} \text{trace}(\Lambda_i (I - B_i^T B_i)). \tag{3.91}$$

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

$$2(\tilde{\boldsymbol{x}}^j - \boldsymbol{x}^j) + \lambda^j D p_n(\tilde{\boldsymbol{x}}^j) = 0. \tag{3.92}$$

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

$$\lambda^j = 2\frac{Dp_n(\tilde{\boldsymbol{x}}^j)^T \boldsymbol{x}^j}{\|Dp_n(\tilde{\boldsymbol{x}}^j)\|^2} \tag{3.93}$$

$$\|\tilde{\boldsymbol{x}}^j - \boldsymbol{x}^j\|^2 = \frac{1}{2} \boldsymbol{x}^{jT} Dp_n(\tilde{\boldsymbol{x}}^j) \lambda^j, \tag{3.94}$$

where we have used the fact that

$$\tilde{\boldsymbol{x}}^{jT} Dp_n(\tilde{\boldsymbol{x}}^j) = \tilde{\boldsymbol{x}}^{jT} \sum_{i=1}^{n} \prod_{\ell \neq i} \frac{\|B_\ell^T \tilde{\boldsymbol{x}}^j\|}{\|B_i^T \tilde{\boldsymbol{x}}^j\|} B_i B_i^T \tilde{\boldsymbol{x}}^j = n \prod_{i=1}^{n} \|B_i^T \tilde{\boldsymbol{x}}^j\| = np_n(\tilde{\boldsymbol{x}}^j) = 0. \tag{3.95}$$

After substituting (3.93) and (3.94) on the objective function (3.88) we obtain

$$\tilde{E}_O(\{\tilde{\boldsymbol{x}}^j\}, \{B_i\}) = \sum_{j=1}^{N} \frac{\left(\boldsymbol{x}^{jT} Dp_n(\tilde{\boldsymbol{x}}^j)\right)^2}{\|Dp_n(\tilde{\boldsymbol{x}}^j)\|^2}. \tag{3.96}$$

We can obtain an objective function on the bases only by considering first order statistics of $p_n(\boldsymbol{x}^j)$. Since this is equivalent to setting $\tilde{\boldsymbol{x}}^j = \boldsymbol{x}^j$ in (3.96) and $\boldsymbol{x}^{jT} Dp_n(\boldsymbol{x}^j) = np_n(\boldsymbol{x}^j)$, we obtain the simplified objective function

$$E_O(B_1, \ldots, B_n) = \sum_{j=1}^{N} \frac{\left(np_n(\boldsymbol{x}^j)\right)^2}{\|Dp_n(\boldsymbol{x}^j)\|^2} = \sum_{j=1}^{N} \frac{n^2 \prod_{i=1}^{n} \|B_i^T \boldsymbol{x}^j\|^2}{\left\| \sum_{i=1}^{n} \prod_{\ell \neq i} \frac{\|B_\ell^T \boldsymbol{x}^j\|}{\|B_i^T \boldsymbol{x}^j\|} B_i B_i^T \boldsymbol{x}^j \right\|^2}, \tag{3.97}$$

which is essentially the same as the algebraic error (3.87), 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 7, or any of the other GPCA algorithms depending on the case.

**Remark 29 (Optimal error in the case of hyperplanes)** *In the particular case of data lying on hyperplanes, we have that $B_i = \boldsymbol{b}_i \in \mathbb{R}^K$ for $i = 1, \ldots, n$. Therefore, the objective function in (3.97) becomes*

$$E_O(\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n) = \sum_{j=1}^{N} \frac{\left(np_n(\boldsymbol{x}^j)\right)^2}{\|Dp_n(\boldsymbol{x}^j)\|^2} = \sum_{j=1}^{N} \frac{n^2 \prod_{i=1}^{n} (\boldsymbol{b}_i^T \boldsymbol{x}^j)^2}{\left\| \sum_{i=1}^{n} \prod_{\ell \neq i} (\boldsymbol{b}_\ell^T \boldsymbol{x}^j) \boldsymbol{b}_i \right\|^2}, \tag{3.98}$$

*as demonstrated in [58].*

## 3.7 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).

### 3.7.1 The K-subspace algorithm

The K-subspace algorithm is an extension of the K-means algorithm described in Section 2.5.1 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

$$\sum_{i=1}^{n}\sum_{j=1}^{N} w_{ij}\|B_i^T\boldsymbol{x}^j\|^2 = \sum_{i=1}^{n}\sum_{j=1}^{N} w_{ij}\text{trace}(B_i^T\boldsymbol{x}^j\boldsymbol{x}^{jT}B_i) = \sum_{i=1}^{n}\text{trace}(B_i^T\Sigma_i B_i) \tag{3.99}$$

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 (3.99) 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

$$w_{ij} = \begin{cases} 1 & i = \arg\min_{\ell=1,\ldots,n} \|B_\ell^T\boldsymbol{x}^j\|^2 \\ 0 & \text{otherwise} \end{cases}. \tag{3.100}$$

In the second step, K-subspace minimizes (3.99) 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

$$\sum_{i=1}^{n}\text{trace}(B_i^T\Sigma_i B_i) + \sum_{i=1}^{n}\text{trace}(\Lambda_i(I - B_i^TB_i)). \tag{3.101}$$

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

$$\Sigma_i B_i = B_i\Lambda_i. \tag{3.102}$$

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 (3.99) we obtain

$$\sum_{i=1}^{n}\sum_{j=1}^{N} w_{ij}\|B_i^T\boldsymbol{x}^j\|^2 = \sum_{i=1}^{n}\text{trace}(\Lambda_i). \tag{3.103}$$

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

### 3.7.2 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 $\{0 \leq \pi_i \leq 1\}_{i=1}^n$, $\sum_{i=1}^n \pi_i = 1$, 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

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

**M-step: Maximizing the expected log-likelihood.** The Lagrangian for $\pi_i$ 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 $\Sigma_i$ associated with the $(K - k_i)$ smallest eigenvalues. Finally, after taking derivatives of the expected log-likelihood with respect to $\sigma_i$ we obtain

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

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

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

## 3.8 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 $\{\boldsymbol{b}_i\}_{i=1}^n$ and the estimates $\{\hat{\boldsymbol{b}}_i\}_{i=1}^n$ is computed as

$$\text{error} = \frac{1}{n}\sum_{i=1}^{n} \text{acos}\left(\boldsymbol{b}_i^T \hat{\boldsymbol{b}}_i\right) \text{(degrees)}. \tag{3.104}$$

**Error versus noise**

Figure 3.4 (left) and Figure 3.4 (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 $\delta$ 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.

**Error versus number of subspaces**

Figure 3.5 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 $n$.

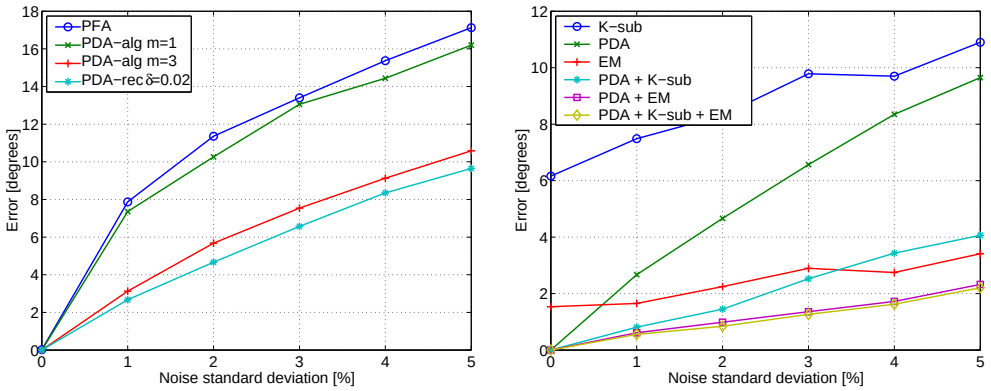



Figure 3.4: Error versus noise for data lying on $n = 4$ subspaces of $\mathbb{R}^3$ of dimension $k = 2$. Left: PFA, PDA-alg ($m = 1$ and $m = 3$) and PDA-rec ($\delta = 0.02$). Right: PDA-rec, K-subspace and EM randomly initialized, K-subspace and EM initialized with PDA-rec, and EM initialized with K-subspace initialized with PDA-rec.

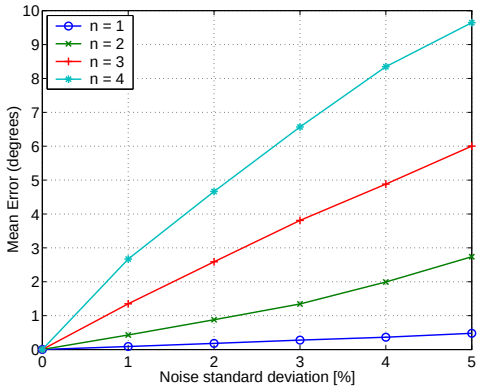



Figure 3.5: Error versus noise for PDA-rec ($\delta = 0.02$) for data lying on $n = 1, \ldots, 4$ subspaces of $\mathbb{R}^3$ of dimension $k = 2$.

### Computing time

Table 3.1 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 $p_n(\boldsymbol{x})$ given $\boldsymbol{c}_n$. The extra cost of PDA-alg and PDA-rec relative to PFA is on building the polynomial $q_n(t)$ and computing $Dp_n(\boldsymbol{x})$ for all $\boldsymbol{x} \in \boldsymbol{X}$, 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 3.1: Mean computing time and mean number of iterations for each one of the algorithms.

| Algorithm | $L_n\boldsymbol{c}=0$ | 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 |

## 3.9 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.

### 3.9.1 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 3.6 (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 3.6 (center) and compared with the ground truth. The error in the estimation of the vanishing points with respect to the ground truth are $1.7^\circ$, $11.1^\circ$ and $1.5^\circ$ for PDA and $0.4^\circ$, $2.0^\circ$, and $0.0^\circ$ for PDA+K-sub. Figure 3.6 (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.

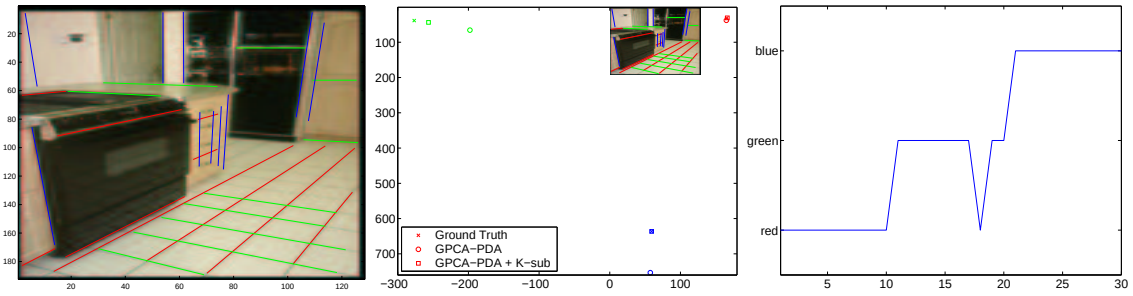



Figure 3.6: Detecting vanishing points using GPCA. Left: Image #364081 from the Corel database with 3 sets of 10 parallel lines superimposed. Center: Comparing the vanishing points estimated by PDA and PDA followed by K-subspace with the ground truth. Right: Segmentation of the 30 lines given by PDA.

### 3.9.2 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.[20] 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*

$$\boldsymbol{y}^T \boldsymbol{u} = I_{x_1}\mathrm{u} + I_{x_2}\mathrm{v} + I_t = 0. \tag{3.105}$$

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

$$\boxed{g_n(\boldsymbol{y}) = (\boldsymbol{u}_1^T \boldsymbol{y})(\boldsymbol{u}_2^T \boldsymbol{y}) \cdots (\boldsymbol{u}_n^T \boldsymbol{y}) = \prod_{i=1}^{n} (\boldsymbol{u}_i^T \boldsymbol{y}) = \tilde{\boldsymbol{u}}^T \nu_n(\boldsymbol{y}) = 0.} \tag{3.106}$$

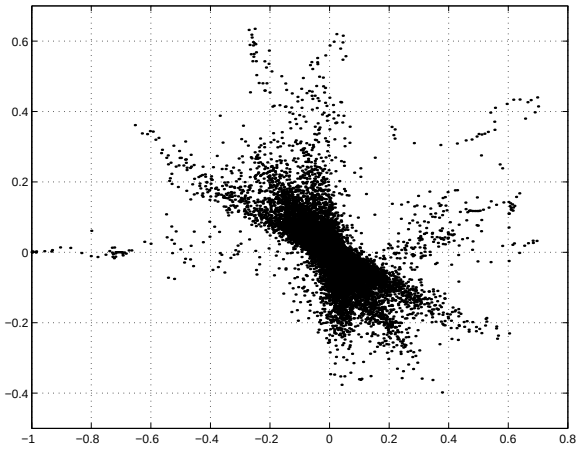
The multibody brightness constancy constraint, $g_n(\boldsymbol{y})$, is a homogeneous polynomial of degree $n$ on $\boldsymbol{y}$. We denote its vector of coefficients $\tilde{\boldsymbol{u}} \in \mathbb{R}^{M_n}$, where $M_n = (n+1)(n+2)/2$, as the *multibody optical flow* associated with the scene. Therefore, the segmentation of purely translational motion models from image intensities can be interpreted as a GPCA problem with

[20] We generalize to the affine motion model in Chapter 4.

$k = 2$ 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 3.7 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 3.7(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



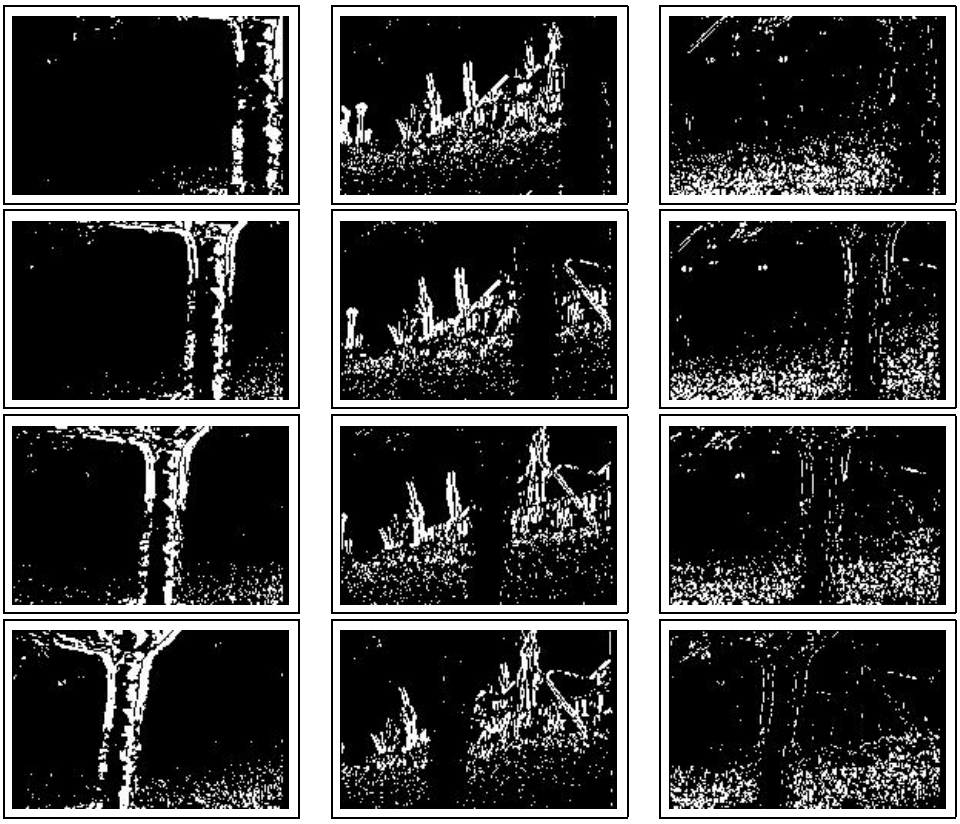

(b) Image data projected onto the $I_x$-$I_z$ plane

Figure 3.7: The flower garden sequence and its image derivatives projected onto the $I_x$-$I_z$ plane.

Figure 3.8 shows segmentation results for frames 1, 11, 21 and 31 of the flower garden sequence. This results are obtained by applying PFA (Algorithm 2) to the image data, followed by the EM algorithm for mixtures of subspaces described in Section 3.7.2. The sequence is segmented into three groups: the tree, the houses and the grass.[21] 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 will describe in Chapter 4.

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

[21] 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 3.8: Segmenting frames 1, 11, 21 and 31 of the the flower garden sequence using GPCA applied to the image derivatives.

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 [65], or a motion profile [44], from a local neighborhood would fail, since there is no local neighborhood containing a single motion model. Figure 3.9 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.

### 3.9.3 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 $k = 3$.

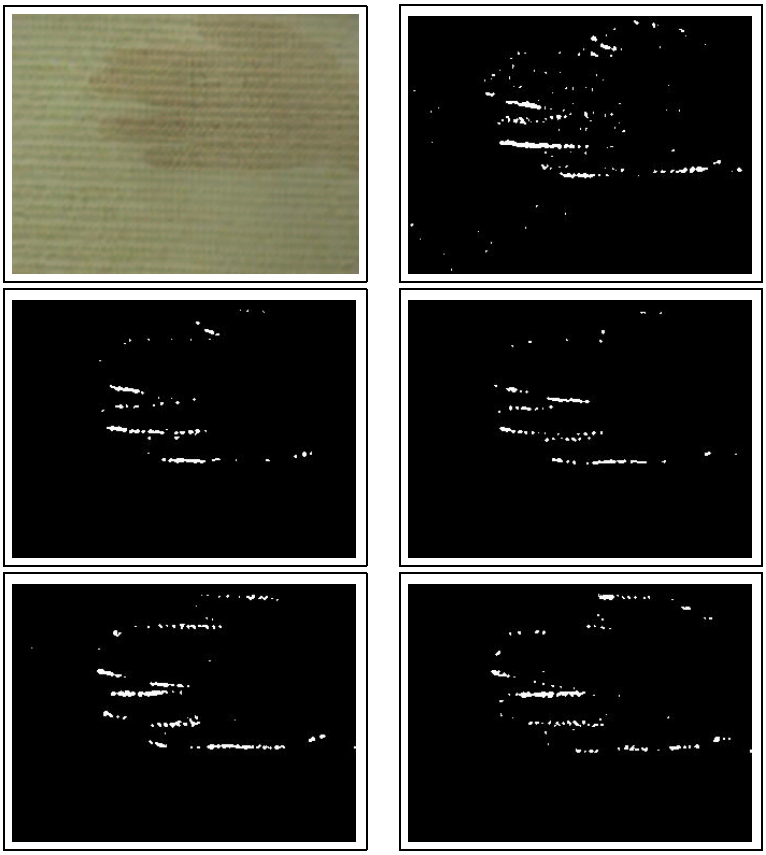


Figure 3.9: Segmenting a sequence with a hand moving behind a moving semi-transparent screen using GPCA applied to the image derivatives.

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

$$\boldsymbol{u}_j = A_i \boldsymbol{x}_j = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ 0 & 0 & 1 \end{bmatrix} \boldsymbol{x}_j. \tag{3.107}$$

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

$$a_{11}x_1 + a_{12}x_2 + a_{13} - \mathrm{u} = 0 \tag{3.108}$$

$$a_{21}x_1 + a_{22}x_2 + a_{23} - \mathrm{v} = 0. \tag{3.109}$$

Equations (3.108) and (3.109) 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 3.4.1, 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 (3.108) and (3.109) 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 3.3.

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

$$\boldsymbol{\ell}_1(j) = i \quad \text{if} \quad i = \arg\min_{i=1,\ldots,n_1} |\boldsymbol{a}_{1i}^T \boldsymbol{x}_j - \mathrm{u}_j| \quad j = 1, \ldots, N \tag{3.110}$$

$$\boldsymbol{\ell}_2(j) = i \quad \text{if} \quad i = \arg\min_{i=1,\ldots,n_2} |\boldsymbol{a}_{2i}^T \boldsymbol{x}_j - \mathrm{v}_j| \quad j = 1, \ldots, N. \tag{3.111}$$

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

$$\boxed{A_i = \begin{bmatrix} \boldsymbol{a}_{1\ell_{i1}}^T \\ \boldsymbol{a}_{2\ell_{i2}}^T \\ e_3^T \end{bmatrix} \quad i = 1, \ldots, n.} \tag{3.112}$$

We therefore have the following algorithm (Algorithm 8) for segmenting a collection of 2-D affine motion models from optical flow measurements.

**Algorithm 8 (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 (3.112).

2. **Segmentation of the image measurements.** Assign image measurement $(\boldsymbol{x}_j, \boldsymbol{u}_j)$ to the affine motion $A_i$ that minimizes $\|\boldsymbol{u}_j - A_i \boldsymbol{x}_j\|^2$.

### 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

$$\boldsymbol{x}_2^j = A_i \boldsymbol{x}_1^j = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ 0 & 0 & 1 \end{bmatrix} \boldsymbol{x}_1^j. \tag{3.113}$$

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 8 to estimate the collection of affine motion models from the given feature points.

### 3.9.4 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,[22] $\{T_i\}_{i=1}^n$, 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 3.10. Therefore $\boldsymbol{x}_1$, $\boldsymbol{x}_2$ and $T_i$ must satisfy the well-known epipolar constraint for linear motions

$$\boldsymbol{x}_2^T (T_i \times \boldsymbol{x}_1) = 0. \tag{3.114}$$

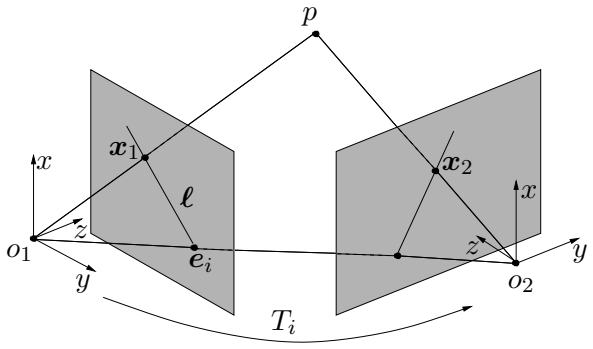



Figure 3.10: Epipolar geometry: Two projections $\boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathbb{R}^3$ of a 3-D point $p$ from two vantage points. The relative Euclidean transformation between the two vantage points is given by $T_i \in \mathbb{R}^3$. The intersection of the line $(o_1, o_2)$ with each 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

$$\boldsymbol{e}_i^T (\boldsymbol{x}_2 \times \boldsymbol{x}_1) = 0, \tag{3.115}$$

where $\boldsymbol{e}_i \in \mathbb{R}^3$ represents the epipole associated with the $i^{th}$ motion, $i = 1, \ldots, n$, if we define the vector $\boldsymbol{\ell} = (\boldsymbol{x}_2 \times \boldsymbol{x}_1) \in \mathbb{R}^3$ as a data point, then we have that $\boldsymbol{e}_i^T \boldsymbol{\ell} = 0$. Therefore, given any image pair $(\boldsymbol{x}_1, \boldsymbol{x}_2)$ corresponding to one of the $n$ moving objects, the vector $\boldsymbol{\ell} = \boldsymbol{x}_2 \times \boldsymbol{x}_1$, the so-called

[22] We will generalize to the case of arbitrary rotation and translation in Chapter 5 where we consider the problem of segmenting a mixture of fundamental matrices.

*epipolar line*, satisfies the following homogeneous polynomial of degree $n$

$$\boxed{g_n(\boldsymbol{\ell}) = (\boldsymbol{e}_1^T\boldsymbol{\ell})(\boldsymbol{e}_2^T\boldsymbol{\ell})\cdots(\boldsymbol{e}_n^T\boldsymbol{\ell}) = \prod_{i=1}^{n}(\boldsymbol{e}_i^T\boldsymbol{\ell}) = \tilde{\boldsymbol{e}}^T\nu_n(\boldsymbol{\ell}) = 0.} \tag{3.116}$$

We denote the vector of coefficients $\tilde{\boldsymbol{e}} \in \mathbb{R}^{M_n}$, 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 $\{\boldsymbol{e}_i\}_{i=1}^n$ correspond to the normal to the planes and the epipolar lines $\{\boldsymbol{\ell}^j\}_{j=1}^N$ are the data points.

Therefore, given a set of images $\{(\boldsymbol{x}_1^j, \boldsymbol{x}_2^j)\}_{j=1}^N$ of a collection of $N$ points in 3D undergoing $n$ *distinct* linear motions $\boldsymbol{e}_1, \ldots, \boldsymbol{e}_n \in \mathbb{R}^3$, one can use the set of epipolar lines $\boldsymbol{\ell}^j = \boldsymbol{x}_2^j \times \boldsymbol{x}_1^j$, where $j = 1, \ldots, N$, to estimate the number of motions $n$ and the epipoles $\boldsymbol{e}_i$ using the GPCA algorithms for hyperplanes (PFA or PDA).

Figure 3.11 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 $500 \times 500$. We run 1000 trials for each noise level. For each trial the error between the true (unit) epipoles $\{\boldsymbol{e}_i\}_{i=1}^n$ and the estimates $\{\hat{\boldsymbol{e}}_i\}_{i=1}^n$ is computed as

$$\text{error} = \frac{1}{n}\sum_{i=1}^{n} \text{acos}\left(\boldsymbol{e}_i^T\hat{\boldsymbol{e}}_i\right) \text{(degrees)}. \tag{3.117}$$

As expected, the performance deteriorates as the level of noise or the number of motion increases. The maximum error is of $12^\circ$ 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 3.12 (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 3.12 (c)), and the error in the translation estimates is $1.2^\circ$ for the truck and $3.3^\circ$ 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 3.12 (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 3.12 (e). The mean error of PDA is $5.9^\circ$ for the truck and $1.7^\circ$ for the car.

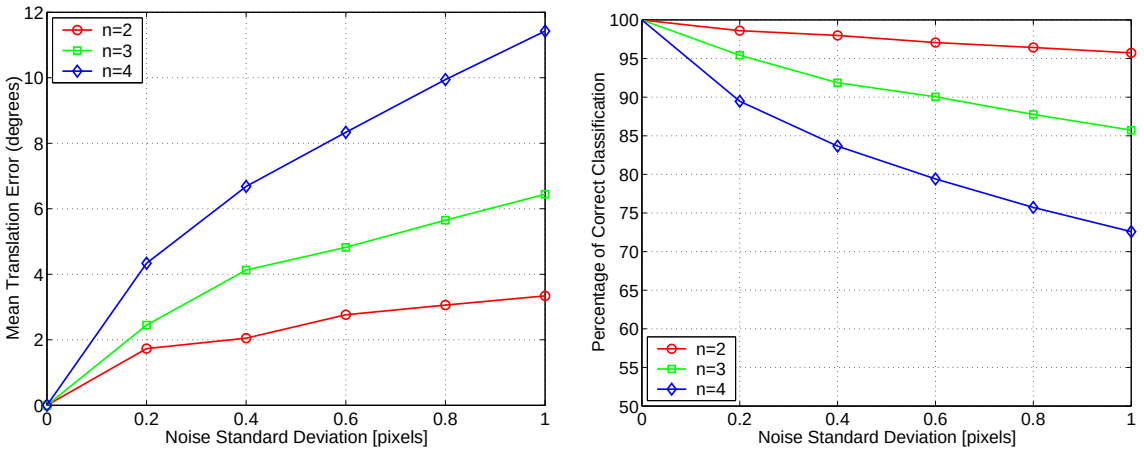
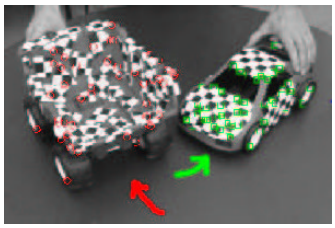
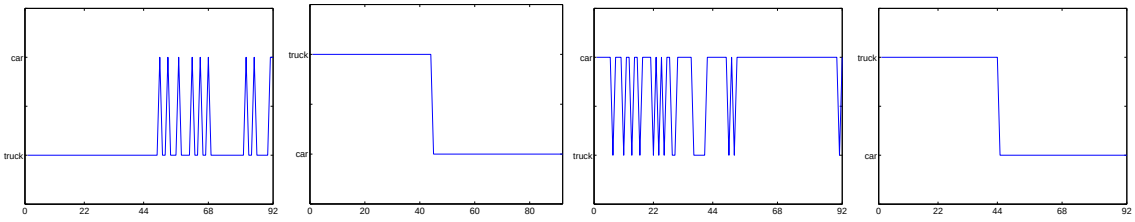



Figure 3.11: Performance of PDA on segmenting 3-D translational motions. Left: Estimation error as a function of noise for $n = 2, 3, 4$ motions. Right: Percentage of correctly classified image pairs as a function of noise for $n = 2, 3, 4$ motions.



(a) First frame




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

Figure 3.12: Segmenting 3-D translational motions using GPCA. Segmentation obtained by PFA and PDA using different changes of coordinates.

### 3.9.5 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.[23] 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.

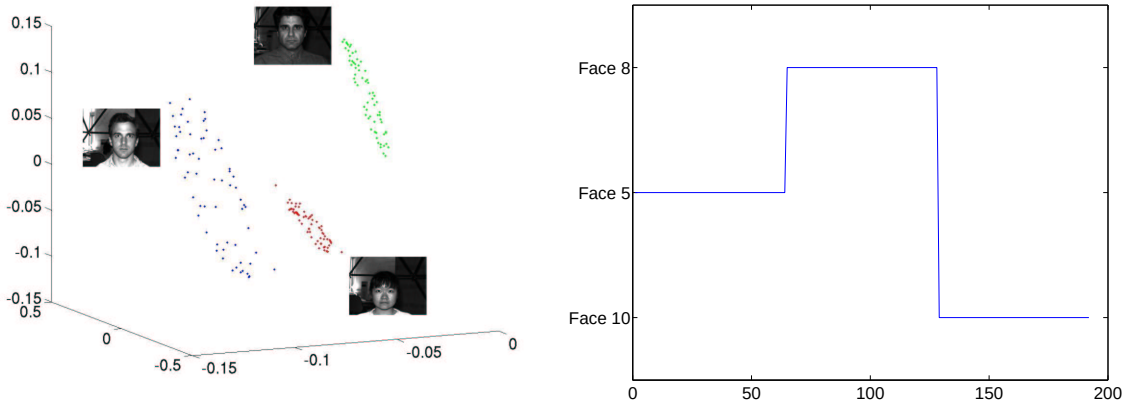



Figure 3.13: Clustering a subset of the Yale Face Database B consisting of 64 frontal views under varying lighting conditions for subjects 5, 8 and 10. Left: Image data projected onto the three principal components. Right: Clustering of the images using PDA.

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 $K = 30 \times 40$ pixels. Then we projected the data onto the first $K' = 3$ principal components, as shown in Figure 3.13 (left). We applied GPCA to this

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

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 3.13 (right) shows the segmentation results.

## 3.10 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

$$y_t = a_1 y_{t-1} + a_2 y_{t-2} + \cdots + a_{n_a} y_{t-n_a} + c_1 u_{t-1} + c_2 u_{t-2} + \cdots + c_{n_c} u_{t-n_c} \tag{3.118}$$

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 $\lambda_t$ 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 $\lambda$ is a deterministic but unknown input that is piecewise constant and finite-valued.

We consider the following identification problem:

**Problem 4 (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 4 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

$$\boldsymbol{x}_t = (u_{t-n_c}, \ldots, u_{t-1}, y_{t-n_a}, \ldots y_{t-1}, -y_t)^T \in \mathbb{R}^{n_a+n_c+1} \tag{3.119}$$

$$\boldsymbol{b} = (c_{n_c}, \cdots, c_1, a_n, \ldots a_1, 1)^T \in \mathbb{R}^{n_a+n_c+1}, \tag{3.120}$$

then we can write equation (3.118) as

$$\boldsymbol{b}^T \boldsymbol{x}_t = 0 \qquad t \geq n, \tag{3.121}$$

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 $\boldsymbol{b}$ 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 $\{\boldsymbol{b}_i\}_{i=1}^n$ 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 $S_i = \{\boldsymbol{x} : \boldsymbol{b}_i^T \boldsymbol{x} = 0\}$. In fact, when $\lambda_t$ 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 $\{\boldsymbol{x}_t\}_{t=n_a}^T$. Therefore, according to our analysis in Section 3.3, if $T - n_a + 1 \geq M_n(K) - 1$, and the evolution of the discrete state is such that the the discrete mode $i = 1, \ldots, n$ is 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 $\{\boldsymbol{b}_i\}_{i=1}^n$ uniquely using either the polynomial factorization or the polynomial differentiation algorithms. Then, given the model parameters, one can determine the discrete state as

$$\lambda_t = \arg \min_{i=1,\ldots,n} (\boldsymbol{b}_i^T \boldsymbol{x}_t)^2 \qquad \text{for} \quad t \geq n. \tag{3.122}$$

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 $\sigma$ as

$$y_t = a_1(\lambda_t)y_{t-1} + a_2(\lambda_t)y_{t-2} + c_1(\lambda_t)u_{t-1} + w_t. \tag{3.123}$$

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

$$\lambda_t = \begin{cases} 1 & 1 \leq t \leq 30 \\ 2 & 31 \leq t \leq 60 \\ 3 & 61 \leq t \leq 100 \end{cases} \tag{3.124}$$

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 3.14 shows the mean error on the estimation of the model parameters[24] and the discrete state[25], respectively, as a function of $\sigma$. 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 3.15 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.

## 3.11 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.

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.

[24] 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.

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

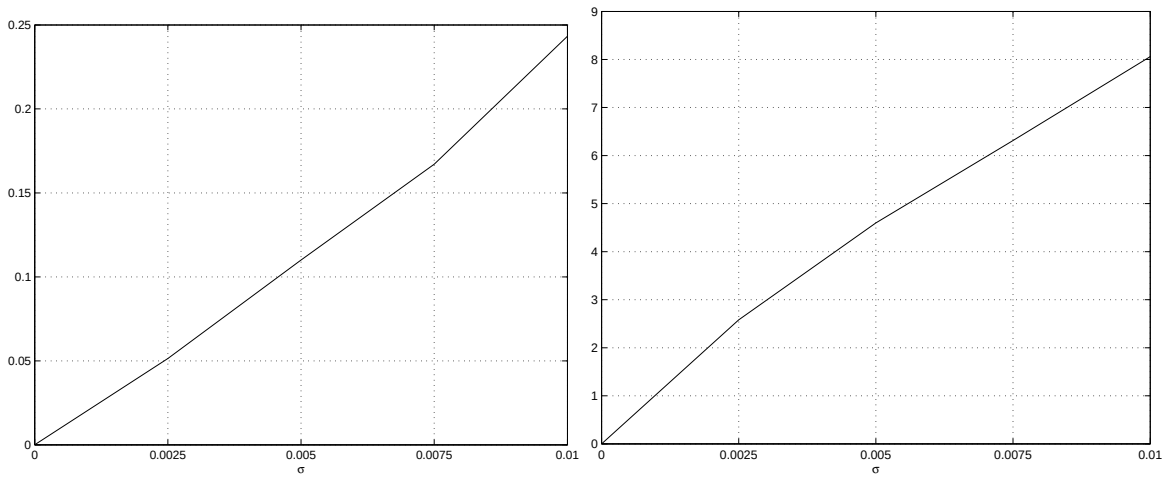



Figure 3.14: Mean error over 1000 trials for the identification of the model parameters (top) and the discrete state (bottom) as a function of the standard deviation of the measurement error $\sigma$.

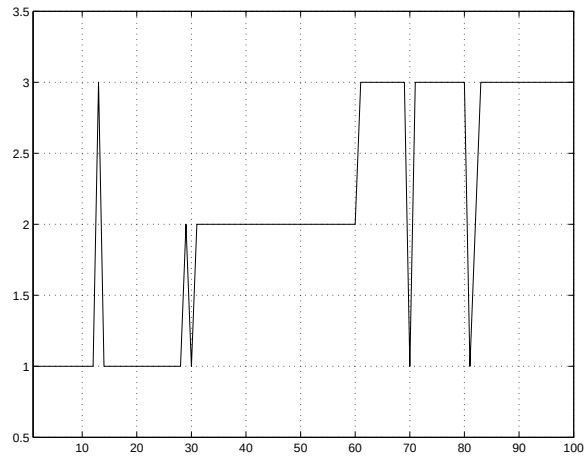



Figure 3.15: Evolution of the estimated discrete state $\hat{\lambda}_t$.

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 [32]. 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 25, it is possible to underestimate the number of subspaces even in the noise free case. Finally, throughout the chapter 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 11), and with spectral clustering techniques, e.g., the polynomial differentiation algorithm allowed us to define a similarity matrix in Remark 27. Further exploring these connections and others will be the subject of future research.

# Chapter 4

# Segmentation of 2-D Affine Motions from Image Intensities

## 4.1 Introduction

Motion estimation and segmentation refers to the problem of estimating multiple motion models from the visual information collected by a moving or static camera. This is a challenging problem in visual motion analysis, because it requires the simultaneous estimation of an unknown number of motion models, without knowing which measurements correspond to which model.

The motion estimation and segmentation problem can be divided into two main categories. 2-D motion segmentation refers to the estimation of the 2-D motion field in the image plane, i.e., the *optical flow*, while 3-D motion segmentation refers to the estimation of the 3-D motion (rotation and translation) of multiple rigidly moving objects relative to the camera. When the scene is static, i.e., when either the camera or the 3-D world undergo a single 3-D motion, one can model the 2-D motion of the scene as a mixture of 2-D motion models such as translational, affine or projective. Even though a single 3-D motion is present, multiple 2-D motion models arise because of perspective effects, depth discontinuities, occlusions, transparent motions, etc. In this case, the task of *2-D motion segmentation* is to estimate these models from the image data. When the scene is dynamic, i.e., when both the camera and multiple objects move, one can still model the scene with a mixture of 2-D motion models. Some of these models are due to independent 3-D motions, e.g., when the motion of an object relative to the camera can be well approximated by the affine motion model. Others are due to perspective effects and/or depth discontinuities, e.g., when some of the 3-D mo-

tions are broken into different 2-D motions. The task of *3-D motion segmentation* is to obtain a collection of 3-D motion models, in spite of perspective effects and/or depth discontinuities.

In this chapter, we will concentrate on the problem of segmenting 2-D affine motions from image intensities. We covered the case of 2-D translational motions from feature points or optical flow in Section 2.6.3, the case of 2-D translational motions from image intensities in Section 3.9.2, the case of 2-D affine motions from feature points or optical flow in Section 3.9.3, and the case of 3-D translational motions from feature points in Section 3.9.4. The problem of segmenting 3-D rigid motions will be covered in Chapter 5.

### 4.1.1 Previous work

Classical approaches to 2-D motion segmentation are based on separating the image flow into different regions by looking for flow discontinuities [50]. Due to the aperture problem, such techniques have trouble dealing with noisy flow estimates, especially in regions with low texture. Black and Anandan [4] deal with this problem by using some regularity constraints to interpolate the flow field. However, since the location of motion discontinuities and occlusion boundaries is unknown, these techniques often have the problem of smoothing across motion boundaries.

Alternative approaches model the scene as a mixture of 2-D parametric motion models, such as translational, affine or projective. Irani et al. [27] propose to estimate such motion models through successive computation of *dominant* motions. That is, they use all the image data to first extract *one* motion model (the dominant motion) using a least squares technique. Then, they subdivide the misaligned regions by computing the next dominant motion and so on. Although this technique can be improved by using robust M-estimators [5] and intensity information [3], it has the disadvantage of erroneously assigning data to models, especially when there is no such a dominant motion in the scene. It also fails in the presence of transparent motions.

To deal with this difficulties, Darrell and Pentland [12] proposed a new representation, the so-called *layered representation*, based on multiple motion models with different layers of support. They compute a translational model for each layer using robust M-estimation. Then they update the regions of support based on the current estimation of the motion models. The number of layers is obtained by minimizing a minimum description length (MDL)–like function. The layered representation has also been formalized as a maximum likelihood estimation problem by modeling the scene as a mixture of probabilistic motion models [28, 2, 66, 67, 55]. The estimation of the models and their regions of support is usually done using an iterative process, the so-called Expectation

Maximization (EM) algorithm, that alternates between the segmentation of the image measurements (E-step) and the estimation of the motion parameters (M-step). Jepson and Black [28] assume that the number of models is known and estimate the motion parameters using least squares. Ayer and Sawhney [2] use MDL to determine the number of models and robust M-estimation to estimate the motion parameters. Weiss [66] incorporates spatial constraints in the E-step via a mean field approximation of a Markov random field (MRF). The number of models is automatically estimated by initializing the algorithm with more models than will be needed and then decreasing the number of models whenever two models are similar. Weiss [67] and Torr et al. [55] noticed that the assumption of a parametric motion model (translational, affine or projective) is too restrictive for scenes which are non planar. [67] proposes a non-parametric mixture model based on a probability distribution that favors smooth motion fields. [55] proposes a parametric model that includes some 3-D information by associating a disparity with each pixel, similar to the plane+parallax model [26]. The model is initialized with using a Bayesian version of RANSAC.

While EM-like approaches have the advantage of providing robust motion estimates by combining information over large regions in the image, they suffer from the disadvantage that the convergence to the optimal solution strongly depends on correct initialization [44, 55]. To deal with the initialization problem, various techniques have been proposed. [65] divides the image in small patches and estimates an affine motion model for each patch using the optical flow of the patch. The parameters of the affine models are then clustered using the K-means algorithm and the regions of support of each motion model are computed by comparing the optical flow at each pixel with that generated by the "clustered" affine motion models. The drawback of this algorithm is that it is based on a local computation of optical flow which is subject to the aperture problem and to the estimation of a single affine model across motion boundaries. Some of these problems can be partially solved by incorporating multiple frames and a local process that forces the clusters to be connected [33].

Alternative approaches are based on first clustering the image data by using local features that incorporate spatial and temporal motion information. Once the segmentation of the pixels has been obtained, one can estimate a motion model for each cluster using, for example, the so-called *direct methods* [26]. Shi and Malik [44] proposed the so-called *motion profile* as a measure of the probability distribution of the image velocity at a given pixel. Such a motion profile is used to build a similarity matrix from which pixels are clustered in two groups using the normalized cuts (Ncut) algorithm. Each group is then further partitioned using recursive Ncuts. The drawback of this approach are that it is unclear when to stop subdividing the clusters and that the two-way partitioning is inappropriate in the presence of multiple motions, especially when no dominant motion is present.

### 4.1.2 Contributions

In this chapter, we propose an algebraic geometric approach to the segmentation of affine motion models from image intensities.[1] We show that one can estimate the number of affine motion models and their parameters *analytically*, without knowing the segmentation of the data. Therefore, one can combine information over large regions of the image, without having the problem of smoothing across motion boundaries or estimating one model from data coming from more than one model. In our approach *all* the image data is used at once to simultaneously recover *all* the motion models, without *ever* iterating between data segmentation and motion estimation.

In Section 4.2 we introduce the so-called *multibody affine constraint*, which is a geometric relationship between the number of models, the affine model parameters, and the image intensities. This constraint is satisfied by all the pixels in the image, regardless of the motion model associated with each pixel, and combines all the motion parameters into a single algebraic structure, the so-called *multibody affine matrix*.

In Section 4.3 we derive a rank constraint on the image measurements from which one can estimate the number of affine motions $n$. Given $n$, we show how to linearly solve for the multibody affine matrix after embedding all the image measurements into a higher-dimensional space.

In Section 4.4 we show how to recover individual affine motions from the multibody affine matrix. In principle this problem is mathematically equivalent to factoring a bi-homogeneous polynomial of degree $n$ in three variables into a product of bilinear forms. However, by exploiting the algebraic structure of the multibody affine matrix, we show that one can reduce the factorization problem to simple polynomial differentiation plus two GPCA problems in $\mathbb{R}^4$ as follows. We first show that one can compute the optical flow at each pixel in the image from the partial derivatives of the multibody affine constraint. Given the optical flow field, we show that the estimation of the affine motion models can be reduced to a collection of two GPCA problems in $\mathbb{R}^4$.

In Section 4.5 we show that, in the presence of zero-mean Gaussian noise in the image measurements, using the algebraic error defined by the multibody affine constraint as an *objective* function for motion segmentation is not optimal. Therefore, we cast the motion segmentation problem as a constrained nonlinear least squares problem which minimizes the negative log-likelihood subject to all multibody affine constraints. By converting this constrained problem into an unconstrained one, we obtain an optimal objective function that depends on the motion parameters only (the affine motions) and is independent on the segmentation of the image data.

[1] Part of the results in this chapter were published in [60].

In Section 4.6 we present experiments on synthetic data that we evaluate the performance of the proposed motion segmentation algorithm with respect to the number of motions $n$ for different levels of noise. We also present experimental results on the segmentation of an outdoor sequence.

## 4.2 Multibody affine geometry

In this section, we derive the basic equations of the affine motion segmentation problem. We introduce the *multibody affine constraint* as a geometric relationship between the number of models, the affine model parameters, and the image intensities generated by them. We also show that this constraint can be written in bilinear form by combining all the motion parameters into a single algebraic structure, the so-called *multibody affine matrix*.

### 4.2.1 The affine motion segmentation problem

We consider a static or dynamic scene whose 2-D motion field can be modeled as a mixture of an *unknown* number $n$ of different *affine motion models*. That is, we assume that the optical flow $\boldsymbol{u} = [\mathrm{u}, \mathrm{v}, 1]^T \in \mathbb{P}^2$ at pixel $\boldsymbol{x} = [x_1, x_2, 1]^T \in \mathbb{P}^2$ can be described by the equations

$$\mathrm{u}(x_1, x_2) = a_{11}x_1 + a_{12}x_2 + a_{13} \tag{4.1}$$

$$\mathrm{v}(x_1, x_2) = a_{21}x_1 + a_{22}x_2 + a_{23} \tag{4.2}$$

where $a_{11}, \ldots, a_{23}$ are the so-called *affine motion parameters*.

We also assume that the surface of each object is Lambertian, so that the optical flow of pixel $\boldsymbol{x}$ can be related to the partials of the image intensity at pixel $\boldsymbol{x}$ by the well-known *brightness constancy constraint*

$$I_{x_1}\mathrm{u} + I_{x_2}\mathrm{v} + I_t = 0. \tag{4.3}$$

Combining (4.1), (4.2) and (4.3) we obtain the *affine* constraint

$$I_{x_1}(a_{11}x_1 + a_{12}x_2 + a_{13}) + I_{x_2}(a_{21}x_1 + a_{22}x_2 + a_{23}) + I_t = 0, \tag{4.4}$$

which can be compactly written in bilinear form as

$$\boldsymbol{y}^T A \boldsymbol{x} = \begin{bmatrix} I_{x_1} & I_{x_2} & I_t \end{bmatrix} \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ 1 \end{bmatrix} = 0, \tag{4.5}$$

where $\boldsymbol{y} = [I_{x_1}, I_{x_2}, I_t]^T \in \mathbb{R}^3$ is the vector of spatial and temporal image derivatives, $A \in \mathbb{R}^{3\times 3}$ is the affine motion, and $\boldsymbol{x} \in \mathbb{P}^2$ is the vector of pixel coordinates.[2]

In the presence of $n = 1$ motion, the affine constraint $\boldsymbol{y}^T A \boldsymbol{x} = 0$ is bilinear on the image measurements $(\boldsymbol{x}, \boldsymbol{y})$ and linear on the affine motion $A$. Therefore, one can estimate $A$ linearly from a collection of $N \geq 6$ image measurements $\{(\boldsymbol{x}_j, \boldsymbol{y}_j)\}_{j=1}^N$ using equation (4.5). In the presence of $n$ *different* affine motions, $\{A_i\}_{i=1}^n$, we cannot solve the problem linearly because we do not know

1. The affine motion associated with each image measurement $(\boldsymbol{x}, \boldsymbol{y})$.
2. The number of affine motion models $n$.

Therefore, we are faced with the following problem.

**Problem 5 (Multibody affine motion segmentation)**

Given a set of image measurements $\{(\boldsymbol{x}_j, \boldsymbol{y}_j)\}_{j=1}^N$ corresponding to an unknown number of affine motions, estimate the number of motions $n$, the motion parameters $\{A_i\}_{i=1}^n$, and the segmentation of the image measurements, i.e., the motion model associated with each image measurement.

**Remark 30 (The translational motion model)** *When $a_{11} = a_{21} = a_{12} = a_{22} = 0$, the affine motion model reduces to the translational motion model $\boldsymbol{u} = [a_{13}, a_{23}, 1]^T$ that we discussed in Section 3.9.2. In this case the affine constraint $\boldsymbol{y}^T A \boldsymbol{x} = 0$ reduces to the brightness constancy constraint $\boldsymbol{y}^T \boldsymbol{u} = 0$. Therefore, the motion segmentation problem reduces to the estimation of a mixture of translational flows $\{\boldsymbol{u}_i\}_{i=1}^n$ from the image data $\{\boldsymbol{y}_j\}_{j=1}^N$. Since the brightness constancy constraint is linear (rather than bilinear) on the image measurements, the motion segmentation problem becomes a direct application of GPCA as discussed in Section 3.9.2.*

### 4.2.2 The multibody affine constraint

Let $(\boldsymbol{x}, \boldsymbol{y})$ be an image measurement associated with any motion. Then, there exists a matrix of motion parameters $A_i$ such that the affine constraint $\boldsymbol{y}^T A_i \boldsymbol{x} = 0$ holds. Therefore, regardless of the motion associated with the image measurement $(\boldsymbol{x}, \boldsymbol{y})$, the following constraint must be satisfied by the number of affine motions $n$, the motion parameters $\{A_i\}_{i=1}^n$ and the image measurement $(\boldsymbol{x}, \boldsymbol{y})$

$$\boxed{\mathcal{E}(\boldsymbol{x}, \boldsymbol{y}) \doteq \prod_{i=1}^{n} (\boldsymbol{y}^T A_i \boldsymbol{x}) = 0.} \tag{4.6}$$

[2] For simplicity, we will represent $\boldsymbol{x}$ as an homogeneous vector $\boldsymbol{x} = [x_1, x_2, x_3]^T \in \mathbb{R}^3$ from now on, unless otherwise stated.

We call this constraint the *multibody affine constraint*, since it is a natural generalization of the affine constraint valid for $n = 1$. The main difference is that the multibody affine constraint is defined for an arbitrary number of motion models, which is typically unknown. Furthermore, even if $n$ is known, the algebraic structure of the multibody affine constraint is neither bilinear in the image measurements nor linear in the affine motions, as it is in the case of $n = 1$ motion. However, we can still convert it into a bilinear constraint after embedding the data into a higher-dimensional space, as we discuss in the next subsection.

### 4.2.3 The multibody affine matrix

The multibody affine constraint converts Problem 5 into one of solving for the number of affine motions $n$ and the motion parameters $\{A_i\}_{i=1}^n$ from the *nonlinear* equation (4.6). This nonlinear constraint defines a bi-homogeneous polynomial of degree $n$ in $(\boldsymbol{x}, \boldsymbol{y})$, i.e., a homogeneous polynomial of degree $n$ in either $\boldsymbol{x}$ or $\boldsymbol{y}$. For example, if we let $\boldsymbol{x} = [x_1, x_2, x_3]^T$, then equation (4.6) viewed as a function of $\boldsymbol{x}$ can be written as a linear combination of the following monomials $\{x_1^n, x_1^{n-1}x_2, x_1^{n-1}x_3, \ldots, x_3^n\}$. It is readily seen that there are $M_n \doteq (n+1)(n+2)/2$ different monomials. Therefore, we can use the Veronese map of degree $n$, $\nu_n : \mathbb{R}^3 \to \mathbb{R}^{M_n}$, $[x_1, x_2, x_3]^T \mapsto [x_1^n, x_1^{n-1}x_2, x_1^{n-1}x_3, \ldots, x_3^n]^T$, to write the multibody affine constraint (4.6) in bilinear form as stated by the following Theorem.

**Theorem 7 (The bilinear multibody affine constraint)** *The multibody affine constraint (4.6) can be written in bilinear form as*

$$\boxed{\nu_n(\boldsymbol{y})^T \mathcal{A} \nu_n(\boldsymbol{x}) = 0,} \tag{4.7}$$

*where $\mathcal{A} \in \mathbb{R}^{M_n \times M_n}$ is a matrix representation of the symmetric tensor product of all the affine matrices $\{A_i\}_{i=1}^n$.*

*Proof.* Let $\boldsymbol{u}_i = A_i \boldsymbol{x} \in \mathbb{R}^3$, for $i = 1, \ldots, n$. Then, the multibody affine constraint

$$\mathcal{E}(\boldsymbol{x}, \boldsymbol{y}) = \prod_{i=1}^{n} (\boldsymbol{y}^T \boldsymbol{u}_i)$$

is a homogeneous polynomial of degree $n$ in $\boldsymbol{y} = [y_1, y_2, y_3]^T$, i.e.,

$$\mathcal{E}(\boldsymbol{x}, \boldsymbol{y}) = \sum c_{n_1,n_2,n_3} y_1^{n_1} y_2^{n_2} y_3^{n_3} \doteq \nu_n(\boldsymbol{y})^T \boldsymbol{c}_n,$$

where $\boldsymbol{c}_n \in \mathbb{R}^{M_n}$ is the vector of coefficients. From the properties of polynomial multiplication, each entry of $\boldsymbol{c}_n$, $c_{n_1,n_2,n_3}$, must be a symmetric multilinear function of $(\boldsymbol{u}_1, \ldots, \boldsymbol{u}_n)$, i.e., it is

linear in each $\boldsymbol{u}_i$ and $c_{n_1,n_2,n_3}(\boldsymbol{u}_1,\ldots,\boldsymbol{u}_n) = c_{n_1,n_2,n_3}(\boldsymbol{u}_{\sigma(1)},\ldots,\boldsymbol{u}_{\sigma(n)})$ for all $\sigma \in \mathfrak{S}_n$, where $\mathfrak{S}_n$ is the permutation group of $n$ elements. Since each $\boldsymbol{u}_i$ is linear in $\boldsymbol{x}$, then each $c_{n_1,n_2,n_3}$ must be a homogeneous polynomial of degree $n$ in $\boldsymbol{x}$, i.e., $c_{n_1,n_2,n_3} = \boldsymbol{a}_{n_1,n_2,n_3}^T \nu_n(\boldsymbol{x})$, where each entry of $\boldsymbol{a}_{n_1,n_2,n_3} \in \mathbb{R}^{M_n}$ is a symmetric multilinear function of the entries of the $A_i$'s. Letting

$$\mathcal{A} \doteq [\boldsymbol{a}_{n,0,0} \ , \ \boldsymbol{a}_{n-1,1,0} \ , \ \ldots \ , \ \boldsymbol{a}_{0,0,n}]^T \ \in \mathbb{R}^{M_n \times M_n},$$

we obtain

$$\mathcal{E}(\boldsymbol{x},\boldsymbol{y}) = \nu_n(\boldsymbol{y})^T \mathcal{A} \nu_n(\boldsymbol{x}) = 0.$$

■

**Remark 31 (Multibody affine tensor)** *The multibody affine matrix is a matrix representation of the symmetric tensor product of all the affine matrices*

$$\sum_{\sigma \in \mathfrak{S}_n} A_{\sigma(1)} \otimes A_{\sigma(2)} \otimes \cdots \otimes A_{\sigma(n)}, \tag{4.8}$$

*where $\mathfrak{S}_n$ is the permutation group of $n$ elements and $\otimes$ represents the tensor product.*

We call the matrix $\mathcal{A}$ the *multibody affine matrix* since it is a natural generalization of the affine matrix to the case of multiple affine motion models. Since equation (4.7) clearly resembles the bilinear form of the affine constraint for a single affine motion, we will refer to both equations (4.6) and (4.7) as the *multibody affine constraint* from now on.

**Example 8 (The two-body affine motion)** *In the case of $n = 2$ affine motions $A_1 = [b_{ij}] \in \mathbb{R}^{3\times 3}$ and $A_2 = [c_{ij}] \in \mathbb{R}^{3\times 3}$, the multibody affine motion $\mathcal{A} \in \mathbb{R}^{6\times 6}$ is given by:*

$$\mathcal{A} = \begin{bmatrix} b_{11}c_{11} & A_{12} & b_{11}c_{13}+b_{13}c_{11} & b_{12}c_{12} & b_{12}c_{13}+b_{13}c_{12} & b_{13}c_{13} \\ A_{21} & A_{22} & A_{23} & A_{24} & A_{25} & A_{26} \\ 0 & 0 & b_{11}+c_{11} & 0 & b_{12}+c_{12} & b_{13}+c_{13} \\ b_{21}c_{21} & A_{42} & b_{21}c_{23}+b_{23}c_{21} & b_{22}c_{22} & b_{22}c_{23}+b_{23}c_{22} & b_{23}c_{23} \\ 0 & 0 & b_{21}+c_{21} & 0 & b_{22}+c_{22} & b_{23}+c_{23} \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix},$$

*where*

$$\begin{aligned} A_{12} &= b_{11}c_{12}+b_{12}c_{11} & , A_{42} &= b_{21}c_{22}+b_{22}c_{21}, \\ A_{22} &= b_{11}c_{22}+b_{21}c_{12}+b_{12}c_{21}+b_{22}c_{11}, & A_{21} &= b_{11}c_{21}+b_{21}c_{11}, \\ A_{23} &= b_{11}c_{23}+b_{21}c_{13}+b_{13}c_{21}+b_{23}c_{11}, & A_{24} &= b_{12}c_{22}+b_{22}c_{12}, \\ A_{25} &= b_{12}c_{23}+b_{22}c_{13}+b_{13}c_{22}+b_{23}c_{12}, & A_{26} &= b_{13}c_{23}+b_{23}c_{13}. \end{aligned}$$

## 4.3 Estimating the number of affine motions $n$ and the multibody affine matrix $\mathcal{A}$

Notice that, by definition, the multibody affine matrix $\mathcal{A}$ depends explicitly on the number of affine motions $n$. Therefore, even though the multibody affine constraint (4.7) is *linear* in $\mathcal{A}$, we cannot use it to estimate $\mathcal{A}$ without knowing $n$ in advance. Fortunately, we can derive a rank constraint on the image measurements from which one can estimate $n$, hence $\mathcal{A}$. We rewrite the multibody affine constraint (4.7) as $(\nu_n(\boldsymbol{y}) \otimes \nu_n(\boldsymbol{x}))^T \boldsymbol{a} = 0$, where $\boldsymbol{a} \in \mathbb{R}^{M_n^2}$ is the stack of the rows of $\mathcal{A}$ and $\otimes$ represents the Kronecker product. Given a collection of image measurements $\{(\boldsymbol{x}_j, \boldsymbol{y}_j)\}_{j=1}^N$, the vector $\boldsymbol{a}$ satisfies the system of linear equations

$$L_n \boldsymbol{a} = 0, \tag{4.9}$$

where the $j^{th}$ row of $L_n \in \mathbb{R}^{N \times M_n^2}$ is $(\nu_n(\boldsymbol{y}_j) \otimes \nu_n(\boldsymbol{x}_j))^T$, for $j = 1, \ldots, N$.

In addition to equation (4.9), the matrix $\mathcal{A}$ has to satisfy other constraints due to the fact that the $3^{rd}$ row of each $A_i$ equals $e_3^T = [0, 0, 1]$. In order to determine these additional constraints, consider the polynomial $\mathcal{E}(\boldsymbol{x}, \boldsymbol{y})$ in (4.6), where $\boldsymbol{x} = [x_1, x_2, x_3]^T$ and $\boldsymbol{y} = [y_1, y_2, y_3]^T$. We observe that the monomials of $\boldsymbol{y}^T A_i \boldsymbol{x}$ involving $y_3$ must also involve $x_3$. Therefore, the coefficients of monomials in $\mathcal{E}(\boldsymbol{x}, \boldsymbol{y})$ which are multiples of $y_3^i x_3^j$ with $0 \leq j < i \leq n$ must be zero. Since the number of monomials which are multiples of $y_3^i x_3^j$ is the number of polynomials of degree $(n-i)$ in two variables ($y_1$ and $y_2$) times the number of polynomials of degree $(n-j)$ in two variables ($x_1$ and $x_2$), i.e., $(n-i+1)(n-j+1)$, the number of zeros in $\mathcal{A}$ is given by

$$Z_n = \sum_{i=1}^{n} \sum_{j=0}^{i-1} (n-i+1)(n-j+1) = \sum_{i=1}^{n} \frac{(n-i+1)(2n+3-i)i}{2} \tag{4.10}$$

$$= \frac{(n+1)(2n+3)}{2} \sum_{i=1}^{n} i - \frac{3n+4}{2} \sum_{i=1}^{n} i^2 + \frac{1}{2} \sum_{i=1}^{n} i^3 \tag{4.11}$$

$$= \frac{n(n+1)}{2} \left[ \frac{(n+1)(2n+3)}{2} - \frac{(3n+4)(2n+1)}{2} + \frac{n(n+1)}{4} \right] \tag{4.12}$$

which reduces to

$$\boxed{Z_n = \frac{n(n+1)(n+2)(3n+5)}{24}}. \tag{4.13}$$

Now, in order to obtain the entries of $\mathcal{A}$ that are zero, we proceed as follows. For each row of $\mathcal{A}$ associated to $y_3^i$, $i = 1, \ldots, n$, we look for the columns of $\mathcal{A}$ associated to $x_3^j$, for $j = 0, \ldots, i-1$.

Finally, notice that the last monomial of $\mathcal{E}(\boldsymbol{x}, \boldsymbol{y})$ is exactly $(y_3 x_3)^n$, hence the entry $(M_n, M_n)$ of $\mathcal{A}$ is one. Therefore, in order to determine $\boldsymbol{a}$ we solve the homogeneous equation

$$\boxed{\tilde{L}_n \tilde{\boldsymbol{a}} = 0,} \tag{4.14}$$

where $\tilde{\boldsymbol{a}} \in \mathbb{R}^{M_n^2 - Z_n}$ is the same as $\boldsymbol{a}$ with the zero entries removed and $\tilde{L}_n \in \mathbb{R}^{N \times (M_n^2 - Z_n)}$ is the same as $L_n$ with the columns associated to zero entries of $\boldsymbol{a}$ removed. The scale of $\boldsymbol{a}$ is obtained by enforcing the additional constraint $\boldsymbol{a}_{M_n^2} = 1$.

In order for the solution of (4.14) to be unique, we must have

$$\text{rank}(\tilde{L}_n) = M_n^2 - Z_n - 1. \tag{4.15}$$

This rank constraint on $\tilde{L}_n$ provides an effective criterion for determining the number of affine motions $n$ from the given image intensities, as stated by the following Theorem.

**Theorem 8 (Number of affine motion models)** *Let $\tilde{L}_i \in \mathbb{R}^{N \times (M_i^2 - Z_i)}$ be the matrix in (4.14), but computed with the Veronese map $\nu_i$ of degree $i \geq 1$. If rank$(A_i) \geq 2$ for all $i = 1, \ldots, n$, and a large enough set of $N$ image measurements in general configuration is given ($N \geq M_n^2 - Z_n - 1$ when $n$ is known), with at least $6$ measurements corresponding to each motion model, then*

$$\text{rank}(\tilde{L}_i) \begin{cases} > M_i^2 - Z_i - 1, & \text{if } i < n, \\ = M_i^2 - Z_i - 1, & \text{if } i = n, \\ < M_i^2 - Z_i - 1, & \text{if } i > n. \end{cases} \tag{4.16}$$

*Therefore, the number of affine motions $n$ is given by*

$$\boxed{n \doteq \min\{i : \text{rank}(\tilde{L}_i) = M_i^2 - Z_i - 1\}.} \tag{4.17}$$

*Proof.* The proof for the case of fundamental matrices can be found in Section 5.3, Theorem 11. Since the proof requires that the polynomial $\boldsymbol{y}^T A_i \boldsymbol{x}$ be irreducible, and this is indeed the case when rank$(A_i) \geq 2$, the proof is also valid for affine matrices. ∎

In summary, we can use Theorem 8 to estimate the number of affine motions $n$ incrementally from equation (4.17). Given $n$, we can linearly solve for the multibody affine motion $\mathcal{A}$ from (4.14). Notice however that the minimum number of image pixels needed is $N \geq M_n^2 - Z_n - 1$, which grows in the order of $O(n^4)$ for large $n$. Since in practice the number of motions is small, say $n \leq 10$, this is not a limitation. For example, for $n = 10$ motions we need $N \geq 2430$ pixels, which is easily satisfied by a $100 \times 100$ image.

## 4.4 Multibody affine motion estimation and segmentation

Given the multibody affine motion $\mathcal{A} \in \mathbb{R}^{M_n \times M_n}$, we now show how to compute the individual affine motions $\{A_i \in \mathbb{R}^{3\times 3}\}_{i=1}^n$. In principle, this problem is equivalent to factoring the bi-homogeneous polynomial $\mathcal{E}(\boldsymbol{x}, \boldsymbol{y})$ into $n$ bilinear expressions of the form $\boldsymbol{y}^T A_i \boldsymbol{x}$. To the best of our knowledge, this is a hard problem in real algebra and we are not aware of an efficient solution to it.[3] However, in this case we can exploit the algebraic structure of the multibody affine matrix to convert the bi-homogeneous factorization problem into a factorization of homogeneous polynomials, i.e., the GPCA problem discussed in Chapter 3, as described in the following subsections.

### 4.4.1 Estimating the optical flow field from the multibody affine motion $\mathcal{A}$

Given the multibody affine matrix $\mathcal{A} \in \mathbb{R}^{M_n \times M_n}$, we now show how to compute the optical flow at every pixel $\boldsymbol{x}$ in the image. We first notice that if the pixel $\boldsymbol{x}$ undergoes an affine motion $A_i$, then its optical flow $\boldsymbol{u}_i$ is given by $\boldsymbol{u}_i \doteq A_i \boldsymbol{x} \in \mathbb{R}^3$, $i = 1, \ldots, n$. Since

$$\nu_n(\boldsymbol{y})^T \mathcal{A} \nu_n(\boldsymbol{x}) = \prod_{i=1}^{n} \left(\boldsymbol{y}^T A_i \boldsymbol{x}\right) = \prod_{i=1}^{n} (\boldsymbol{y}^T \boldsymbol{u}_i), \tag{4.18}$$

the vector $\tilde{\boldsymbol{u}} \doteq \mathcal{A}\nu_n(\boldsymbol{x}) \in \mathbb{R}^{M_n}$ represents the coefficients of the homogeneous polynomial in $\boldsymbol{y}$

$$\boxed{g_n(\boldsymbol{y}) \doteq (\boldsymbol{y}^T \boldsymbol{u}_1)(\boldsymbol{y}^T \boldsymbol{u}_2) \cdots (\boldsymbol{y}^T \boldsymbol{u}_n) = \nu_n(\boldsymbol{y})^T \tilde{\boldsymbol{u}}.} \tag{4.19}$$

We call the vector $\tilde{\boldsymbol{u}} \doteq \mathcal{A}\nu_n(\boldsymbol{x}) \in \mathbb{R}^{M_n}$ the *multibody optical flow* associated with pixel $\boldsymbol{x}$ since it is a combination of all the optical flows $\{\boldsymbol{u}_i\}_{i=1}^n$ that pixel $\boldsymbol{x}$ can undergo depending on the affine motion associated with it[4]. From equation (4.19), we observe that recovering the optical flows $\{\boldsymbol{u}_i\}_{i=1}^n$ associated with pixel $\boldsymbol{x}$ from the multibody optical flow $\tilde{\boldsymbol{u}} = \mathcal{A}\nu_n(\boldsymbol{x})$ is equivalent to factoring the homogeneous polynomial of degree $n$, $g_n(\boldsymbol{y})$, into the $n$ homogeneous polynomials of degree one $\{\boldsymbol{y}^T \boldsymbol{u}_i\}_{i=1}^n$. This is simply a GPCA problem with $k = 2$ and $K = 3$, i.e., clustering of two-dimensional subspaces of $\mathbb{R}^3$, which can be solved using any of the GPCA algorithms discussed in Section 3.3. Recall that the GPCA problem has a unique solution (up to a scale factor for each $\boldsymbol{u}_i$) provided that $\boldsymbol{u}_1 \neq \boldsymbol{u}_2 \neq \cdots \neq \boldsymbol{u}_n$. Since in this case the vectors $\boldsymbol{u}_i$ are of the form $[\mathrm{u}, \mathrm{v}, 1]^T$, the unknown scale can actually be computed. Therefore, all the optical flows $\{\boldsymbol{u}_i\}_{i=1}^n$ associated with pixel $\boldsymbol{x}$ can be *uniquely* recovered using GPCA provided that those optical flows are different. Then, given the $n$ candidate optical flows associated with pixel $\boldsymbol{x}$, it remains to decide which one is

[3] Of course it can be solved in double exponential time using Gröebner basis and quantifier elimination.

[4] Mathematically, the multibody optical flow $\tilde{\boldsymbol{u}}$ is the symmetric tensor product of the individual optical flows $\{\boldsymbol{u}_i\}_{i=1}^n$.

*the* optical flow associated with that pixel. Since we should have $\boldsymbol{y}^T\boldsymbol{u}_i = 0$, where $\boldsymbol{y}$ is the vector of image partial derivatives at $\boldsymbol{x}$, we choose the vector $\boldsymbol{u}_i$ that minimizes $(\boldsymbol{y}^T\boldsymbol{u}_i)^2$.

**Remark 32 (Pixels with repeated flows)** *Notice that there could be pixels whose corresponding optical flows $\{\boldsymbol{u}_i\}_{i=1}^n$ are not all different from each other. This happens whenever $(A_i - A_j)\boldsymbol{x} = 0$ for some $i \neq j = 1, \ldots n$. Therefore, if we think of the image plane as a compact subset of $\mathbb{R}^2$, then the set of pixels with repeated optical flows is a zero-measure set in the image plane. In fact, it is either a collection of up to $n(n-1)/2$ points, or a collection of up to $n(n-1)/2$ lines, or a combination of both, depending on the number of solutions to the linear systems $(A_i - A_j)\boldsymbol{x} = 0$.*

Inspired by the GPCA polynomial differentiation algorithm described in Section 3.3.3, we now present a simpler and more elegant way of computing the optical flow $\boldsymbol{u}$ at every pixel $\boldsymbol{x}$ from the multibody affine matrix $\mathcal{A}$ and the vector of image partials $\boldsymbol{y}$. To this end, we notice from (4.18) that the partial derivate of the multibody affine constraint with respect to $\boldsymbol{y}$ is given by

$$\frac{\partial}{\partial \boldsymbol{y}} \nu_n(\boldsymbol{y})^T \mathcal{A} \nu_n(\boldsymbol{x}) = \sum_{i=1}^{n} \prod_{\ell \neq i} (\boldsymbol{y}^T A_\ell \boldsymbol{x})(A_i \boldsymbol{x}). \tag{4.20}$$

Therefore, if the image measurement $(\boldsymbol{x}, \boldsymbol{y})$ corresponds to motion $i$, i.e., if $\boldsymbol{y}^T A_i \boldsymbol{x} = 0$, then

$$\frac{\partial}{\partial \boldsymbol{y}} \nu_n(\boldsymbol{y})^T \mathcal{A} \nu_n(\boldsymbol{x}) = \prod_{\ell \neq i} (\boldsymbol{y}^T A_\ell \boldsymbol{x})(A_i \boldsymbol{x}) \sim A_i \boldsymbol{x}. \tag{4.21}$$

In other words, the derivative of the multibody affine constraint evaluated at $(\boldsymbol{x}, \boldsymbol{y})$ is proportional to *the* optical flow at pixel $\boldsymbol{x}$. In order to eliminate the scale factor, we notice that the third entry of $\boldsymbol{u}$ must be equal to one. Therefore, we just need to divide the derivative of the multibody affine constraint by its third coordinate. However, notice that this can be done only for pixels $\boldsymbol{x}$ whose associated optical flows $\{A_i \boldsymbol{x}\}_{i=1}^n$ are different from each other. Otherwise, the derivative of the multibody affine matrix is zero because it becomes a polynomial with one or more repeated factors.

We summarize our discussion so far with the following statement.

**Theorem 9 (Estimating the optical flow from the multibody affine matrix)** *Let $\mathcal{A} \in \mathbb{R}^{M_n \times M_n}$ be a multibody affine matrix generated by $n$ different affine motions $\{A_i\}_{i=1}^n$. Also let $\boldsymbol{y}$ be the vector of image partial derivatives at pixel $\boldsymbol{x}$ and assume that $(A_i - A_j)\boldsymbol{x} \neq 0$ for all $i \neq j = 1, \ldots, n$. Then, given $\mathcal{A}$ and $\boldsymbol{y}$ and without knowing the affine motion model associated with the image measurement $(\boldsymbol{x}, \boldsymbol{y})$, one can compute the optical flow $\boldsymbol{u}$ at pixel $\boldsymbol{x}$ as*

$$\boxed{\boldsymbol{u} = \frac{\frac{\partial}{\partial \boldsymbol{y}} \nu_n(\boldsymbol{y})^T \mathcal{A} \nu_n(\boldsymbol{x})}{e_3^T \frac{\partial}{\partial \boldsymbol{y}} \nu_n(\boldsymbol{y})^T \mathcal{A} \nu_n(\boldsymbol{x})}.} \tag{4.22}$$

### 4.4.2 Estimating individual affine motions $\{A_i\}_{i=1}^n$ from the optical flow field

Given the optical flow $\{\boldsymbol{u}_j\}_{j=1}^N$ at each pixel $\{\boldsymbol{x}_j\}_{j=1}^N$, we are now interested in computing the individual affine motions $\{A_i \in \mathbb{R}^{3\times 3}\}_{i=1}^n$. We presented a complete solution to this problem in Section 3.9.3, which we now repeat for the sake of completeness. Let us first recall that the optical flow $\boldsymbol{u} = [\mathrm{u}, \mathrm{v}, 1]^T$ at pixel $\boldsymbol{x} = [x_1, x_2, 1] \in \mathbb{P}^2$ satisfies the equations

$$a_{11}x_1 + a_{12}x_2 + a_{13} - \mathrm{u} = 0 \tag{4.23}$$

$$a_{21}x_1 + a_{22}x_2 + a_{23} - \mathrm{v} = 0. \tag{4.24}$$

Equations (4.23) and (4.24) 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 $n$ affine motion models from measurements of the optical flow $\boldsymbol{u}$ at each pixel $\boldsymbol{x}$ is then a GPCA problem with $k = 3$ and $K = 5$.

According to our discussion in Section 3.4.1, 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 (4.23) and (4.24) 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 3.3. 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. Let $\boldsymbol{\ell}_1 \in \mathbb{R}^N$ and $\boldsymbol{\ell}_2 \in \mathbb{R}^N$ be vectors of labels giving the segmentation of the data according to each projection, i.e.,

$$\boldsymbol{\ell}_1(j) = i \quad \text{if} \quad i = \arg\min_{i=1,\ldots,n_1} |\boldsymbol{a}_{1i}^T \boldsymbol{x}_j - \mathrm{u}_j| \quad j = 1, \ldots, N \tag{4.25}$$

$$\boldsymbol{\ell}_2(j) = i \quad \text{if} \quad i = \arg\min_{i=1,\ldots,n_2} |\boldsymbol{a}_{2i}^T \boldsymbol{x}_j - \mathrm{v}_j| \quad j = 1, \ldots, N. \tag{4.26}$$

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

$$\boxed{A_i = \begin{bmatrix} \boldsymbol{a}_{1\ell_{i1}} & \boldsymbol{a}_{2\ell_{i2}} & e_3 \end{bmatrix}^T \quad i = 1, \ldots, n.} \tag{4.27}$$

Once the affine motion models $\{A_i\}_{i=1}^n$ have been estimated, we can segment the image measurements by assigning pixel $\boldsymbol{x}_j$ to the affine motion $A_i$ that minimizes the algebraic error $(\boldsymbol{y}_j^T A_i \boldsymbol{x}_j)^2$. However, in the presence of noise on the image partials, we normalize the algebraic error by its variance. That is, pixel $\boldsymbol{x}_j$ is assigned to motion $A_i$ if

$$i = \arg \min_{\ell=1,\ldots,n} \frac{(\boldsymbol{y}_j^T A_\ell \boldsymbol{x}_j)^2}{\|A_\ell \boldsymbol{x}_j\|^2}. \tag{4.28}$$

We therefore have the following algorithm for segmenting a collection of affine motion models from image intensities.

**Algorithm 9 (Multibody affine motion segmentation)**

Given a collection of image measurements $\{(\boldsymbol{x}_j, \boldsymbol{y}_j)\}_{j=1}^N$ of pixels undergoing $n$ different affine motions $\{A_i\}_{i=1}^n$, 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. **Number of motions.** Compute the number of different motions $n$ from the rank constraint in (4.17), by applying the Veronese map of degree $i = 1, 2, \ldots, n$ to the image measurements.

2. **Multibody affine matrix.** Compute the multibody affine matrix $\mathcal{A} \in \mathbb{R}^{M_n \times M_n}$ by solving the linear system $\tilde{L}_n \tilde{\boldsymbol{a}} = 0$ in (4.14).

3. **Optical flow field.** Compute the optical flow $\boldsymbol{u}_j = [\mathrm{u}_j, \mathrm{v}_j, 1]^T$ at pixel $\boldsymbol{x}_j$ from the derivative of the multibody affine constraint evaluated at $(\boldsymbol{x}_j, \boldsymbol{y}_j)$, as shown in (4.22).

4. **Individual affine motions.** Estimate the individual affine motions $\{A_i\}_{i=1}^n$ as follows:

   (a) Apply a GPCA algorithm with $k = 3$ and $K = 4$ (e.g., PDA) 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 $i = 1, \ldots, n_1$.

   (b) Repeat step 4(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 $[\boldsymbol{\ell}_1 \ \boldsymbol{\ell}_2] \in \mathbb{R}^{N \times 2}$ and use them to compute the affine motions $\{A_i\}_{i=1}^n$ as in (4.27).

5. **Segmentation of the image measurements.** Assign image measurement $(\boldsymbol{x}_j, \boldsymbol{y}_j)$ to the affine motion $A_i$ that minimizes $\frac{(\boldsymbol{y}_j^T A_i \boldsymbol{x}_j)^2}{\|A_i \boldsymbol{x}_j\|^2}$.

## 4.5 Optimal segmentation of 2-D affine motions

The segmentation algorithm described in the previous section provides a unique global solution to the multibody affine motion segmentation problem. Although the problem is nonlinear, the solution is based on linear algebraic techniques thanks to the embedding of the image data into a higher-dimensional space via the Veronese map. However, such a linear solution is obtained at the cost of neglecting the algebraic structure of the multibody affine motion $\mathcal{A}$. Recall that we treat the $M_n^2 - Z_n \sim O(n^4)$ nonzero entries of $\mathcal{A}$ as unknowns, although there are only $6n$ unknowns in $\{A_i\}_{i=1}^n$. While it is algebraically correct to neglect the internal structure of $\mathcal{A}$ provided that the conditions of Theorem 8 are met, the estimation of $\mathcal{A}$ may not be robust in the presence of noisy image measurements.

One way of dealing with noisy image measurements is to set up an optimization problem that searches for the affine motions $\{A_i\}_{i=1}^n$ that minimize the algebraic error defined by the multibody affine constraint in a least squares sense, i.e.,

$$E_A(A_1, \ldots, A_n) = \sum_{j=1}^{N} \left( \prod_{i=1}^{n} (\boldsymbol{y}_j^T A_i \boldsymbol{x}_j) \right)^2. \tag{4.29}$$

However, the solution to this optimization problem may be biased, because the algebraic error in (4.29) does not coincide with the negative log-likelihood of the motion parameters.

In this section, we derive the optimal error function when the partial derivatives $\{\boldsymbol{y}_j\}_{j=1}^N$ at pixel $\{\boldsymbol{x}_j\}_{j=1}^N$ are corrupted with i.i.d. zero-mean Gaussian noise. The problem is to find a collection of affine matrices $\{A_i\}_{i=1}^n$ such that the corresponding noise free derivatives $\{\tilde{\boldsymbol{y}}_j\}_{j=1}^N$ satisfy the multibody affine constraint $\nu_n(\tilde{\boldsymbol{y}}_j)^T A \nu_n(\boldsymbol{x}_j) = 0$. This is equivalent to the constrained nonlinear least squares problem:

$$\begin{array}{ll} \min & \sum_{j=1}^N \|\tilde{\boldsymbol{y}}_j - \boldsymbol{y}_j\|^2 \\ \text{subject to} & \nu_n(\tilde{\boldsymbol{y}}_j)^T \mathcal{A} \nu_n(\boldsymbol{x}_j) = 0 \quad j = 1, \ldots, N. \end{array} \tag{4.30}$$

By using Lagrangian multipliers $\lambda_j$ for each constraint, the above optimization problem is equivalent to minimizing

$$\sum_{j=1}^{N} \|\tilde{\boldsymbol{y}}_j - \boldsymbol{y}_j\|^2 + \lambda_j \nu_n(\tilde{\boldsymbol{y}}_j)^T \mathcal{A} \nu_n(\boldsymbol{x}_j). \tag{4.31}$$

From the first order condition for optimality we get

$$2(\tilde{\boldsymbol{y}}_j - \boldsymbol{y}_j) + \lambda_j \boldsymbol{g}_j = 0, \tag{4.32}$$

where $\boldsymbol{g}_j = (D\nu_n(\tilde{\boldsymbol{y}}_j))^T A\nu_n(\boldsymbol{x}_j)$. Since $D\nu_n(\tilde{\boldsymbol{y}})\tilde{\boldsymbol{y}} = n\nu_n(\tilde{\boldsymbol{y}})$ and $\nu_n(\tilde{\boldsymbol{y}})A\nu_n(\boldsymbol{x}) = 0$, we have $\boldsymbol{g}_j^T\tilde{\boldsymbol{y}}_j = 0$. Therefore, after multiplying (4.32) on the left by $\boldsymbol{g}_j^T$, we obtain

$$-2\boldsymbol{g}_j^T\boldsymbol{y}_j + \lambda_j\|\boldsymbol{g}_j\|^2 = 0, \tag{4.33}$$

from which we can solve for $\lambda_j$ as

$$\frac{\lambda_j}{2} = \frac{\boldsymbol{g}_j^T\boldsymbol{y}_j}{\|\boldsymbol{g}_j\|^2} = \frac{\boldsymbol{y}_j^T(D\nu_n(\tilde{\boldsymbol{y}}_j))^T A\nu_n(\boldsymbol{x}_j)}{\|(D\nu_n(\tilde{\boldsymbol{y}}_j))^T A\nu_n(\boldsymbol{x}_j)\|^2}. \tag{4.34}$$

Similarly, after multiplying (4.32) on the left by $(\tilde{\boldsymbol{y}}_j - \boldsymbol{y}_j)^T$ we get

$$2\|\tilde{\boldsymbol{y}}_j - \boldsymbol{y}_j\|^2 - \lambda_j\boldsymbol{g}_j^T\boldsymbol{y}_j = 0, \tag{4.35}$$

from which the error in image derivative $j$ is given by

$$\|\tilde{\boldsymbol{y}}_j - \boldsymbol{y}_j\|^2 = \frac{\lambda_j}{2}\boldsymbol{g}_j^T\boldsymbol{y}_j = \frac{(\boldsymbol{g}_j^T\boldsymbol{y}_j)^2}{\|\boldsymbol{g}_j\|^2}. \tag{4.36}$$

After replacing $\boldsymbol{g}_j$ in the previous equation, we obtain the following expression for the total error

$$\tilde{E}_O(\{A_i\}_{i=1}^n, \{\tilde{\boldsymbol{y}}_j\}_{j=1}^N) \doteq \sum_{j=1}^{N} \frac{\left(\boldsymbol{y}_j^T(D\nu_n(\tilde{\boldsymbol{y}}_j))^T\mathcal{A}\nu_n(\boldsymbol{x}_j)\right)^2}{\|(D\nu_n(\tilde{\boldsymbol{y}}_j))^T\mathcal{A}\nu_n(\boldsymbol{x}_j)\|^2}. \tag{4.37}$$

**Remark 33** *Notice that the optimal error $\tilde{E}_O$ has a very intuitive interpretation. If measurement $j$ corresponds to motion $i$, then $\tilde{\boldsymbol{y}}_j^T A_i\boldsymbol{x}_j = 0$. This implies that*

$$\boldsymbol{g}_j = \frac{\partial}{\partial\tilde{\boldsymbol{y}}_j}\left(\nu_n(\tilde{\boldsymbol{y}}_j)^T\mathcal{A}\nu_n(\boldsymbol{x}_j)\right) = \sum_{i=1}^{n}\left(\prod_{\ell\neq i}^{n}\tilde{\boldsymbol{y}}_j^T A_\ell\boldsymbol{x}_j\right)(A_i\boldsymbol{x}_j) = \Big(\prod_{\ell\neq i}^{n}\tilde{\boldsymbol{y}}_j^T A_\ell\boldsymbol{x}_j\Big)(A_i\boldsymbol{x}_j).$$

*Therefore, the factor $\prod_{\ell\neq i}^{n}(\tilde{\boldsymbol{x}}_j^T A_\ell\boldsymbol{x}_j)$ is in both the numerator and the denominator of $\tilde{E}_O$. Hence the contribution of point $j$ to the error $\tilde{E}_O$ reduces to $\frac{(\boldsymbol{y}_j^T A_i\boldsymbol{x}_j)^2}{\|A_i\boldsymbol{x}_j\|^2}$, which is the optimal error to minimize for a single affine motion model under the Gaussian noise. Therefore, the objective function $\tilde{E}_O$ is just a clever algebraic way of simultaneously writing a mixture of optimal objective functions for individual affine matrices into a* single *objective function for* all *the affine matrices.*

We now derive an objective function the depends on the motion parameters only by considering first order statistics of $\nu_n(\boldsymbol{y}_j)^T\mathcal{A}\nu_n(\boldsymbol{x}_j)$. It turns out that this is equivalent to setting $\tilde{\boldsymbol{y}}_j = \boldsymbol{y}_j$ in the above expression for $\tilde{E}_O$. Since $D\nu_n(\boldsymbol{y})\boldsymbol{y} = n\nu_n(\boldsymbol{y})$ we get

$$\boxed{E_O(A_1, \ldots, A_n) \doteq \sum_{j=1}^{N} \frac{\left(n\nu_n(\boldsymbol{y}_j)^T\mathcal{A}\nu_n(\boldsymbol{x}_j)\right)^2}{\|(D\nu_n(\boldsymbol{y}_j))^T\mathcal{A}\nu_n(\boldsymbol{x}_j)\|^2}.} \tag{4.38}$$

The affine motion parameters $\{A_i\}_{i=1}^n$ are then recovered by minimizing the optimal error function using standard nonlinear optimization techniques. We use Algorithm 9 for initialization.

## 4.6 Experimental results

In this section, we present simulation results that we evaluate the performance of the proposed motion segmentation algorithm with respect to the number of motions $n$ for different levels of noise. We also present experimental results on intensity based motion segmentation of an outdoor sequence.

We first test the algorithm on synthetic data. We randomly pick $n = 2, 3, 4$ collections of $N = 600$ pixel coordinates and apply a different (randomly chosen) affine motion model to each collection of pixels to generate their optical flow field. From the optical flow associated with each pixel, we randomly choose a vector $\boldsymbol{y}$ of spatial and temporal image derivatives satisfying the brightness constancy constraint (4.3). Zero-mean Gaussian noise with std. from 0% to 5% is added to the partial derivatives $\boldsymbol{y}$. We run 1000 trials for each noise level. For each trial the error between the true affine motions $\{A_i\}_{i=1}^n$ and the estimates $\{\hat{A}_i\}_{i=1}^n$ is computed as

$$\text{Affine error} = \frac{1}{n}\sum_{i=1}^{n}\frac{\|A_i - \hat{A}_i\|}{\|A_i\|} \quad (\%). \tag{4.39}$$

Figure 4.1 plots the mean error as a function of the noise level. In all trials the number of motions was correctly estimated from equation (4.17) as $n = 2, 3, 4$[5]. Notice that the estimates of the algebraic algorithm are within $6\%$ of the true affine motions for $n = 2$, even for a noise level of $5\%$ in the image derivatives. However the performance deteriorates for $n = 3$. This is expected, because the algebraic algorithm uses an over-parameterized representation of the multibody affine matrix. On the other hand, the estimates of the optimal algorithm are within 2.1%, 8.2% and 13.3% of the ground truth for $n = 2, 3$ and 4, respectively. The nonlinear algorithm is less sensitive to noise, because it uses a minimal parameterization of the multibody affine motion.

We also tested the proposed approach by segmenting the real scene shown in Figure 4.2. The scene displays a leader-follower configuration with two robots moving in a circle and a static background. Even though the follower is tracking the leader, their 3D motions are not identical, yet they are similar enough to make their segmentation challenging. We computed the image partial derivatives using standard derivative filters in one dimension and smoothing filters in the other two dimensions. For computational efficiency, out of the $150 \times 200$ pixels in the image derivatives, we extracted $N = 1995$ pixels for which $|I_t| > \delta = 0.115$. Figure 4.2 shows the results of applying our algorithm to the image sequence. Notice that the two moving robots are correctly segmented.

[5] We declared the rank of $L_n$ to be $r$ if $\sigma_{r+1}/(\sigma_1 + \cdots + \sigma_r) < \epsilon$, where $\sigma_i$ is the $i$-th singular value of $L_n$ and $\epsilon = 3 \times 10^{-3}$.

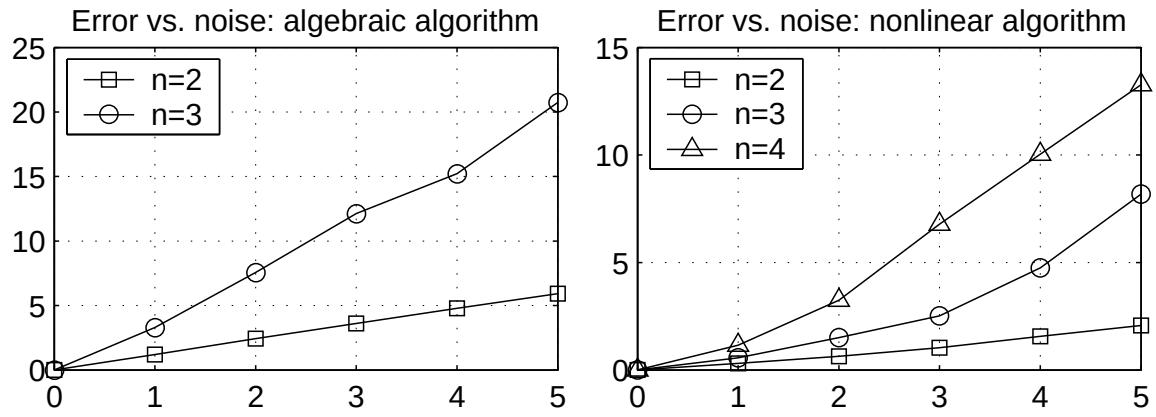
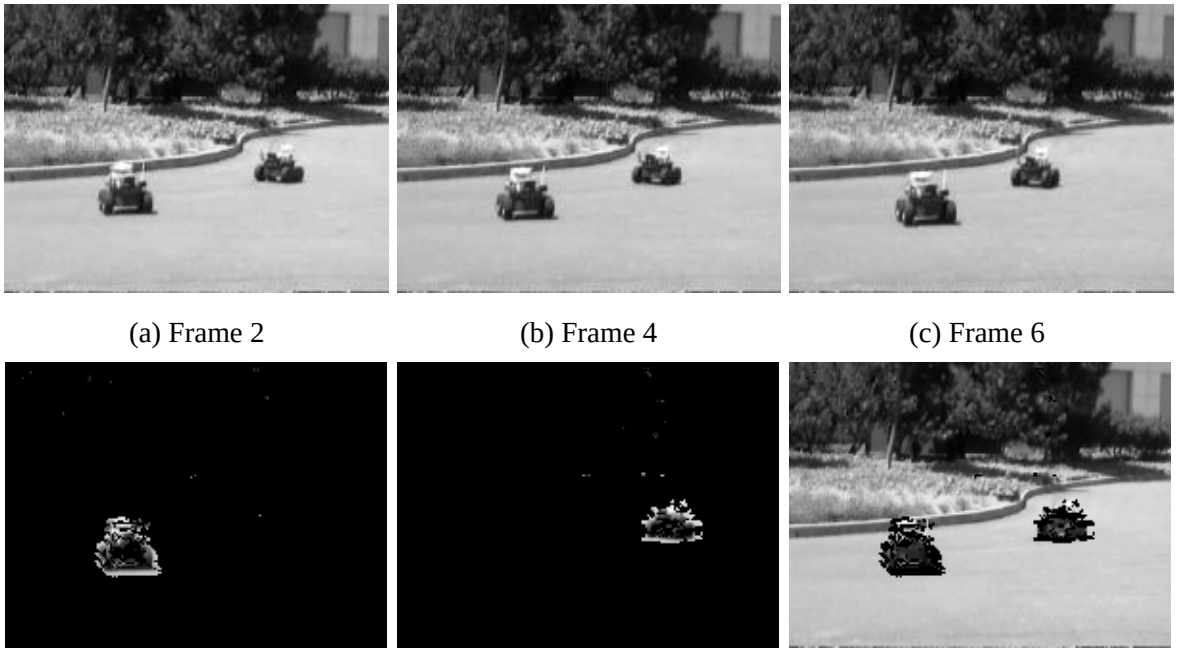



Figure 4.1: Error in the estimation of the 2-D affine motion models as a function of noise in the image partial derivatives (std. in %).



(a) Frame 2 (b) Frame 4 (c) Frame 6

(d) Robot 1 (e) Robot 2 (f) Background

Figure 4.2: Segmenting a sequence with two affine motions from image intensities.

## 4.7 Conclusions

We have proposed a novel geometric approach for segmenting 2-D affine motion models from image intensities. We showed that one can determine the number of affine motions and the motion parameters directly from image intensities with no prior segmentation or correspondences and in spite of depth discontinuities.

Our solution is based on a clear geometric interpretation the *multibody affine constraint* and its associated *multibody affine matrix*. We first showed than one can estimate the number of affine motions and the multibody affine matrix from a rank constraint on the image measurements. Given the multibody affine matrix, one can estimate the optical flow at every pixel in the image from the partial derivatives of the multibody affine constraint. Given the optical flow field, we showed that the estimation of the affine motion parameters becomes a GPCA problem. Therefore, the affine motion segmentation problem has a unique global solution that can be computed using all the image measurements, without prior segmentation.

Such a unique solution can be used to initialize probabilistic methods such as [56, 67] or any other EM-like algorithm. We used such a solution to initialize an optimal algorithm in the case of zero-mean Gaussian noise in the image partial derivatives. Instead of iterating between feature segmentation and single body motion estimation, our algorithm algebraically eliminates the feature segmentation stage and directly optimizes over the motion parameters. We presented simulation and experimental results validating the proposed approach.

# Chapter 5

# Segmentation of 3-D Rigid Motions: Multibody Structure from Motion

## 5.1 Introduction

3-D motion segmentation refers to the problem of estimating the 3-D motion (rotation and translation) of multiple rigidly moving objects from image measurements collected by a static or moving camera. This is a challenging problem in visual motion analysis, because it requires the simultaneous estimation of an unknown number of rigid motion models, without knowing which measurements correspond to which model.

Prior work on 3-D motion segmentation subdivides the problem in two stages: *feature segmentation* and *single body motion estimation*. In the first stage, image measurements corresponding to the same motion model are grouped together using algorithms such as K-means, normalized cuts, spectral clustering, etc. In the second stage, a different motion model is estimated from the image measurements corresponding to each group using, e.g., , the eight-point algorithm.

Since this two-stage approach to motion segmentation is clearly not optimal in the presence of noise, probabilistic approaches model the image data as being generated by a mixture of motion models in which the image points are corrupted with noise, typically i.i.d zero-mean and Gaussian. The membership of the data to each one of the groups is also modeled with a probability distribution, typically multinomial, by using the so-called mixing proportions. Unfortunately, the simultaneous maximum likelihood estimation of both mixture and motion parameters is in general a hard problem. Therefore, most of the existing methods solve the problem in two stages. One first

computes the expected value of the membership of the data given a prior on the motion parameters and then maximizes the expected log-likelihood of the motion parameters given a prior on the grouping of the data. This is usually done in an iterative manner using the Expectation Maximization (EM) algorithm.

One of the main reasons for using either one of these two-stage approaches (iterative or not) is that the problem of estimating a single motion model from image measurements is reasonably well understood, both from a geometric and from an optimization point of view. For example, it is well known that two views of a scene are related by the so-called epipolar constraint [35] and that multiple views are related by the so-called multilinear constraints [24]. These constraints can be naturally used to estimate the motion parameters using linear techniques, such as the eight-point algorithm and its generalizations. In the presence of noise, many optimal nonlinear algorithms for single body motion estimation have been proposed. For example, see [36, 51, 57, 69] for the discrete case and [48] for the differential case.

The geometry of multiple moving objects is, on the other hand, not as well understood. Previous work in this area includes particular cases such as multiple points moving linearly with constant speed [20, 43] or in a conic section [1], multiple moving objects seen by an orthographic camera [10, 31], self-calibration from multiple motions [17, 21], and two-object segmentation from two perspective views [70]. Alternative probabilistic approaches to 3-D motion segmentation are based on model selection techniques [56, 31], combine normalized cuts with a mixture of probabilistic models [16], or compute statistics of the innovation process of a recursive filter [49].

### 5.1.1 Contributions

In this chapter, we present a novel approach to 3-D motion segmentation that does *not* iterate between feature segmentation and single body motion estimation. Instead, our approach algebraically eliminates the feature segmentation stage and directly solves for the motion parameters, either in a purely algebraic fashion, or else using an optimal algorithm in the presence of noise. Therefore, our results provide a natural generalization of the geometric, statistical, and optimization theoretic aspects of the classical two-view structure from motion problem to the case of multiple rigidly moving objects.[1]

In Section 5.2 we show how to eliminate the feature segmentation stage by using the so-called *multibody epipolar constraint*, a geometric relationship between the motion of the objects

[1] Part of the results in this chapter were published in [62, 59, 61].

and the image data that is satisfied by all the image points, regardless of the body with which they are associated. The multibody epipolar constraint combines all the motion parameters into a single algebraic structure, the so-called *multibody fundamental matrix*, which is a natural generalization of the fundamental matrix to the case of multiple rigid motions.

Section 5.3 shows that one can estimate the number of independent motions from a rank constraint on the image data, which is obtained after embedding all the image points into a higher-dimensional space. Given the number of independent motions, we show that one can linearly solve for the multibody fundamental matrix given enough image points in general configuration.

In Section 5.4 we study the geometry of the multibody fundamental matrix. We prove that the epipoles of each independent motion lie exactly in the intersection of the left null space of the multibody fundamental matrix with the so-called Veronese surface.

In Section 5.5 we present an algebraic algorithm for simultaneous motion estimation and segmentation. We show that, given the multibody fundamental matrix, one can estimate the epipolar line associated with each image pair from the partial derivatives of the multibody epipolar constraint. Given the epipolar lines, we show that the estimation of the individual epipoles is a GPCA problem with $k = 2$ and $K = 3$, i.e., clustering a collection of two-dimensional subspaces of $\mathbb{R}^3$. Given the epipoles and the epipolar lines, the estimation of the individual fundamental matrices becomes a simple linear problem. Then, motion and feature point segmentation can be automatically obtained from either the epipoles and epipolar lines or from the individual fundamental matrices.

In Section 5.6 we consider the 3-D motion segmentation problem in the presence of zero-mean Gaussian noise in the image points. We cast the motion segmentation problem as a constrained nonlinear least squares problem which minimizes the re-projection error subject to all multibody epipolar constraints. By converting this constrained problem into an unconstrained one, we obtain an optimal objective function that depends on the motion parameters only (fundamental matrices) and is independent on the segmentation of the image data. We then use standard nonlinear optimization techniques to simultaneously recover all the fundamental matrices, without prior feature segmentation. As before, once the individual fundamental matrices have been estimated, one can trivially obtain the segmentation of the image points.

In Section 5.7 we present experimental results on 3-D motion segmentation that evaluate the performance of our algorithm with respect to the number of motions $n$ and the level of noise. We also present experimental results on the segmentation of an indoor sequence.

## 5.2 Multibody epipolar geometry

### 5.2.1 Two-view multibody structure from motion problem

Consider two images of a scene containing an *unknown* number $n$ of rigidly moving objects. We describe the rigid motion of object $i = 1, \ldots, n$ relative to the camera between the two frames with a rank-2 (nonzero) fundamental matrix $F_i \in \mathbb{R}^{3\times 3}$, $i = 1, \ldots, n$. We assume that the rigid motions of the objects relative to the camera are independent from each other, that is, we assume that the $n$ fundamental matrices are *distinct* (up to a scale factor).

The image of a point $q^j \in \mathbb{R}^3$ in any of the objects with respect to image frame $I_k$ is denoted as $\boldsymbol{x}_k^j \in \mathbb{P}^2$, for $j = 1, \ldots, N$ and $k = 1, 2$. In order to avoid degenerate cases, we will assume that the image points are in general position in 3-D space, i.e., they do not all lie in any critical surface, for example. We will drop the superscript when we refer to a generic image pair $(\boldsymbol{x}_1, \boldsymbol{x}_2)$. Also, we will use the homogeneous representation $\boldsymbol{x} = [x, y, z]^T \in \mathbb{R}^3$ to refer to an arbitrary (image) point in $\mathbb{P}^2$, unless otherwise stated.

We define the *two-view multibody structure from motion problem* as follows:

**Problem 6 (Two-view multibody structure from motion problem).**

Given a collection of image pairs $\{(\boldsymbol{x}_1^j, \boldsymbol{x}_2^j)\}_{j=1}^N$ corresponding to an unknown number of independent and rigidly moving objects, estimate the number of independent motions $n$, the fundamental matrices $\{F_i\}_{i=1}^n$, and the object to which each image pair belongs.

### 5.2.2 The multibody epipolar constraint

Given an image pair $(\boldsymbol{x}_1, \boldsymbol{x}_2)$ corresponding to the $i^{th}$ moving object, we know that the image pair and the associated fundamental matrix $F_i \in \mathbb{R}^{3\times 3}$ satisfy the so-called epipolar constraint [35]

$$\boldsymbol{x}_2^T F_i \boldsymbol{x}_1 = 0. \tag{5.1}$$

If we do not know the motion associated with the image pair $(\boldsymbol{x}_1, \boldsymbol{x}_2)$, then we know that there exists an object $i$ such that $\boldsymbol{x}_2^T F_i \boldsymbol{x}_1 = 0$. Therefore, regardless of the object to which the image pair belongs, the following constraint must be satisfied by the number of independent motions $n$, the fundamental matrices $\{F_i\}_{i=1}^n$ and the image pair $(\boldsymbol{x}_1, \boldsymbol{x}_2)$

$$\boxed{\mathcal{E}(\boldsymbol{x}_1, \boldsymbol{x}_2) \doteq \prod_{i=1}^{n} \left(\boldsymbol{x}_2^T F_i \boldsymbol{x}_1\right) = 0.} \tag{5.2}$$

We call this constraint the *multibody epipolar constraint*, since it is a natural generalization of the epipolar constraint (5.1) valid for $n = 1$. The main difference is that the multibody epipolar constraint is defined for an arbitrary number of objects, which is typically unknown (e.g., traffic surveillance). Furthermore, even if $n$ is known, the algebraic structure of the constraint is neither bilinear in the image points nor linear in the fundamental matrices as illustrated in the following example.

**Example 9 (Two rigid body motions)** *Imagine the simplest scenario of a scene containing only two independently moving objects as shown in Figure 5.1.*

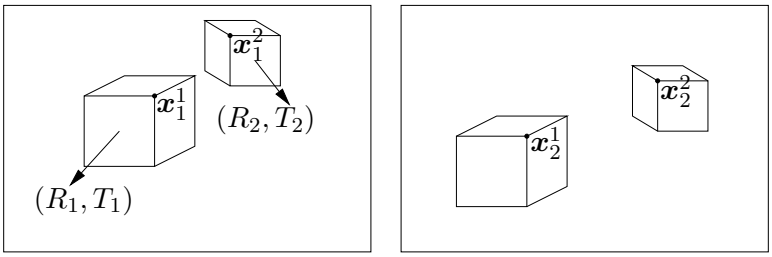



Figure 5.1: Two views of two independently moving objects, with two different rotations and translations: $(R_1, T_1)$ and $(R_2, T_2)$ relative to the camera frame.

*In this case, both image pairs $(\boldsymbol{x}_1^1, \boldsymbol{x}_2^1)$ and $(\boldsymbol{x}_1^2, \boldsymbol{x}_2^2)$ satisfy the equation*

$$\left(\boldsymbol{x}_2^T F_1 \boldsymbol{x}_1\right)\left(\boldsymbol{x}_2^T F_2 \boldsymbol{x}_1\right) = 0$$

*for $F_1 = [T_1]_\times R_1$ and $F_2 = [T_2]_\times R_2$.[2] This equation is no longer bilinear but rather bi-quadratic in the two images $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ of any point $q$ on one of these objects. Furthermore, the equation is no longer linear in $F_1$ or $F_2$ but rather bilinear in $(F_1, F_2)$. However, if enough number of image pairs $(\boldsymbol{x}_1, \boldsymbol{x}_2)$ are given, we can still recover some information about the two fundamental matrices $F_1$ and $F_2$ from such equations, in spite of the fact that we do not know the object or motion to which each image pair belongs. This special case ($n = 2$) has been studied in [70]. In this chapter, we provide a general solution for an arbitrary number of motions $n$.*

### 5.2.3 The multibody fundamental matrix

The multibody epipolar constraint allows to convert the multibody structure from motion problem (Problem 6) into one of solving for the number of independent motions $n$ and the fundamental matrices $\{F_i\}_{i=1}^n$ from the *nonlinear* equation (5.2). A standard technique used in algebra

[2] We use $[u]_\times$ to denote the $3 \times 3$ skew symmetric matrix representing the cross product with a vector $u \in \mathbb{R}^3$, i.e., we have that $[u]_\times v = u \times v$ for all $v \in \mathbb{R}^3$.

to render a nonlinear problem into a linear one is to find an "embedding" that lifts the problem into a higher-dimensional space. In this case, we notice that the multibody epipolar constraint defines a homogeneous polynomial of degree $n$ in either $\boldsymbol{x}_1$ or $\boldsymbol{x}_2$. For example, if we let $\boldsymbol{x}_1 = [x_1, y_1, z_1]^T$, then equation (5.2) viewed as a function of $\boldsymbol{x}_1$ can be written as a linear combination of the following monomials $\{x_1^n, x_1^{n-1}y_1, x_1^{n-1}z_1, \ldots, z_1^n\}$. It is readily seen that there are a total of

$$M_n \doteq \binom{n+2}{2} = \frac{(n+1)(n+2)}{2} \tag{5.3}$$

different monomials, thus the dimension of the space of homogeneous polynomials in 3 variables with real coefficients is $M_n$. Therefore, we can use the Veronese map of degree $n$, $\nu_n : \mathbb{R}^3 \to \mathbb{R}^{M_n}$, $[x_1, y_1, z_1]^T \mapsto [x_1^n, x_1^{n-1}y_1, x_1^{n-1}z_1, \ldots, z_1^n]^T$, to write the multibody epipolar constraint (5.2) as a bilinear expression in $\nu_n(\boldsymbol{x}_1)$ and $\nu_n(\boldsymbol{x}_2)$ as stated by the following theorem.

**Theorem 10 (The bilinear multibody epipolar constraint)** *The multibody epipolar constraint (5.2) can be written in bilinear form as*

$$\boxed{\nu_n(\boldsymbol{x}_2)^T \mathcal{F} \nu_n(\boldsymbol{x}_1) = 0,} \tag{5.4}$$

*where $\mathcal{F} \in \mathbb{R}^{M_n \times M_n}$ is a matrix whose entries are symmetric multilinear functions of degree $n$ on the entries of the fundamental matrices $\{F_i\}_{i=1}^n$.*

*Proof.* See proof of Theorem 7. ∎

We call the matrix $\mathcal{F}$ the *multibody fundamental matrix* since it is a natural generalization of the fundamental matrix to the case of multiple moving objects. Since equation (5.4) clearly resembles the bilinear form of the epipolar constraint for a single rigid body motion, we will refer to both equations (5.2) and (5.4) as the *multibody epipolar constraint*.

**Remark 34 (Multibody fundamental tensor)** *The multibody fundamental matrix is a matrix representation of the symmetric tensor product of all the fundamental matrices*

$$\sum_{\sigma \in \mathfrak{S}_n} F_{\sigma(1)} \otimes F_{\sigma(2)} \otimes \cdots \otimes F_{\sigma(n)}, \tag{5.5}$$

*where $\mathfrak{S}_n$ is the permutation group of $n$ elements and $\otimes$ represents the tensor product.*

Although the multibody fundamental matrix $\mathcal{F}$ seems a complicated mixture of all the individual fundamental matrices $F_1, \ldots, F_n$, it is still possible to recover all the individual fundamental matrices from $\mathcal{F}$ (alone), under some mild conditions (e.g., the $F_i$'s are distinct). The rest of the chapter is devoted to providing a constructive proof for this. Section 5.3 shows how to recover $n$ and $\mathcal{F}$ from data, and Section 5.5 shows how to recover $\{F_i\}_{i=1}^n$ from $\mathcal{F}$.

## 5.3 Estimating the number of motions $n$ and the multibody fundamental matrix $\mathcal{F}$

Notice that, by definition, the multibody fundamental matrix $\mathcal{F}$ depends explicitly on the number of independent motions $n$. Therefore, even though the multibody epipolar constraint (5.4) is *linear* in $\mathcal{F}$, we cannot use it to estimate $\mathcal{F}$ without knowing $n$ in advance. However, it turns out that one can use the multibody epipolar constraint to derive a rank constraint on the image measurements that allows to compute $n$ explicitly. Given $n$, the estimation of $\mathcal{F}$ becomes a *linear* problem.

To this end, let us first rewrite the multibody epipolar constraint (5.4) as

$$(\nu_n(\boldsymbol{x}_2) \otimes \nu_n(\boldsymbol{x}_1))^T \boldsymbol{f} = 0, \tag{5.6}$$

where $\boldsymbol{f} \in \mathbb{R}^{M_n^2}$ is the stack of the rows of $\mathcal{F}$ and $\otimes$ represents the Kronecker product. Then, given a collection of image pairs $\{(\boldsymbol{x}_1^j, \boldsymbol{x}_2^j)\}_{j=1}^N$, the vector $\boldsymbol{f}$ satisfies the system of linear equations

$$L_n \boldsymbol{f} \doteq \begin{bmatrix} \big(\nu_n(\boldsymbol{x}_2^1) \otimes \nu_n(\boldsymbol{x}_1^1)\big)^T \\ \big(\nu_n(\boldsymbol{x}_2^2) \otimes \nu_n(\boldsymbol{x}_1^2)\big)^T \\ \vdots \\ \big(\nu_n(\boldsymbol{x}_2^N) \otimes \nu_n(\boldsymbol{x}_1^N)\big)^T \end{bmatrix} \boldsymbol{f} = 0. \tag{5.7}$$

In order to determine $\boldsymbol{f}$ uniquely (up to a scale factor) from (5.7), we must have that

$$\text{rank}(L_n) = M_n^2 - 1.$$

The above rank condition on the matrix $L_n$ provides an effective criterion to determine the number of independent motions $n$ from the given image pairs, as stated by the following Theorem.

**Theorem 11 (Number of independent motions)** *Let $\{(\boldsymbol{x}_1^j, \boldsymbol{x}_2^j)\}_{j=1}^N$ be a collection of image pairs corresponding to 3-D points in general configuration and undergoing an unknown number $n$ of independent rigid body motions with nonzero translation. Let $L_i \in \mathbb{R}^{N \times M_i^2}$ be the matrix defined in (5.7), but computed using the Veronese map $\nu_i$ of degree $i \geq 1$. Then, if the number of image pairs is big enough ($N \geq M_n^2 - 1$ when $n$ is known) and at least $8$ points correspond to each motion, we have*

$$\text{rank}(L_i) \begin{cases} > M_i^2 - 1, & \text{if } i < n, \\ = M_i^2 - 1, & \text{if } i = n, \\ < M_i^2 - 1, & \text{if } i > n. \end{cases} \tag{5.8}$$

*Therefore, the number of independent motions $n$ is given by*

$$\boxed{n \doteq \min\{i : \text{rank}(L_i) = M_i^2 - 1\}.} \tag{5.9}$$

*Proof.* Since each fundamental matrix $F_i$ has rank 2, the polynomial $p_i = \boldsymbol{x}_2^T F_i \boldsymbol{x}_1$ is irreducible over the real field $\mathbb{R}$. Let $Z_i$ be the set of $(\boldsymbol{x}_1, \boldsymbol{x}_2)$ that satisfy $\boldsymbol{x}_2^T F_i \boldsymbol{x}_1 = 0$. Then due to the irreducibility of $p_i$, any polynomial $p$ in $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ that vanishes on the entire set $Z_i$ must be of the form $p = p_i h$, where $h$ is some polynomial. Hence if $F_1, \ldots, F_n$ are distinct, a polynomial which vanishes on the set $\cup_{i=1}^n Z_i$ must be of the form $p = p_1 p_2 \cdots p_n h$ for some $h$. Therefore, the only polynomial of *minimal* degree that vanishes on the same set is

$$p = p_1 p_2 \cdots p_n = \left(\boldsymbol{x}_2^T F_1 \boldsymbol{x}_1\right) \left(\boldsymbol{x}_2^T F_2 \boldsymbol{x}_1\right) \cdots \left(\boldsymbol{x}_2^T F_n \boldsymbol{x}_1\right). \tag{5.10}$$

Since the rows of $L_n$ are of the form $(\nu_n(\boldsymbol{x}_2) \otimes \nu_n(\boldsymbol{x}_1))^T$ and the entries of $\nu_n(\boldsymbol{x}_2) \otimes \nu_n(\boldsymbol{x}_1)$ are exactly the independent monomials of $p$ (as we will show below), this implies that if the number of data points per motion is at least 8 and $N \geq M_n^2 - 1$, then:

1. There is no polynomial of degree $2i < 2n$ whose coefficients are in the null space of $L_i$, i.e., $\text{rank}(L_i) = M_i^2 > M_i^2 - 1$ for $i < n$.

2. There is a unique polynomial of degree $2n$, namely $p$, whose coefficients are in the null space of $L_n$, i.e., $\text{rank}(L_n) = M_n^2 - 1$.

3. There is more than one polynomial of degree $2i > 2n$ (one for each independent choice of the $2(i-n)$-degree polynomial $h$ in $p = p_1 p_2 \cdots p_n h$) with coefficients in the null space of $L_i$, i.e., $\text{rank}(L_i) < M_i^2 - 1$ for $i > n$.

The rest of the proof is to show that the entries of $\nu_n(\boldsymbol{x}_2) \otimes \nu_n(\boldsymbol{x}_1)$ are exactly the independent monomials in the polynomial $p$, which we do by induction. Since the claim is obvious for $n = 1$, we assume that it is true for $n$ and prove it for $n+1$. Let $\boldsymbol{x}_1 = [x_1, y_1, z_1]^T$ and $\boldsymbol{x}_2 = [x_2, y_2, z_2]^T$. Then the entries of $\nu_n(\boldsymbol{x}_2) \otimes \nu_n(\boldsymbol{x}_1)$ are of the form $(x_2^{m_1} y_2^{m_2} z_2^{m_3})(x_1^{n_1} y_1^{n_2} z_1^{n_3})$ with $m_1 + m_2 + m_3 = n_1 + n_2 + n_3 = n$, while the entries of $\boldsymbol{x}_2 \otimes \boldsymbol{x}_1$ are of the form $(x_2^{i_1} y_2^{i_2} z_2^{i_3})(x_1^{j_1} y_1^{j_2} z_1^{j_3})$ with $i_1 + i_2 + i_3 = j_1 + j_2 + j_3 = 1$. Thus a basis for the product of these monomials is given by the entries of $\nu_{n+1}(\boldsymbol{x}_2) \otimes \nu_{n+1}(\boldsymbol{x}_1)$. ∎

The significance of Theorem 11 is that the number of independent motions can now be determined incrementally using equation (5.9). Once the number $n$ of motions is found, the multibody fundamental matrix $\mathcal{F}$ is simply the 1-D null space of the corresponding matrix $L_n$, which can

be linearly obtained. Nevertheless, in order for this scheme to work, the minimum number of image pairs needed is $N \geq M_n^2 - 1$. For $n = 1, 2, 3, 4$, the minimum $N$ is $8, 35, 99, 225$, respectively. If $n$ is large, $N$ grows approximately in the order of $O(n^4)$ – a price to pay for working with a linear representation of Problem 6. In Section 5.5.5 we will discuss many variations to the general scheme that will reduce the number of data points required, especially for large $n$.

## 5.4 Null space of the multibody fundamental matrix

In this section, we study the relationships between the multibody fundamental matrix $\mathcal{F}$ and the epipoles $\boldsymbol{e}_1, \ldots, \boldsymbol{e}_n$ associated with the fundamental matrices $F_1, \ldots, F_n$. The relationships between epipoles and epipolar lines will be studied in the next section, where we will show how they can be computed from the multibody fundamental matrix $\mathcal{F}$.

First of all, recall that the epipole $\boldsymbol{e}_i$ associated with the $i^{th}$ motion in the second image is defined as the left kernel of the rank-2 fundamental matrix $F_i$, that is

$$\boldsymbol{e}_i^T F_i \doteq 0. \tag{5.11}$$

Hence, the following polynomial (in $\boldsymbol{x}$) is zero for any $\boldsymbol{e}_i$, $i = 1, \ldots, n$

$$\left(\boldsymbol{e}_i^T F_1 \boldsymbol{x}\right)\left(\boldsymbol{e}_i^T F_2 \boldsymbol{x}\right) \cdots \left(\boldsymbol{e}_i^T F_n \boldsymbol{x}\right) = \nu_n(\boldsymbol{e}_i)^T \mathcal{F} \nu_n(\boldsymbol{x}) = 0. \tag{5.12}$$

We call the vector $\nu_n(\boldsymbol{e}_i)$ the *embedded epipole* associated with the $i^{th}$ motion. Since $\nu_n(\boldsymbol{x})$ as a vector spans the entire $\mathbb{R}^{M_n}$ when $\boldsymbol{x}$ ranges over $\mathbb{P}^2$ (or $\mathbb{R}^3$),[3] we have

$$\nu_n(\boldsymbol{e}_i)^T \mathcal{F} = 0. \tag{5.13}$$

Therefore, the embedded epipoles $\{\nu_n(\boldsymbol{e}_i)\}_{i=1}^n$ lie on the left null space of $\mathcal{F}$ while the epipoles $\{\boldsymbol{e}_i\}_{i=1}^n$ lie on the left null space of $\{F_i\}_{i=1}^n$. Hence, the rank of $\mathcal{F}$ is bounded depending on the number of *distinct* (pairwise linearly independent) epipoles as stated by Lemmas 5 and 6.

**Lemma 5 (Null space of $\mathcal{F}$ when the epipoles are distinct)** *Let $\mathcal{F}$ be the multibody fundamental matrix generated by the fundamental matrices $F_1, \ldots, F_n$. If the epipoles $\boldsymbol{e}_1, \ldots, \boldsymbol{e}_n$ are distinct (up to a scale factor), then the (left) null space of $\mathcal{F} \in \mathbb{R}^{M_n \times M_n}$ contains at least the $n$ linearly independent vectors*

$$\nu_n(\boldsymbol{e}_i) \in \mathbb{R}^{M_n}, \quad i = 1, \ldots, n. \tag{5.14}$$

[3] This is simply because the $M_n$ monomials in $\nu_n(\boldsymbol{x})$ are linearly independent.

*Therefore, the rank of the multibody fundamental matrix $\mathcal{F}$ is bounded by*

$$\boxed{\text{rank}(\mathcal{F}) \leq (M_n - n).} \tag{5.15}$$

*Proof.* We only need to show that if the $\boldsymbol{e}_i$'s are distinct, then the $\nu_n(\boldsymbol{e}_i)$'s are linearly independent. If we let $\boldsymbol{e}_i = [x_i, y_i, z_i]^T, i = 1, \ldots, n$, then we only need to prove the rank of the following matrix

$$U \doteq \begin{bmatrix} \nu_n(\boldsymbol{e}_1)^T \\ \nu_n(\boldsymbol{e}_2)^T \\ \vdots \\ \nu_n(\boldsymbol{e}_n)^T \end{bmatrix} = \begin{bmatrix} x_1^n & x_1^{n-1}y_1 & x_1^{n-1}z_1 & \cdots & z_1^n \\ x_2^n & x_2^{n-1}y_2 & x_2^{n-1}z_2 & \cdots & z_2^n \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_n^n & x_n^{n-1}y_n & x_n^{n-1}z_n & \cdots & z_n^n \end{bmatrix} \in \mathbb{R}^{n \times M_n} \tag{5.16}$$

is exactly $n$. Since the $\boldsymbol{e}_i$'s are distinct, we can assume without loss of generality that $\{[x_i, z_i]\}_{i=1}^n$ are already distinct and that $z_i \neq 0$.[4] Then, after dividing the $i^{th}$ row of $U$ by $z_i^n$ and letting $t_i = x_i/z_i$, we can extract the following Van Der Monde sub-matrix of $U$

$$V \doteq \begin{bmatrix} t_1^{n-1} & t_1^{n-2} & \cdots & 1 \\ t_2^{n-1} & t_2^{n-2} & \cdots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ t_n^{n-1} & t_n^{n-2} & \cdots & 1 \end{bmatrix} \in \mathbb{R}^{n \times n}. \tag{5.17}$$

Since $\det(V) = \prod_{i<j}(t_i - t_j)$, the Van Der Monde matrix $V$ has rank $n$ if and only if $t_1, \ldots, t_n$ are distinct. Hence $\text{rank}(U) = \text{rank}(V) = n$. ∎

Even though we know that the linearly independent vectors $\nu_n(\boldsymbol{e}_i)$'s lie on the left null space of $\mathcal{F}$, we do not know if the $n$-dimensional subspace spanned by them will be exactly the left null space of $\mathcal{F}$, i.e., we do not know if $\text{rank}(\mathcal{F}) = M_n - n$.

Now, if one of the epipoles is repeated, then the null space of $\mathcal{F}$ is actually enlarged by higher-order derivatives of the Veronese map as stated by the following Lemma.

**Lemma 6 (Null space of $\mathcal{F}$ when one epipole is repeated)** *Let $\mathcal{F}$ be the multibody fundamental matrix generated by the fundamental matrices $F_1, \ldots, F_n$ with epipoles $\boldsymbol{e}_1, \ldots, \boldsymbol{e}_n$. Let $\boldsymbol{e}_1$ be repeated $k$ times, i.e., $\boldsymbol{e}_1 = \cdots = \boldsymbol{e}_k$, and let the other $n-k$ epipoles be distinct. Then the rank of the multibody fundamental matrix $\mathcal{F}$ is bounded by*

$$\text{rank}(\mathcal{F}) \leq M_n - M_{k-1} - (n-k). \tag{5.18}$$

[4] This assumption is not always satisfied, e.g., for $n = 3$ motions with epipoles along the $X$, $Y$ and $Z$ axis. However, as long as the $\boldsymbol{e}_i$'s are distinct, one can always find a non-singular linear transformation $\boldsymbol{e}_i \mapsto T\boldsymbol{e}_i$ on $\mathbb{R}^3$ that makes the assumption true. Furthermore, this linear transformation induces a linear transformation on the lifted space $\mathbb{R}^{M_n}$ that preserves the rank of the matrix $U$.

*Proof.* When $k = 2$, $\boldsymbol{e}_1 = \boldsymbol{e}_2$ is a "repeated root" of $\nu_n(\boldsymbol{x})^T\mathcal{F}$ as a polynomial (matrix) in $\boldsymbol{x} = [x, y, z]^T$. Hence we have

$$\frac{\partial \nu_n(\boldsymbol{x})^T}{\partial x}\mathcal{F}\Big|_{\boldsymbol{x}=\boldsymbol{e}_1} = 0, \quad \frac{\partial \nu_n(\boldsymbol{x})^T}{\partial y}\mathcal{F}\Big|_{\boldsymbol{x}=\boldsymbol{e}_1} = 0, \quad \frac{\partial \nu_n(\boldsymbol{x})^T}{\partial z}\mathcal{F}\Big|_{\boldsymbol{x}=\boldsymbol{e}_1} = 0.$$

Notice that the Jacobian of the Veronese map $D\nu_n(\boldsymbol{x})$ is full rank for all $\boldsymbol{x} \in \mathbb{P}^2$, because for $n \geq 2$ we have $D\nu_n(\boldsymbol{x})^T D\nu_n(\boldsymbol{x}) \succeq \boldsymbol{x}^T\boldsymbol{x}\, I_{3\times 3}$. Thus, the vectors $\frac{\partial \nu_n(\boldsymbol{e}_1)}{\partial x}, \frac{\partial \nu_n(\boldsymbol{e}_1)}{\partial y}, \frac{\partial \nu_n(\boldsymbol{e}_1)}{\partial z}$ are linearly independent, because they are the columns of $D\nu_n(\boldsymbol{e}_1)$ and $\boldsymbol{e}_1 \neq 0$. In addition, their span contains $\nu_n(\boldsymbol{e}_1)$, because

$$n\nu_n(\boldsymbol{x}) = D\nu_n(\boldsymbol{x})\boldsymbol{x} \doteq \left[\frac{\partial \nu_n(\boldsymbol{x})}{\partial x}, \frac{\partial \nu_n(\boldsymbol{x})}{\partial y}, \frac{\partial \nu_n(\boldsymbol{x})}{\partial z}\right]\boldsymbol{x}, \quad \forall \boldsymbol{x} \in \mathbb{R}^3, \tag{5.19}$$

but does not containt $\nu_n(\boldsymbol{e}_i)$ for $i = 3, \ldots, n$. Hence rank$(\mathcal{F}) \leq M_n - M_1 - (n-1) = M_n - 3 - (n-1)$. Now if $k > 2$, one should consider the $(k-1)^{th}$ order partial derivatives of $\nu_n(\boldsymbol{x})$ evaluated at $\boldsymbol{e}_1$. There is a total of $M_{k-1}$ such partial derivatives, which give rise to $M_{k-1}$ linearly independent vectors in the (left) null space of $\mathcal{F}$. Similarly to the case $k = 2$, one can show that the embedded epipole $\nu_n(\boldsymbol{e}_1)$ is in the span of these higher-order partials. ∎

**Example 10 (Two repeated epipoles)** *In the two-body problem, if $F_1$ and $F_2$ have the same (left) epipole, i.e., $F_1 = [T]_\times R_1$ and $F_2 = [T]_\times R_2$, then the rank of the two-body fundamental matrix $\mathcal{F}$ is $M_2 - M_1 - (2-2) = 6 - 3 = 3$ instead of $M_2 - 2 = 4$.*

Since the null space of $\mathcal{F}$ is enlarged by higher-order derivatives of the Veronese map evaluated at repeated epipoles, in order to identify the embedded epipoles $\nu_n(\boldsymbol{e}_i)$ from the left null space of $\mathcal{F}$ we will need to exploit the algebraic structure of the Veronese map. Let us denote the image of the real projective space $\mathbb{P}^2$ under the Veronese map of degree $n$ as $\nu_n(\mathbb{P}^2)$.[5] The following theorem establishes a key relationship between the null space of $\mathcal{F}$ and the epipoles of each fundamental matrix.

**Theorem 12 (Veronese null space of multibody fundamental matrix)** *The intersection of the left null space of the multibody fundamental matrix $\mathcal{F}$,* null$(\mathcal{F})$*, with the Veronese surface $\nu_n(\mathbb{P}^2)$ is exactly*

$$\boxed{\text{null}(\mathcal{F}) \cap \nu_n(\mathbb{P}^2) = \{\nu_n(\boldsymbol{e}_i)\}_{i=1}^n.} \tag{5.20}$$

[5] This is the so-called (real) Veronese surface in Algebraic Geometry [22].

*Proof.* Let $\boldsymbol{x} \in \mathbb{P}^2$ be a vector whose Veronese map is in the left null space of $\mathcal{F}$. We then have

$$\nu_n(\boldsymbol{x})^T \mathcal{F} = 0 \quad \Leftrightarrow \quad \nu_n(\boldsymbol{x})^T \mathcal{F} \nu_n(\boldsymbol{y}) = 0, \forall \boldsymbol{y} \in \mathbb{P}^2. \tag{5.21}$$

Since $\mathcal{F}$ is a multibody fundamental matrix,

$$\nu_n(\boldsymbol{x})^T \mathcal{F} \nu_n(\boldsymbol{y}) = \prod_{i=1}^{n} \left( \boldsymbol{x}^T F_i \boldsymbol{y} \right).$$

This means for this $\boldsymbol{x}$,

$$\prod_{i=1}^{n} (\boldsymbol{x}^T F_i \boldsymbol{y}) = 0, \quad \forall \boldsymbol{y} \in \mathbb{P}^2. \tag{5.22}$$

If $\boldsymbol{x}^T F_i \neq 0$ for all $i = 1, \ldots, n$, then the set of $\boldsymbol{y}$ that satisfy the above equation is simply the union of $n$ 2-dimensional subspaces in $\mathbb{P}^2$, which will never fill the entire space $\mathbb{P}^2$. Hence we must have $\boldsymbol{x}^T F_i = 0$ for some $i$. Therefore $\boldsymbol{x}$ is one of the epipoles. ∎

The significance of Theorem 12 is that, in spite of the fact that repeated epipoles may enlarge the null space of $\mathcal{F}$, and that we do not know if the dimension of the null space equals $n$ for distinct epipoles, one may always find the epipoles exactly by intersecting the left null space of $\mathcal{F}$ with the Veronese surface $\nu_n(\mathbb{P}^2)$, as illustrated in Figure 5.2.

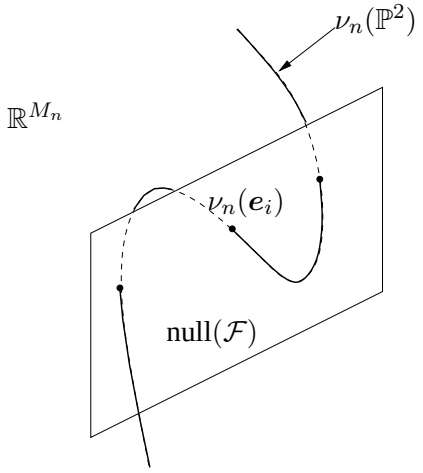



Figure 5.2: The intersection of $\nu_n(\mathbb{P}^2)$ and null($\mathcal{F}$) is exactly $n$ points representing the Veronese map of the $n$ epipoles, repeated or not.

The question is now how to actually compute the intersection of null($\mathcal{F}$) with $\nu_n(\mathbb{P}^2)$. One possible approach, explored in [70] for $n = 2$ and generalized in [64] to $n \geq 2$, consists of determining a vector $v \in \mathbb{R}^n$ such that $Bv \in \nu_n(\mathbb{P}^2)$, where $B$ is a matrix whose columns form a

basis for the (left) null space of $\mathcal{F}$. Finding $v$, hence the epipoles, is equivalent to solving for the roots of polynomials of degree $n$ in $n-1$ variables. Although this is feasible for $n=2$ and even for $n=3$, it is computationally formidable for $n>3$.

In the next section, we take a completely different approach that combines the multibody epipolar geometry developed so far with the GPCA problem studied in Chapter 3. In essence, we will show that the epipolar lines can be directly computed from the partial derivatives of the multibody epipolar constraint, while the epipoles can be computed by applying GPCA to the set of epipolar lines. Given the epipoles and the epipolar lines, the computation of individual fundamental matrices becomes a *linear* problem.

## 5.5 Multibody motion estimation and segmentation

Given the multibody fundamental matrix $\mathcal{F}$ and the number of independent motions $n$, we are now interested in recovering the motion parameters (or fundamental matrices) and the segmentation of the image points. In this section, we show how to solve these two problems from the epipoles of each fundamental matrix and the epipolar lines associated with each image point. We first show that one can estimate the epipolar line associated with each image pair from the partial derivatives of the multibody epipolar constraint. Given the epipolar lines, the estimation of the epipoles will be based on the factorization of a given homogeneous polynomial of degree $n$ in 3 variables into $n$ distinct polynomials of degree 1, i.e., the GPCA problem discussed in Chapter 3. Once the epipoles and the epipolar lines have been estimated, the estimation of individual fundamental matrices becomes a simple *linear* problem from which the segmentation of the image points can be automatically obtained.

### 5.5.1 Estimating the epipolar lines $\{\boldsymbol{\ell}^j\}_{j=1}^N$

Given a point $\boldsymbol{x}_1$ in the first image frame, the epipolar lines associated with it are defined as $\boldsymbol{\ell}_i \doteq F_i \boldsymbol{x}_1 \in \mathbb{R}^3$, $i = 1, \ldots, n$. From the epipolar constraint, we know that one of such lines passes through the corresponding point in the second frame $\boldsymbol{x}_2$, i.e., there exists an $i$ such that $\boldsymbol{x}_2^T \boldsymbol{\ell}_i = 0$. Let $\mathcal{F}$ be the multibody fundamental matrix. We have that

$$\mathcal{E}(\boldsymbol{x}_1, \boldsymbol{x}_2) = \nu_n(\boldsymbol{x}_2)^T \mathcal{F} \nu_n(\boldsymbol{x}_1) = \prod_{i=1}^{n} \left(\boldsymbol{x}_2^T F_i \boldsymbol{x}_1\right) = \prod_{i=1}^{n} (\boldsymbol{x}_2^T \boldsymbol{\ell}_i), \tag{5.23}$$

from which we conclude that the vector $\tilde{\boldsymbol{\ell}} \doteq \mathcal{F}\nu_n(\boldsymbol{x}_1) \in \mathbb{R}^{M_n}$ represents the coefficients of the homogeneous polynomial in $\boldsymbol{x}$

$$\boxed{g_n(\boldsymbol{x}) \doteq (\boldsymbol{x}^T\boldsymbol{\ell}_1)(\boldsymbol{x}^T\boldsymbol{\ell}_2)\cdots(\boldsymbol{x}^T\boldsymbol{\ell}_n) = \nu_n(\boldsymbol{x})^T\tilde{\boldsymbol{\ell}}.} \tag{5.24}$$

We call the vector $\tilde{\boldsymbol{\ell}}$ the *multibody epipolar line* associated with $\boldsymbol{x}_1$. Notice that $\tilde{\boldsymbol{\ell}}$ is a vector representation of the symmetric tensor product of all the epipolar lines $\boldsymbol{\ell}_1, \ldots, \boldsymbol{\ell}_n$ and is in general *not* the Veronese map (or lifting) $\nu_n(\boldsymbol{\ell}_i)$ of any particular epipolar line $\boldsymbol{\ell}_i, i = 1, \ldots, n$.

From $\tilde{\boldsymbol{\ell}}$, we can compute the individual epipolar lines $\{\boldsymbol{\ell}_i\}_{i=1}^n$ associated with any image point $\boldsymbol{x}_1$ using the polynomial factorization technique given in Section 3.3.2. Given the $n$ epipolar lines $\{\boldsymbol{\ell}_i\}_{i=1}^n$ associated with $\boldsymbol{x}_1$, we can compute *the* epipolar line associated with the image pair $(\boldsymbol{x}_1, \boldsymbol{x}_2)$ in the second view as the vector $\boldsymbol{\ell}_i$ that minimizes $(\boldsymbol{x}_2^T\boldsymbol{\ell}_i)^2$.

In essence, the multibody fundamental matrix $\mathcal{F}$ allows us to "transfer" a point $\boldsymbol{x}_1$ in the first image to a set of epipolar lines in the second image. This is exactly the multibody version of the conventional "epipolar transfer" that maps a point in the first image to an epipolar line in the second image. The *multibody epipolar transfer* process can be described by the sequence of maps

$$\boldsymbol{x}_1 \overset{Veronese}{\longmapsto} \nu_n(\boldsymbol{x}_1) \overset{Epipolar\ Transfer}{\longmapsto} \mathcal{F}\nu_n(\boldsymbol{x}_1) \overset{Polynomial\ Factorization}{\longmapsto} \{\boldsymbol{\ell}_i\}_{i=1}^n,$$

which is illustrated geometrically in Figure 5.3.

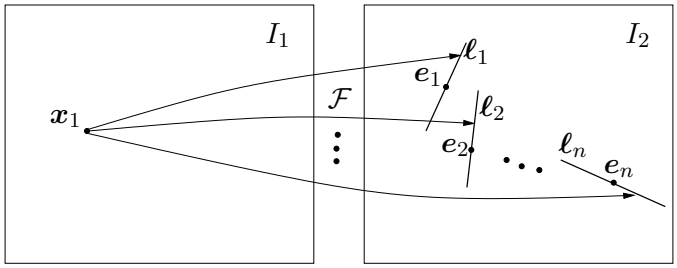



Figure 5.3: The multibody fundamental matrix $\mathcal{F}$ maps each point $\boldsymbol{x}_1$ in the first image to $n$ epipolar lines $\boldsymbol{\ell}_1, \ldots, \boldsymbol{\ell}_n$ which pass through the $n$ epipoles $\boldsymbol{e}_1, \ldots, \boldsymbol{e}_n$ respectively. Furthermore, one of these epipolar lines passes through $\boldsymbol{x}_2$.

Inspired by the GPCA polynomial differentiation algorithm described in Section 3.3.3, we now present a simpler and more elegant way of computing the epipolar line $\boldsymbol{\ell}$ associated with an image pair $(\boldsymbol{x}_1, \boldsymbol{x}_2)$. To this end, we notice from equation (5.23) that the partial derivate of the multibody epipolar constraint with respect to $\boldsymbol{x}_2$ is given by

$$\frac{\partial}{\partial \boldsymbol{x}_2}\nu_n(\boldsymbol{x}_2)^T\mathcal{F}\nu_n(\boldsymbol{x}_1) = \sum_{i=1}^{n}\prod_{\ell\neq i}(\boldsymbol{x}_2^T F_\ell \boldsymbol{x}_1)(F_i\boldsymbol{x}_1). \tag{5.25}$$

Therefore, if the image pair $(\boldsymbol{x}_1, \boldsymbol{x}_2)$ corresponds to motion $i$, i.e., if $\boldsymbol{x}_2^T F_i \boldsymbol{x}_1 = 0$, then

$$\frac{\partial}{\partial \boldsymbol{x}_2} \nu_n(\boldsymbol{x}_2)^T \mathcal{F} \nu_n(\boldsymbol{x}_1) = \prod_{\ell \neq i} (\boldsymbol{x}_2^T F_\ell \boldsymbol{x}_1)(F_i \boldsymbol{x}_1) \sim F_i \boldsymbol{x}_1 = \boldsymbol{\ell}_i. \tag{5.26}$$

In other words, the partial derivative of the multibody epipolar constraint with respect to $\boldsymbol{x}_2$ evaluated at $(\boldsymbol{x}_1, \boldsymbol{x}_2)$ is proportional to *the* epipolar line associated with $(\boldsymbol{x}_1, \boldsymbol{x}_2)$ in the second view. Similarly, the partial derivative of the multibody epipolar constraint with respect to $\boldsymbol{x}_1$ evaluated at $(\boldsymbol{x}_1, \boldsymbol{x}_2)$ is proportional to *the* epipolar line associated with $(\boldsymbol{x}_1, \boldsymbol{x}_2)$ in the first view.

**Remark 35** *Notice that if a particular image pair $(\boldsymbol{x}_1, \boldsymbol{x}_2)$ belongs to two motions, then the multibody epipolar constraint has a repeated factor, hence its derivative at that image pair is zero. Therefore, we cannot compute the epipolar line associated with image pairs that correspond to two or more motions from the derivatives of the multibody fundamental matrix.*

We summarize our discussion so far with the following statement.

**Theorem 13 (Epipolar lines from the multibody fundamental matrix)** *Let $\mathcal{F} \in \mathbb{R}^{M_n \times M_n}$ be a multibody fundamental matrix generated by $n$ different fundamental matrices $\{F_i\}_{i=1}^n$. Also let $(\boldsymbol{x}_1, \boldsymbol{x}_2)$ be an image pair associated with only one of the motions, i.e., $\boldsymbol{x}_2^T (F_i - F_j) \boldsymbol{x}_1 \neq 0$ for $i \neq j = 1, \ldots, n$. Then one can compute the epipolar line $\boldsymbol{\ell}_1$ associated with the image pair $(\boldsymbol{x}_1, \boldsymbol{x}_2)$ in the first view as*

$$\boxed{\boldsymbol{\ell}_1 \sim \frac{\partial}{\partial \boldsymbol{x}_1} \nu_n(\boldsymbol{x}_2)^T \mathcal{F} \nu_n(\boldsymbol{x}_1),} \tag{5.27}$$

*and the epipolar line $\boldsymbol{\ell}_2$ associated with the image pair $(\boldsymbol{x}_1, \boldsymbol{x}_2)$ in the second view as*

$$\boxed{\boldsymbol{\ell}_2 \sim \frac{\partial}{\partial \boldsymbol{x}_2} \nu_n(\boldsymbol{x}_2)^T \mathcal{F} \nu_n(\boldsymbol{x}_1).} \tag{5.28}$$

### 5.5.2 Estimating the epipoles $\{\boldsymbol{e}_i\}_{i=1}^n$

Given a set of epipolar lines, we now describe how to compute the epipoles. Recall that the (left) epipole associated with each rank-2 fundamental matrix $F_i \in \mathbb{R}^{3 \times 3}$ is defined as the vector $\boldsymbol{e}_i \in \mathbb{R}^3$ lying in the (left) null space of $F_i$, that is $\boldsymbol{e}_i$ satisfies that $\boldsymbol{e}_i^T F_i = 0$. Now let $\boldsymbol{\ell} \in \mathbb{R}^3$ be an arbitrary epipolar line associated with some image point in the first frame. Then there exists an $i$ such that $\boldsymbol{e}_i^T \boldsymbol{\ell} = 0$. Therefore, every epipolar line $\boldsymbol{\ell}$ has to satisfy the following polynomial constraint

$$\boxed{h_n(\boldsymbol{\ell}) \doteq (\boldsymbol{e}_1^T \boldsymbol{\ell})(\boldsymbol{e}_2^T \boldsymbol{\ell}) \cdots (\boldsymbol{e}_n^T \boldsymbol{\ell}) = \tilde{\boldsymbol{e}}^T \nu_n(\boldsymbol{\ell}) = 0,} \tag{5.29}$$

regardless of the motion with which it is associated. We call the vector $\tilde{\boldsymbol{e}} \in \mathbb{R}^{M_n}$ the *multibody epipole* associated with the $n$ motions. As before, $\tilde{\boldsymbol{e}}$ is a vector representation of the symmetric tensor product of the individual epipoles $\boldsymbol{e}_1, \ldots, \boldsymbol{e}_n$ and it is in general different from any of the embedded epipoles $\nu_n(\boldsymbol{e}_i)$, $i = 1, \ldots, n$.

Given a collection $\{\boldsymbol{\ell}^j\}_{j=1}^m$ of $m \geq M_n - 1$ epipolar lines (which can be computed from the multibody epipolar transfer or from the derivatives of the multibody epipolar constraint), we can obtain the multibody epipole $\tilde{\boldsymbol{e}} \in \mathbb{R}^{M_n}$ as the solution to the linear system

$$P_n \tilde{\boldsymbol{e}} \doteq \begin{bmatrix} \nu_n(\boldsymbol{\ell}^1)^T \\ \nu_n(\boldsymbol{\ell}^2)^T \\ \vdots \\ \nu_n(\boldsymbol{\ell}^m)^T \end{bmatrix} \tilde{\boldsymbol{e}} = 0. \tag{5.30}$$

In order for equation (5.30) to have a unique solution (up to a scale factor), we will need to replace $n$ by the number of distinct epipoles $n_e$, as stated by the following proposition:

**Proposition 5 (Number of distinct epipoles)** *Let $\{\boldsymbol{\ell}^j\}_{j=1}^m$ be a given collection of $m \geq M_n - 1$ epipolar lines corresponding to 3-D points in general configuration and undergoing $n$ distinct rigid body motions with nonzero translation (relative to the camera). Let $P_i \in \mathbb{R}^{N \times M_i}$ be the matrix defined in (5.30), but computed using the Veronese map $\nu_i$ of degree $i \geq 1$. Then we have*

$$\textit{rank}(P_i) \begin{cases} > M_i - 1, & \textit{if } i < n_e, \\ = M_i - 1, & \textit{if } i = n_e, \\ < M_i - 1, & \textit{if } i > n_e. \end{cases} \tag{5.31}$$

*Therefore, the number of distinct epipoles $n_e \leq n$ is given by*

$$\boxed{n_e \doteq \min\{i : \textit{rank}(P_i) = M_i - 1\}.} \tag{5.32}$$

*Proof.* Similar to the proof of Theorem 11. ∎

Once the number of distinct epipoles, $n_e$, has been computed, the vector $\tilde{\boldsymbol{e}} \in M_{n_e}$ can be obtained from the linear system $P_{n_e} \tilde{\boldsymbol{e}} = 0$. Once $\tilde{\boldsymbol{e}}$ has been computed, the individual epipoles $\{\boldsymbol{e}_i\}_{i=1}^{n_e}$ can be computed from $\tilde{\boldsymbol{e}}$ by applying any of the GPCA algorithms for hyperplanes to the data $\{\boldsymbol{\ell}^j\}_{j=1}^m$, as described in Section 3.3. We illustrate the computation of the epipoles in Figure 5.4. Each epipole $\boldsymbol{e}_i$ corresponds to the intersection of the epipolar lines associated with the $i^{th}$ motion. Either of the GPCA algorithms performs all such intersections simultaneously *without* knowing the segmentation of the epipolar lines.

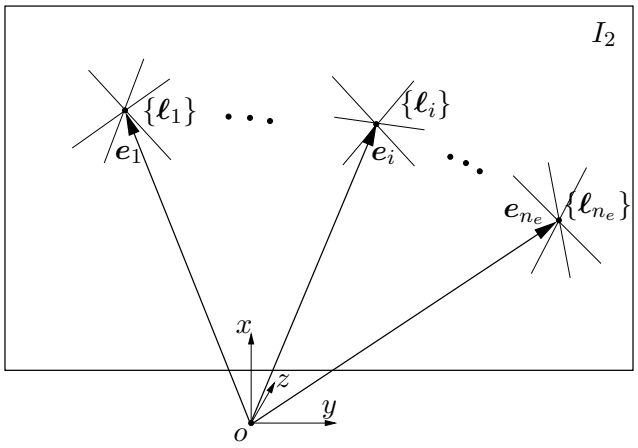



Figure 5.4: When $n$ objects move independently, the epipolar lines in the second view associated with each image point in the first view form $n_e$ groups and intersect respectively at $n_e$ distinct epipoles in the second view.

### 5.5.3 Estimating the individual fundamental matrices $\{F_i\}_{i=1}^n$

Given the epipolar lines and the epipoles, we show now how to recover each one of the individual fundamental matrices $\{F_i\}_{i=1}^n$. To avoid degenerate cases, we assume that all the epipoles are distinct,[6] i.e.,

$$n_e = n.$$

We first notice that, thanks to Theorem 13, given each image pair $(\boldsymbol{x}_1^j, \boldsymbol{x}_2^j)$, $j = 1, \ldots, N$, we know how to compute its epipolar line $\boldsymbol{\ell}^j$ in the second view. Furthermore, from our discussion in Section 5.5.2, we also know how to compute the $n$ different epipoles. Therefore, we can immediately compute the fundamental matrices using either of the following procedures

1. *Fundamental matrices from eight-point algorithm*: Assign image pair $(\boldsymbol{x}_1^j, \boldsymbol{x}_2^j)$ to group $i$ if $i = \arg\min_{\ell=1,\ldots n} (\boldsymbol{e}_i^T \boldsymbol{\ell}^j)^2$. Then compute the fundamental matrix $F_i$ by applying the eight-point algorithm to the image pairs in group $i$, where $i = 1, \ldots, n$.

2. *Fundamental matrices from epipolar lines*: Assign the $j^{th}$ image point $\boldsymbol{x}_1^j$ to group $i$ if $i = \arg\min_{\ell=1,\ldots n} (\boldsymbol{e}_i^T \boldsymbol{\ell}^j)^2$. If the image point $\boldsymbol{x}_1^j$ belongs to group $i$, then we must have $\boldsymbol{\ell}^j \sim$

[6] Notice that this is not a strong assumption. If two individual fundamental matrices share the same (left) epipoles, one can consider the right epipoles (in the first image frame) instead, because it is extremely rare that two motions give rise to the same left and right epipoles. In fact, this happens only when the rotation axes of the two motions are equal to each other and parallel to the translation direction.

$F_i \boldsymbol{x}_1^j$. Therefore, we can compute fundamental matrix $F_i$ by solving the set of equations

$$[\boldsymbol{\ell}^j]_\times F_i \boldsymbol{x}_1^j = 0 \qquad \text{for all} \quad j \quad \text{in group} \quad i. \tag{5.33}$$

We summarize the results of Sections 5.5.1- 5.5.3 with the following statement:

**Theorem 14 (Factorization of the multibody fundamental matrix)**
*Let $\mathcal{F} \in \mathbb{R}^{M_n \times M_n}$ be the multibody fundamental matrix associated with fundamental matrices $\{F_i \in \mathbb{R}^{3\times 3}\}_{i=1}^n$. If the $n$ epipoles $\{\boldsymbol{e}_i : \boldsymbol{e}_i^T F_i = 0\}$ are distinct, then the matrices $\{F_i\}_{i=1}^n$ can be uniquely determined (up to a scale factor) using polynomial factorization. Therefore, motion estimation and segmentation can be solved in closed form if and only if $n \leq 4$.*

### 5.5.4 Segmenting the feature points

Feature segmentation refers to the problem of assigning each image pair $\{(\boldsymbol{x}_1^j, \boldsymbol{x}_2^j)\}_{j=1}^N$, to the motion it corresponds. We notice from the previous section that both algorithms for estimating the fundamental matrices are based on first clustering the image pairs from epipoles and epipolar lines. More specifically, an image pair $(\boldsymbol{x}_1^j, \boldsymbol{x}_2^j)$ is assigned to group $i$ if

$$i = \arg \min_{\ell=1,\ldots n} (\boldsymbol{e}_i^T \boldsymbol{\ell}^j)^2.$$

In the presence of noise, one could improve the clustering of the feature points by using the already computed fundamental matrices. For example, we can assign image pair $(\boldsymbol{x}_1^j, \boldsymbol{x}_2^j)$ to group $i$ if

$$i = \arg \min_{\ell=1,\ldots,n} (\boldsymbol{x}_2^{jT} F_i \boldsymbol{x}_1^j)^2. \tag{5.34}$$

However, the square of the epipolar constraint is only an algebraic way of measuring how close an image pair $(\boldsymbol{x}_1, \boldsymbol{x}_2)$ is to satisfying the epipolar constraint. In the presence of noise, we assign image pair $(\boldsymbol{x}_1^j, \boldsymbol{x}_2^j)$ to the group $i$ that minimizes the square of the normalized epipolar constraint[7] [36], i.e.,

$$i = \arg \min_{\ell=1,\ldots,n} \frac{(\boldsymbol{x}_2^{jT} F_i \boldsymbol{x}_1^j)^2}{\|[e_3]_\times F_i^T \boldsymbol{x}_2^j\|^2 + \|[e_3]_\times F_i \boldsymbol{x}_1^j\|^2}. \tag{5.35}$$

Figure 5.5 illustrates how a particular image pair, say $(\boldsymbol{x}_1, \boldsymbol{x}_2)$, which belongs to the $i^{th}$ motion, $i = 1, \ldots, n$ is successfully segmented.

[7] We will justify the choice of the normalized epipolar constraint in Section 5.6.

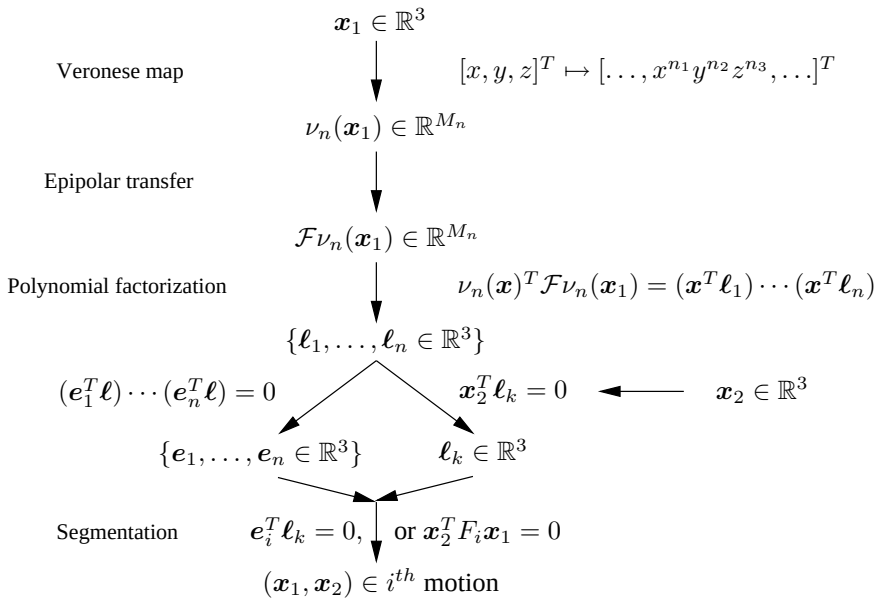



Figure 5.5: Transformation diagram associated with the segmentation of an image pair $(\boldsymbol{x}_1, \boldsymbol{x}_2)$ in the presence of $n$ motions.

### 5.5.5 Two-view multibody structure from motion algorithm

We are now ready to present a complete algorithm for multibody motion estimation and segmentation from two perspective views. Given a collection of $N \geq M_n^2 - 1$ image pairs $\{(\boldsymbol{x}_1^j, \boldsymbol{x}_2^j)\}_{j=1}^N$, Algorithm 10 determines the number of independent motions $n$, the individual fundamental matrices $\{F_i\}_{i=1}^n$ and the segmentation of the image pairs. Therefore, Algorithm 10 is a natural generalization of the eight-point algorithm to the case of multiple rigid motions.

However, we must notice that a drawback of Algorithm 10 is that it needs a lot of image pairs in order to compute the multibody fundamental matrix, which often makes it impractical for large $n$ (See Remark 36 below). In practice, one can significantly reduce the data requirements by incorporating partial knowledge about the motion or segmentation of the objects with minor changes in the general algorithm. We discuss a few of such possible variations below.

**Multiple linearly moving objects.** In many practical situations, the motion of the objects can be well approximated by a linear motion, i.e., there is only translation but no rotation. As discussed in Section 3.9.4, in this case the epipolar constraint reduces to $\boldsymbol{x}_2^T [\boldsymbol{e}_i]_\times \boldsymbol{x}_1 = 0$ or $\boldsymbol{e}_i^T [\boldsymbol{x}_2]_\times \boldsymbol{x}_1 = 0$, where $\boldsymbol{e}_i \in \mathbb{R}^3$ represents the epipole associated with the $i^{th}$ motion, $i = 1, \ldots, n$. Therefore, the vector $\boldsymbol{\ell} = [\boldsymbol{x}_2]_\times \boldsymbol{x}_1 \in \mathbb{R}^3$ is an epipolar line satisfying $g_n(\boldsymbol{\ell}) = (\boldsymbol{e}_1^T \boldsymbol{\ell})(\boldsymbol{e}_2^T \boldsymbol{\ell}) \cdots (\boldsymbol{e}_n^T \boldsymbol{\ell}) = 0$.

**Algorithm 10 (Two-view multibody structure from motion algorithm).**

Given a collection of image pairs $\{(\boldsymbol{x}_1^j, \boldsymbol{x}_2^j)\}_{j=1}^N$ of points undergoing $n$ different rigid motions relative to the camera, recover the number of independent motions $n$, the fundamental matrix $F_i$ associated with motion $i = 1, \ldots, n$, and the motion associated with each image pair as follows:

1. **Number of motions.** Compute the number of independent motions $n$ from the rank constraint in (5.9), by applying the Veronese map of degree $i = 1, 2, \ldots, n$ to the image points $\{(\boldsymbol{x}_1^j, \boldsymbol{x}_2^j)\}_{j=1}^N$.

2. **Multibody fundamental matrix.** Compute the multibody fundamental matrix $\mathcal{F}$ as the solution of the linear system $L_n \boldsymbol{f} = 0$ in (5.7), using the Veronese map of degree $n$.

3. **Epipolar transfer.** For each image point $\{\boldsymbol{x}_1^j\}_{j=1}^N$ compute its epipolar line in the second view $\{\boldsymbol{\ell}^j\}_{j=1}^N$ from the partial derivative of the multibody epipolar constraint with respect to $\boldsymbol{x}_2$ evaluated at $(\boldsymbol{x}_1^j, \boldsymbol{x}_2^j)$, as described in Theorem 13.

4. **Multibody epipole.** Use the epipolar lines $\{\boldsymbol{\ell}^j\}_{j=1}^N$ to estimate the multibody epipole $\tilde{\boldsymbol{e}}$ as the coefficients of the polynomial $h_n(\boldsymbol{\ell})$ in (5.29) by solving the system $P_n \tilde{\boldsymbol{e}} = 0$ in (5.30).

5. **Individual epipoles.** Use the polynomial factorization algorithm of Section 3.3.2 or the polynomial differentiation algorithm of Section 3.3.3 to compute the individual epipoles $\{\boldsymbol{e}_i\}_{i=1}^n$ from the multibody epipole $\tilde{\boldsymbol{e}} \in \mathbb{R}^{M_n}$.

6. **Individual fundamental matrices.** Assign each image pair $(\boldsymbol{x}_1^j, \boldsymbol{x}_2^j)$ to group $i$ if $i = \arg\min_{\ell=1,\ldots n} (\boldsymbol{e}_i^T \boldsymbol{\ell}^j)^2$, and then compute the fundamental matrix $F_i$ from the set of linear equations $\boldsymbol{x}_2^{jT} F_i \boldsymbol{x}_1^j = 0$ or $[\boldsymbol{\ell}^j]_\times F_i \boldsymbol{x}_1^j = 0$, as described in Section 5.5.3.

7. **Feature segmentation from fundamental matrices.** Assign image pair $(\boldsymbol{x}_1^j, \boldsymbol{x}_2^j)$ to motion $i$ if

$$i = \arg \min_{\ell=1,\ldots,n} \frac{(\boldsymbol{x}_2^{jT} F_i \boldsymbol{x}_1^j)^2}{\|[e_3]_\times F_i^T \boldsymbol{x}_2^j\|^2 + \|[e_3]_\times F_i \boldsymbol{x}_1^j\|^2}.$$

Therefore, given a set of image pairs $\{(\boldsymbol{x}_1^j, \boldsymbol{x}_2^j)\}_{j=1}^N$ corresponding to $N$ points undergoing $n$ *distinct* linear motions $\{\boldsymbol{e}_i \in \mathbb{R}^3\}_{i=1}^n$, one can use the set of epipolar lines $\{\boldsymbol{\ell}^j = [\boldsymbol{x}_2^j]_\times \boldsymbol{x}_1^j\}_{j=1}^N$ to estimate the epipoles $\boldsymbol{e}_i$ using Steps 4 and 5 of Algorithm 10. Notice that the epipoles are recovered directly using polynomial factorization or differentiation *without* estimating the multibody fundamental matrix $\mathcal{F}$ first. Furthermore, given the epipoles, the fundamental matrix is trivially obtained as $F_i = [\boldsymbol{e}_i]_\times$. The segmentation of the image points is then obtained from Step 7 of Algorithm 10. We conclude that if the motions are linear, we only need $N = M_n - 1$ image pairs versus $N = M_n^2 - 1$ needed in the general case. So when $n$ is large, the number of image pairs needed grows as $O(n^2)$ for the linear motion case versus $O(n^4)$ for the general case. In other words, the number of feature points that need to be tracked on each object grows linearly in the number of independent motions. For instance, when $n = 10$, one only needs to track 7 points on each object, which is a mild requirement given that the case $n = 10$ occurs rather rarely in most applications.

**Constant motions.** In many vision and control applications, the motion of the objects in the scene changes slowly relative to the sampling rate. Thus, if the image sampling rate is even, we may assume that for a number of image frames, say $m$, the motion of each object between consecutive pairs of images is the same. Hence *all* the feature points corresponding to the $m - 1$ image pairs in between can be used to estimate the *same* multibody fundamental matrix. For example, when $m = 5$ and $n = 4$, we only need to track $(M_4^2 - 1)/4 = 225/4 \approx 57$ image points between each of the $4$ consecutive pairs of images instead of $255$. That is about $57/4 \approx 15$ features on each object on each image frame, which is rather feasible to do in practice. In general if $m = O(n)$, $O(n^2)$ feature points per object need to be tracked in each image. For example, when $m = n + 1 = 6$, one needs to track about $18$ points on each object, which is not so demanding given the nature of the problem.

**Internal structure of the multibody fundamental matrix.** The only step of Algorithm 10 that requires $O(n^4)$ image pairs is the estimation of the multibody fundamental matrix $\mathcal{F}$. Step 2 requires a lot of data points, because $\mathcal{F}$ is estimated linearly without taking into account the rich internal (algebraic) structure of $\mathcal{F}$ (e.g., $\text{rank}(\mathcal{F}) \leq M_n - n$). In the future, we expect to be able to reduce the number of image pairs needed by considering constraints among entries of $\mathcal{F}$, in the same spirit that the well-known $8$-point algorithm for $n = 1$ can be reduced to 7 points if the algebraic property $\det(F) = 0$ is used.

**Remark 36 (Comments about the algorithm)**

1. **Repeated epipoles.** *If two individual fundamental matrices share the same (left) epipoles, we cannot segment the epipolar lines as described in Step 6 of Algorithm 10. In this case, one can consider the right epipoles (in the first image frame) instead, since it is extremely rare that two motions give rise to the same left and right epipoles.*[8]

2. **Repeated roots.** *If in Step 5 one uses polynomial factorization to recover the epipoles, then recall from Section 3.3.2 that when the polynomial $q_n(t)$ in (3.22) has repeated roots or more than one of its leading coefficient is zero one must pre-apply a linear transformation (3.31) to the polynomial $p_n(\boldsymbol{x})$ before factoring it.*

3. **Algebraic solvability.** *The only nonlinear part of Algorithm 10 is to solve for the roots of univariate polynomials of degree $n$ in Step 5. Therefore, the multibody structure from motion problem is algebraically solvable (i.e., there is closed form solution) if and only if the number of motions is $n \leq 4$ (see [34]). When $n \geq 5$, the above algorithm must rely on a numerical solution for the roots of those polynomials.*

4. **Computational complexity.** *In terms of data, Algorithm 10 requires $O(n^4)$ image pairs to estimate the multibody fundamental matrix $\mathcal{F}$ associated with the $n$ motions. In terms of numerical computation, it needs to factor $O(n)$ polynomials*[9] *and hence solve for the roots of $O(n)$ univariate polynomials of degree $n$.*[10] *As for the rest of computation, which can be well approximated by the most costly Steps 1 and 2, the complexity is about $O(n^6)$.*

5. **Special motions and structures.** *Algorithm 10 works for distinct motions with nonzero translation. Future research is needed for special motions, e.g., pure rotation or repeated epipoles parallel to the rotation axis, and for special structures, e.g., planar objects. We gave a solution for the case of linearly moving objects in Section 3.9.4.*

6. **Noise sensitivity.** *Algorithm 10 gives a purely algebraic solution to the multibody structure from motion problem. Future research will need to address the sensitivity of the algorithm to*

[8] This happens only when the rotation axes of the two motions are equal to each other and parallel to the translation direction.

[9] One needs about $M_n - 1 \approx O(n^2)$ epipolar lines to compute the epipoles and fundamental matrices, which can be obtained from $O(n)$ polynomial factorizations since each one generates $n$ epipolar lines. Hence it is not necessary to compute the epipolar lines for all $N = M_n^2 - 1 \approx O(n^4)$ image pairs in Step 3.

[10] The numerical complexity of solving for the roots for an $n^{th}$ order polynomial in one variable is polynomial in $n$ for a given error bound, see [47].

*noise in the image measurements. In particular, one should pay attention to Step 2, which is sensitive to noise, because it does not exploit the algebraic structure of the multibody fundamental matrix $\mathcal{F}$.*

7. **Optimality.** *Notice that linearly solving for the multibody fundamental matrix through the Veronese embedding is sub-optimal from a statistical point of view. We will derive the optimal function for motion estimation and segmentation in Section 5.6.*

At the end of our theoretical development, Table 5.1 summarizes our results with a comparison of the geometric entities associated with two views of 1 rigid body motion and two views of $n$ rigid body motions.

| Comparison of | 2 views of 1 body | 2 views of $n$ bodies |
|---|---|---|
| An image pair | $\boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathbb{R}^3$ | $\nu_n(\boldsymbol{x}_1), \nu_n(\boldsymbol{x}_2) \in \mathbb{R}^{M_n}$ |
| Epipolar constraint | $\boldsymbol{x}_2^T F \boldsymbol{x}_1 = 0$ | $\nu_n(\boldsymbol{x}_2)^T \mathcal{F} \nu_n(\boldsymbol{x}_1) = 0$ |
| Fundamental matrix | $F \in \mathbb{R}^{3\times 3}$ | $\mathcal{F} \in \mathbb{R}^{M_n \times M_n}$ |
| Linear estimation from $N$ image pairs | $\begin{bmatrix} \boldsymbol{x}_2^1 \otimes \boldsymbol{x}_1^1 \\ \boldsymbol{x}_2^2 \otimes \boldsymbol{x}_1^2 \\ \vdots \\ \boldsymbol{x}_2^N \otimes \boldsymbol{x}_1^N \end{bmatrix} \boldsymbol{f} = 0$ | $\begin{bmatrix} \nu_n(\boldsymbol{x}_2^1) \otimes \nu_n(\boldsymbol{x}_1^1) \\ \nu_n(\boldsymbol{x}_2^2) \otimes \nu_n(\boldsymbol{x}_1^2) \\ \vdots \\ \nu_n(\boldsymbol{x}_2^N) \otimes \nu_n(\boldsymbol{x}_1^N) \end{bmatrix} \boldsymbol{f} = 0$ |
| Epipole | $\boldsymbol{e}^T F = 0$ | $\nu_n(\boldsymbol{e})^T \mathcal{F} = 0$ |
| Epipolar lines | $\boldsymbol{\ell} = F\boldsymbol{x}_1 \in \mathbb{R}^3$ | $\tilde{\boldsymbol{\ell}} = \mathcal{F}\nu_n(\boldsymbol{x}_1) \in \mathbb{R}^{M_n}$ |
| Epipolar line & point | $\boldsymbol{x}_2^T \boldsymbol{\ell} = 0$ | $\nu_n(\boldsymbol{x}_2)^T \tilde{\boldsymbol{\ell}} = 0$ |
| Epipolar line & epipole | $\boldsymbol{e}^T \boldsymbol{\ell} = 0$ | $\tilde{\boldsymbol{e}}^T \nu_n(\boldsymbol{\ell}) = 0$ |

Table 5.1: Comparison between the geometry for two views of 1 rigid body motion and the geometry of $n$ rigid body motions.

## 5.6 Optimal segmentation of 3-D rigid motions

The two-view multibody structure from motion algorithm provides an algebraic geometric solution to the problem of estimating a collection of fundamental matrices $\{F_i\}_{i=1}^n$ from image pairs $\{(\boldsymbol{x}_1^j, \boldsymbol{x}_2^j)\}_{j=1}^N$. In essence, Algorithm 10 solves the set of nonlinear equations $\prod_{i=1}^n (\boldsymbol{x}_2^{jT} F_i \boldsymbol{x}_1^j) = 0$, $j = 1, \ldots, N$, in a "linear" fashion by embedding the image pairs into a higher-dimensional space via the Veronese map.

However, such a "linear" solution is obtained at the cost of neglecting the internal nonlinear structure of the multibody fundamental matrix $\mathcal{F}$. For example, the algorithm solves for $M_n^2 - 1$ unknowns in $\mathcal{F} \in \mathbb{R}^{M_n \times M_n}$ from equation (5.7), even though there are only $8n$ unknowns in the fundamental matrices $\{F_i\}_{i=1}^n$ ($5n$ in the calibrated case). In practice, solving for an over-parameterized representation of the multibody fundamental matrix can be very sensitive in the presence of noise. One way of resolving this problem is to replace the algebraic algorithm by an optimization problem in which one obtains the individual fundamental matrices by minimizing the algebraic error

$$E_A(F_1, \ldots, F_n) = \sum_{j=1}^{N} (\nu_n(\boldsymbol{x}_2^j)^T \mathcal{F} \nu_n(\boldsymbol{x}_1^j))^2 = \sum_{j=1}^{N} \prod_{i=1}^{n} (\boldsymbol{x}_2^{jT} F_i^T \boldsymbol{x}_1^j)^2. \tag{5.36}$$

This nonlinear solution in fact provides a more robust estimate of the fundamental matrices, because it uses a minimal representation of the unknowns. However, the solution to this optimization problem is not optimal, because the algebraic error in (5.36) does not coincide with the negative log-likelihood of the motion parameters.

In this section, we derive an optimal algorithm for estimating the fundamental matrices when the image pairs 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 (5.36), but properly normalized. We cast the motion segmentation problem as a constrained nonlinear least squares problem which minimizes the re-projection error subject to all the multibody epipolar constraints. Since the multibody epipolar constraint is satisfied by *all* image pairs, irrespective of the segmentation, we do not need to model the membership of the image pairs to each one of the rigid motions with a probability distribution. Hence, we do not need to iterate between feature segmentation and single body motion estimation, as in EM-like techniques. In fact, the segmentation (E-step) is *algebraically eliminated* by the multibody epipolar constraint, which leads to an objective function that depends only on the motion parameters (fundamental matrices).

Let $\{(\boldsymbol{x}_1^j, \boldsymbol{x}_2^j)\}_{j=1}^N$ be the given collection of noisy image pairs. We would like to find a collection of fundamental matrices $\{F_i\}_{i=1}^n$ such that the corresponding noise free image pairs $\{(\tilde{\boldsymbol{x}}_1^j, \tilde{\boldsymbol{x}}_2^j)\}_{j=1}^N$ satisfy the multibody epipolar constraint $\nu_n(\tilde{\boldsymbol{x}}_2^j)^T \mathcal{F} \nu_n(\tilde{\boldsymbol{x}}_1^j) = 0$. Since for the Gaussian noise model the negative log-likelihood is equal to the re-projection error (up to constant terms),

we obtain the constrained optimization problem[11]

$$\begin{array}{ll} \min & \sum_{j=1}^{N} \|\tilde{\boldsymbol{x}}_1^j - \boldsymbol{x}_1^j\|^2 + \|\tilde{\boldsymbol{x}}_2^j - \boldsymbol{x}_2^j\|^2 \\ \text{subject to} & \nu_n(\tilde{\boldsymbol{x}}_2^j)^T \mathcal{F} \nu_n(\tilde{\boldsymbol{x}}_1^j) = 0 \quad j = 1, \ldots, N. \end{array} \tag{5.37}$$

By using Lagrange multipliers $\lambda^j \in \mathbb{R}$ for each constraint, the above optimization problem is equivalent to minimizing

$$\sum_{j=1}^{N} \|\tilde{\boldsymbol{x}}_1^j - \boldsymbol{x}_1^j\|^2 + \|\tilde{\boldsymbol{x}}_2^j - \boldsymbol{x}_2^j\|^2 + \lambda^j \nu_n(\tilde{\boldsymbol{x}}_2^j)^T \mathcal{F} \nu_n(\tilde{\boldsymbol{x}}_1^j). \tag{5.38}$$

Let

$$\Lambda = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} = [e_3]_\times^T [e_3]_\times, \tag{5.39}$$

with $e_3 = [0, 0, 1]^T \in \mathbb{R}^3$, be the projection matrix eliminating the third entry of any image point $\boldsymbol{x} = [x, y, 1]^T \in \mathbb{R}^3$. Since $\Lambda(\tilde{\boldsymbol{x}} - \boldsymbol{x}) = (\tilde{\boldsymbol{x}} - \boldsymbol{x})$, after left-multiplying the partial derivatives of the Lagrangian (5.38) with respect to $\tilde{\boldsymbol{x}}_1^j$ and $\tilde{\boldsymbol{x}}_2^j$ by the projection matrix $\Lambda$ we obtain

$$2(\tilde{\boldsymbol{x}}_1^j - \boldsymbol{x}_1^j) + \lambda^j \Lambda \left( D\nu_n(\tilde{\boldsymbol{x}}_1^j) \right)^T \mathcal{F}^T \nu_n(\tilde{\boldsymbol{x}}_2^j) = 0, \tag{5.40}$$

$$2(\tilde{\boldsymbol{x}}_2^j - \boldsymbol{x}_2^j) + \lambda^j \Lambda \left( D\nu_n(\tilde{\boldsymbol{x}}_2^j) \right)^T \mathcal{F} \nu_n(\tilde{\boldsymbol{x}}_1^j) = 0, \tag{5.41}$$

where $D\nu_n(\boldsymbol{x}) \in \mathbb{R}^{M_n \times 3}$ is the Jacobian of the Veronese map $\nu_n$.

For ease of notation, let us also define

$$\boldsymbol{g}_1^j = (D\nu_n(\tilde{\boldsymbol{x}}_1^j))^T \mathcal{F}^T \nu_n(\tilde{\boldsymbol{x}}_2^j) \quad \text{and} \quad \boldsymbol{g}_2^j = (D\nu_n(\tilde{\boldsymbol{x}}_2^j))^T \mathcal{F} \nu_n(\tilde{\boldsymbol{x}}_1^j). \tag{5.42}$$

Then, since $\Lambda^T \Lambda = \Lambda^2 = \Lambda$, after left-multiplying (5.40) and (5.41) by $\boldsymbol{g}_1^{jT} \Lambda$ and $\boldsymbol{g}_2^{jT} \Lambda$, respectively, we obtain

$$2\boldsymbol{g}_1^{jT} \Lambda(\tilde{\boldsymbol{x}}_1^j - \boldsymbol{x}_1^j) + \lambda^j \|[e_3]_\times \boldsymbol{g}_1^j\|^2 = 0, \tag{5.43}$$

$$2\boldsymbol{g}_2^{jT} \Lambda(\tilde{\boldsymbol{x}}_2^j - \boldsymbol{x}_2^j) + \lambda^j \|[e_3]_\times \boldsymbol{g}_2^j\|^2 = 0. \tag{5.44}$$

[11] Notice that the optimization problem (5.37) does not include as an additional constraint the fact that the third entry of each image point $\boldsymbol{x} = [x, y, 1]^T \in \mathbb{R}^3$ is equal to one. We will implicitly eliminate such constraints and their associated Lagrange multipliers by left-multiplying the partial derivatives of the Lagrangian (5.38) by the projection matrix $\Lambda$ in (5.39).

Since $\Lambda(\tilde{\boldsymbol{x}} - \boldsymbol{x}) = (\tilde{\boldsymbol{x}} - \boldsymbol{x})$, $D\nu_n(\tilde{\boldsymbol{x}})\tilde{\boldsymbol{x}} = n\nu_n(\tilde{\boldsymbol{x}})$ and $\nu_n(\tilde{\boldsymbol{x}}_2)\mathcal{F}\nu_n(\tilde{\boldsymbol{x}}_1) = 0$, we obtain $\boldsymbol{g}_1^{jT}\tilde{\boldsymbol{x}}_1^j = \boldsymbol{g}_2^{jT}\tilde{\boldsymbol{x}}_2^j = 0$. Therefore, we have

$$-2\boldsymbol{g}_1^{jT}\boldsymbol{x}_1^j + \lambda^j \|[e_3]_\times \boldsymbol{g}_1^j\|^2 = 0, \tag{5.45}$$

$$-2\boldsymbol{g}_2^{jT}\boldsymbol{x}_2^j + \lambda^j \|[e_3]_\times \boldsymbol{g}_2^j\|^2 = 0, \tag{5.46}$$

from which we can solve for $\lambda^j$ as

$$\lambda^j = \frac{2\big(\boldsymbol{g}_1^{jT}\boldsymbol{x}_1^j + \boldsymbol{g}_2^{jT}\boldsymbol{x}_2^j\big)}{\|[e_3]_\times \boldsymbol{g}_1^j\|^2 + \|[e_3]_\times \boldsymbol{g}_2^j\|^2}. \tag{5.47}$$

Similarly, after left-multiplying (5.40) by $(\tilde{\boldsymbol{x}}_1^j - \boldsymbol{x}_1^j)^T$ and (5.41) by $(\tilde{\boldsymbol{x}}_2^j - \boldsymbol{x}_2^j)^T$ we get

$$2\|\tilde{\boldsymbol{x}}_1^j - \boldsymbol{x}_1^j\|^2 - \lambda^j \boldsymbol{g}_1^{jT}\boldsymbol{x}_1^j = 0, \tag{5.48}$$

$$2\|\tilde{\boldsymbol{x}}_2^j - \boldsymbol{x}_2^j\|^2 - \lambda^j \boldsymbol{g}_2^{jT}\boldsymbol{x}_2^j = 0, \tag{5.49}$$

from which the re-projection error for point $j$ is given by

$$\|\tilde{\boldsymbol{x}}_1^j - \boldsymbol{x}_1^j\|^2 + \|\tilde{\boldsymbol{x}}_2^j - \boldsymbol{x}_2^j\|^2 = \frac{\lambda^j}{2}(\boldsymbol{g}_1^{jT}\boldsymbol{x}_1^j + \boldsymbol{g}_2^{jT}\boldsymbol{x}_2^j). \tag{5.50}$$

After replacing (5.47) in the previous equation, we obtain the following expression for the total re-projection error

$$\tilde{E}_O(\{F_i\}_{i=1}^n, \{(\tilde{\boldsymbol{x}}_1^j, \tilde{\boldsymbol{x}}_2^j)\}_{j=1}^N) \doteq \sum_{j=1}^N \frac{(\boldsymbol{g}_1^{jT}\boldsymbol{x}_1^j + \boldsymbol{g}_2^{jT}\boldsymbol{x}_2^j)^2}{\|[e_3]_\times \boldsymbol{g}_1^j\|^2 + \|[e_3]_\times \boldsymbol{g}_2^j\|^2} = \tag{5.51}$$

$$= \sum_{j=1}^N \frac{(\boldsymbol{x}_1^{jT}(D\nu_n(\tilde{\boldsymbol{x}}_1^j))^T\mathcal{F}^T\nu_n(\tilde{\boldsymbol{x}}_2^j) + \boldsymbol{x}_2^{jT}(D\nu_n(\tilde{\boldsymbol{x}}_2^j))^T\mathcal{F}\nu_n(\tilde{\boldsymbol{x}}_1^j))^2}{\|[e_3]_\times (D\nu_n(\tilde{\boldsymbol{x}}_1^j))^T\mathcal{F}^T\nu_n(\tilde{\boldsymbol{x}}_2^j)\|^2 + \|[e_3]_\times (D\nu_n(\tilde{\boldsymbol{x}}_2^j))^T\mathcal{F}\nu_n(\tilde{\boldsymbol{x}}_1^j)\|^2}.$$

Since $\nu_1(\boldsymbol{x}) = \boldsymbol{x}$ and $D\nu_1(\boldsymbol{x}) = I$, by letting $n = 1$ in the above expression we notice that $\tilde{E}_O$ is a natural generalization of the well-known optimal function for estimating a *single* fundamental matrix $F \in \mathbb{R}^{3\times 3}$, which is given by [36]

$$\tilde{E}_O(F, \{(\tilde{\boldsymbol{x}}_1^j, \tilde{\boldsymbol{x}}_2^j)\}_{j=1}^N) = \sum_{j=1}^N \frac{(\boldsymbol{x}_1^{jT}F^T\tilde{\boldsymbol{x}}_2^j + \boldsymbol{x}_2^{jT}F\tilde{\boldsymbol{x}}_1^j)^2}{\|[e_3]_\times F^T\tilde{\boldsymbol{x}}_2^j\|^2 + \|[e_3]_\times F\tilde{\boldsymbol{x}}_1^j\|^2}. \tag{5.52}$$

**Remark 37** *Notice that the optimal error $\tilde{E}_O$ has a very intuitive interpretation. If point $j$ belongs to group $i$, then $\tilde{\boldsymbol{x}}_2^{jT} F_i \tilde{\boldsymbol{x}}_1^j = 0$. This implies that*

$$\boldsymbol{g}_1^{jT} = \nu_n(\tilde{\boldsymbol{x}}_2^j)^T \mathcal{F} D\nu_n(\tilde{\boldsymbol{x}}_1^j) = \frac{\partial}{\partial \tilde{\boldsymbol{x}}_1^j}\left(\nu_n(\tilde{\boldsymbol{x}}_2^j)^T \mathcal{F} \nu_n(\tilde{\boldsymbol{x}}_1^j)\right) \tag{5.53}$$

$$= \frac{\partial}{\partial \tilde{\boldsymbol{x}}_1^j}\left(\prod_{i=1}^{n} \tilde{\boldsymbol{x}}_2^{jT} F_i \tilde{\boldsymbol{x}}_1^j\right) = \sum_{i=1}^{n}\left(\prod_{\ell\neq i}^{n} \tilde{\boldsymbol{x}}_2^{jT} F_\ell \tilde{\boldsymbol{x}}_1^j\right)(\tilde{\boldsymbol{x}}_2^{jT} F_i) \tag{5.54}$$

$$= \Big(\prod_{\ell\neq i}^{n} \tilde{\boldsymbol{x}}_2^{jT} F_\ell \tilde{\boldsymbol{x}}_1^j\Big)(\tilde{\boldsymbol{x}}_2^{jT} F_i), \tag{5.55}$$

*and similarly*

$$\boldsymbol{g}_2^{jT} = \Big(\prod_{\ell\neq i}^{n} \tilde{\boldsymbol{x}}_2^{jT} F_\ell \tilde{\boldsymbol{x}}_1^j\Big)(\tilde{\boldsymbol{x}}_1^{jT} F_i^T). \tag{5.56}$$

*Therefore, the factor $\prod_{\ell\neq i}^{n}(\tilde{\boldsymbol{x}}_2^{jT} F_\ell \tilde{\boldsymbol{x}}_1^j)$ is in both the numerator and the denominator of $\tilde{E}_O$. Hence the contribution of point $j$ to the error $\tilde{E}_O$ reduces to*

$$\frac{(\tilde{\boldsymbol{x}}_2^{jT} F_i \boldsymbol{x}_1^j + \boldsymbol{x}_2^{jT} F_i \tilde{\boldsymbol{x}}_1^j)^2}{\|[e_3]_\times F_i^T \tilde{\boldsymbol{x}}_2^j\|^2 + \|[e_3]_\times F_i \tilde{\boldsymbol{x}}_1^j\|^2}, \tag{5.57}$$

*which is the same as the contribution of point $j$ to the optimal function for a single fundamental matrix $F_i$ in (5.52). Therefore, the objective function $\tilde{E}_O$ is just a clever algebraic way of simultaneously writing a mixture of optimal objective functions for individual fundamental matrices into a* single *objective function for* all *the fundamental matrices.*

We now derive an objective function that depends on the motion parameters only. As in the case of a single fundamental matrix [36], this can be done by considering the first order statistics of $\nu_n(\boldsymbol{x}_2^j)^T \mathcal{F} \nu_n(\boldsymbol{x}_1^j)$. It turns out that this is equivalent to setting $\tilde{\boldsymbol{x}}^j = \boldsymbol{x}^j$ in the above expression for $\tilde{E}_O$. Since $D\nu_n(\boldsymbol{x})\boldsymbol{x} = n\nu_n(\boldsymbol{x})$ we obtain the following function of the fundamental matrices

$$\boxed{E_O(F_1, \ldots, F_n) \doteq \sum_{j=1}^{N} \frac{4n^2(\nu_n(\boldsymbol{x}_2^j)^T \mathcal{F} \nu_n(\boldsymbol{x}_1^j))^2}{\|[e_3]_\times (D\nu_n(\boldsymbol{x}_1^j))^T \mathcal{F}^T \nu_n(\boldsymbol{x}_2^j)\|^2 + \|[e_3]_\times (D\nu_n(\boldsymbol{x}_2^j))^T \mathcal{F} \nu_n(\boldsymbol{x}_1^j)\|^2}.}$$

Notice that $E_O$ is just a normalized version of the algebraic error (5.36). Furthermore, when $n = 1$, $E_O$ reduces to the well-known objective function, the so-called normalized epipolar constraint or Sampson distance, for estimating a single fundamental matrix $F \in \mathbb{R}^{3\times 3}$ [36]

$$E_O(F) = \sum_{j=1}^{N} \frac{4(\boldsymbol{x}_2^{jT} F \boldsymbol{x}_1^j)^2}{\|[e_3]_\times F^T \boldsymbol{x}_2^j\|^2 + \|[e_3]_\times F \boldsymbol{x}_1^j\|^2}. \tag{5.58}$$

Notice that (5.58) can also be obtained by setting $\tilde{\boldsymbol{x}} = \boldsymbol{x}$ in (5.52). Therefore, the objective function $E_O(F_1, \ldots, F_n)$ is a natural generalization of well-known objective function $E_O(F)$ in single body structure from motion.

In summary, we have derived an objective function from which one can simultaneously estimate all the fundamental matrices $\{F_i\}_{i=1}^n$ using all the image pairs $\{(\boldsymbol{x}_1^j, \boldsymbol{x}_2^j)\}_{j=1}^N$, without prior segmentation of the image measurements. The fundamental matrices can be obtained by minimizing $E_O$ using standard nonlinear optimization techniques. One can use the multibody linear algorithm (Algorithm 10) to initialize the number of motions and the fundamental matrices.

**Remark 38 (Pure translation and calibrated cases)** *The case of linearly moving objects (see Section 3.9.4) or calibrated cameras can be easily handled by properly parameterizing the fundamental matrices and then minimizing over fewer parameters.*

## 5.7 Experimental results

In this section, we present simulation results that evaluate the performance of our nonlinear motion segmentation algorithm with respect to the number of motions $n$ for different levels of noise. We also present experimental results on the segmentation of an indoor sequence.

We first test the algorithm on synthetic data. We randomly pick $n = 1, 2, 3, 4$ collections of $N = 100$ feature points and apply a different (randomly chosen) rigid body motion $(R_i, T_i)$, with $R_i \in SO(3)$ the rotation and $T_i \in \mathbb{R}^3$ the translation. Zero-mean Gaussian noise with standard deviation (std.) from 0 to 2.5 pixels is added to the images $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$. We run 1000 trials for each noise level. For each trial the error between the true motions $\{(R_i, T_i)\}_{i=1}^n$ and the estimates $\{(\hat{R}_i, \hat{T}_i)\}_{i=1}^n$ is computed as

$$\text{Rotation error} = \frac{1}{n}\sum_{i=1}^{n} \text{acos}\Big(\frac{\text{trace}(R_i \hat{R}_i^T) - 1}{2}\Big) \qquad \text{(degrees)}.$$

$$\text{Translation error} = \frac{1}{n}\sum_{i=1}^{n} \text{acos}\Big(\frac{T_i^T \hat{T}_i}{\|T_i\|\|\hat{T}_i\|}\Big) \qquad \text{(degrees)}.$$

Figure 5.6 plots the mean error in rotation and translation as a function of noise. The algorithm gives an error of less than $3^\circ$ for rotation and less than $10^\circ$ for translation. As expected, the performance deteriorates as the number of motions $n$ increases, especially for the error in rotation.

We also tested the proposed approach by segmenting a real image sequence with $n = 3$ moving objects: a truck, a car and a box. Figure 5.7(a) shows the first frame of the sequence

with the tracked features superimposed. We used the algorithm in [8] to track a total of $N = 173$ point features: 44 "$\circ$" for the truck, 48 "$\square$" for the car and 81 "$\triangle$" for the box. For comparison purposes, we estimated the ground truth motion $(R_i, T_i)$ of each object by manually segmenting the feature points and then minimizing the standard error for single-body structure from motion $E_O(F_i)$ in (5.52). Then, we estimated the number of motions from (5.9) as $n = 3$ and minimized $E_O(F_1, F_2, F_3)$ to obtain $(\hat{R}_i, \hat{T}_i)$. The error in rotation was $1.5^\circ$, $1.9^\circ$ and $0.1^\circ$ and the error in translation was $1.7^\circ$, $1.8^\circ$ and $0.4^\circ$ for the truck, car and box, respectively.

In order to obtain the segmentation of the feature pairs, we computed the three re-projection errors $E_O(\hat{F}_i)$ for each feature pair as shown in Figures 5.7(c)-(e). Each feature pair was assigned to the motion $i = 1, 2, 3$ with the minimum error. Figure 5.7(b) plots the segmentation of the image points. There are no mismatches for motions 1 and 3. However 5 features corresponding to motion 2 are assigned to motion 1 and 6 features corresponding to motion 2 are assigned to motion 3. This is because the motion of the car was smaller and hence its re-projection error is small for all $\hat{F}_i$'s.

## 5.8 Conclusions

We have presented a novel geometric approach for the analysis of dynamic scenes containing multiple rigidly moving objects seen in two perspective views. Our approach exploited the algebraic and geometric properties of the so-called *multibody epipolar constraint* and its associated *multibody fundamental matrix*, which are natural generalizations of the epipolar constraint and of the fundamental matrix to multiple moving objects. We derived a rank constraint on the image points from which one can estimate the number of independent motions and linearly solve for the multibody fundamental matrix. We proved that the epipoles of each independent motion lie exactly in the intersection of the left null space of the multibody fundamental matrix with the so-called Veronese surface. We then showed that epipolar lines can be computed from the derivatives of the multibody epipolar constraint, and individual epipoles can be computed by applying GPCA to the set of epipolar lines. Given the epipoles and epipolar lines, the estimation of individual fundamental matrices becomes a *linear* problem. Then, motion and feature point segmentation is automatically obtained from either the epipoles and epipolar lines or the individual fundamental matrices. We then presented a novel algorithm for optimally segmenting dynamic scenes containing multiple rigidly moving objects in the presence of noise. Instead of iterating between feature segmentation and single body motion estimation, our approach eliminates the segmentation and directly optimizes over the motion parameters.

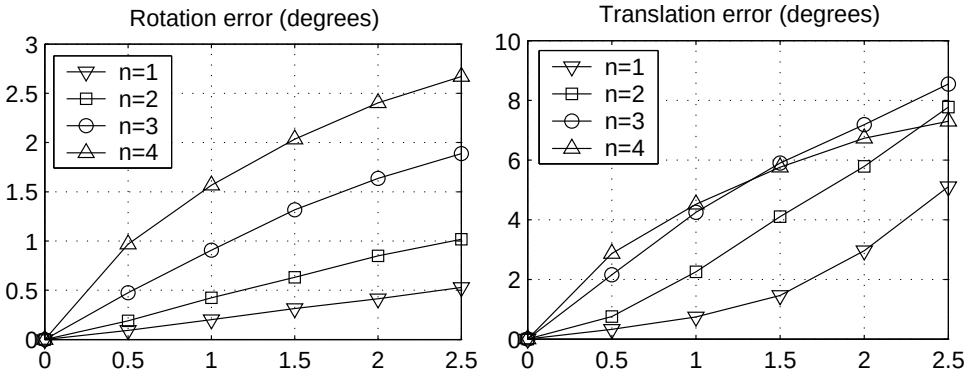
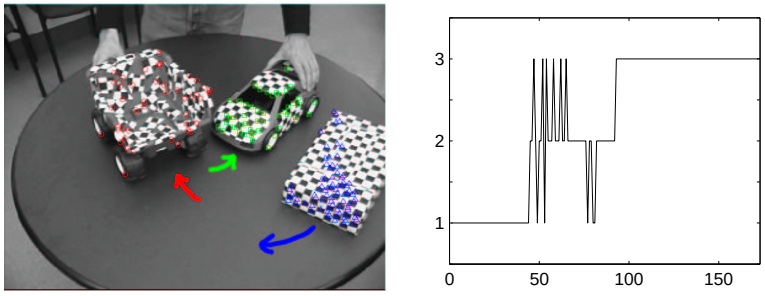
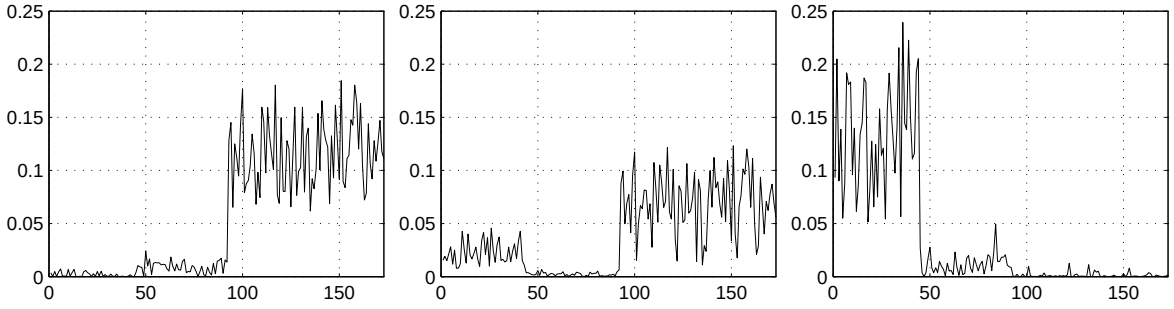



Figure 5.6: Error in the estimation of the rotation and translation as a function of noise in the image points (std. in pixels).



(a) First frame (b) Feature segmentation



(c) $E_O(\hat{F}_1)$ (d) $E_O(\hat{F}_2)$ (e) $E_O(\hat{F}_3)$

Figure 5.7: Segmenting a sequence with 3-D independent rigid motions.

# Chapter 6

# Conclusions

This thesis presented an algebraic geometric approach to simultaneous data segmentation and model estimation, which is applicable to segmentation problems in which the data has a piecewise constant, piecewise linear, or piecewise bilinear structure.

For these three classes of problems, we showed that one can represent the data with a collection of multivariate polynomials of a certain degree. The degree of the polynomials corresponds to the number of groups and the factors of the polynomials encode the model parameters associated with each group. Therefore, we showed that the problem of estimating multiple models from data is mathematically equivalent to factoring multivariate polynomials. We presented a novel solution to such problem based on computing roots of univariate polynomials, plus a combination of linear algebra with polynomial differentiation and division.

The development of our theory was motivated by various problems in computer vision. We illustrated the piecewise constant case with applications to segmentation of static scenes based on different cues such as intensity, texture and motion. We illustrated the piecewise linear case with applications to detection of vanishing points, clustering of faces under varying illumination, and segmentation of dynamic scenes with linearly moving objects. In the case of piecewise bilinear data, we gave a complete solution to the multibody structure from motion problem, i.e., the problem of segmenting dynamic scenes with multiple rigidly moving objects from two perspective views.

We also showed that our algebraic algorithms can be naturally used to initialize iterative approaches to data clustering, such as K-means or EM, and established some basic connections with other machine learning algorithms such as KPCA. Future research will further explore connections with other probabilistic methods, and extend our theory to mixtures of models in which each class has a possibly different algebraic structure.

# Bibliography

[1] S. Avidan and A. Shashua. Trajectory triangulation: 3D reconstruction of moving points from a monocular image sequence. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 22(4):348–357, 2000.

[2] S. Ayer and H. Sawhney. Layered representation of motion video using robust maximum-likelihood estimation of mixture models and MDL encoding. In *IEEE International Conference on Computer Vision*, pages 777–785, 1995.

[3] S. Ayer, P. Schroeter, and J. Bigün. Segmentation of moving objects by robust motion parameter estimation over multiple frames. In *European Conference on Computer Vision*, pages 316–327, 1994.

[4] M. Black and P. Anandan. Robust dynamic motion estimation over time. In *International Conference on Computer Vision and Pattern Recognition*, pages 296–302, 1991.

[5] M. Black and P. Anandan. The robust estimation of multiple motions: Parametric and piecewise-smooth flow fields. *Computer Vision and Image Understanding*, 63(1):75–104, 1996.

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

[7] 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.

[8] A. Chiuso, P. Favaro, H. Jin, and S. Soatto. Motion and structure causally integrated over time. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 24(4):523–535, April 2002.

[9] 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.

[10] J. Costeira and T. Kanade. Multi-body factorization methods for motion analysis. In *IEEE International Conference on Computer Vision*, pages 1071–1076, 1995.

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

[12] T. Darrel and A. Pentland. Robust estimation of a multi-layered motion representation. In *IEEE Workshop on Visual Motion*, pages 173–178, 1991.

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

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

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

[16] X. Feng and P. Perona. Scene segmentation from 3D motion. In *International Conference on Computer Vision and Pattern Recognition*, pages 225–231, 1998.

[17] A. Fitzgibbon and A. Zisserman. Multibody structure and motion: 3D reconstruction of independently moving objects. In *European Conference on Computer Vision*, pages 891–906, 2000.

[18] D. Forsyth and J. Ponce. *Computer Vision: A Modern Approach*. Prentice Hall, 2002.

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

[20] M. Han and T. Kanade. Reconstruction of a scene with multiple linearly moving objects. In *International Conference on Computer Vision and Pattern Recognition*, volume 2, pages 542–549, 2000.

[21] M. Han and T. Kanade. Multiple motion scene reconstruction from uncalibrated views. In *IEEE International Conference on Computer Vision*, volume 1, pages 163–170, 2001.

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

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

[24] A. Heyden and K. Åström. Algebraic properties of multilinear constraints. *Mathematical Methods in Applied Sciences*, 20(13):1135–62, 1997.

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

[26] M. Irani and P. Anandan. About direct methods. In *Workshop on Vision Algorithms*, pages 267–277, 1999.

[27] M. Irani, B. Rousso, and S. Peleg. Detecting and tracking multiple moving objects using temporal integration. In *European Conference on Computer Vision*, pages 282–287, 1992.

[28] A. Jepson and M. Black. Mixture models for optical flow computation. In *International Conference on Computer Vision and Pattern Recognition*, pages 760–761, 1993.

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

[30] M. I. Jordan. *An Introduction to Probabilistic Graphical Models*. 2002. In Preparation.

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

[32] 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.

[33] Q. Ke and T. Kanade. A robust subspace approach to layer extraction. In *IEEE Workshop on Motion and Video Computing*, pages 37–43, 2002.

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

[35] H. C. Longuet-Higgins. A computer algorithm for reconstructing a scene from two projections. *Nature*, 293:133–135, 1981.

[36] Y. Ma, J. Košecká, and S. Sastry. Optimization criteria and geometric algorithms for motion and structure estimation. *International Journal of Computer Vision*, 44(3):219–249, 2001.

[37] M. Machline, L. Zelnik-Manor, and Michal Irani. Multi-body segmentation: Revisiting motion consistency. In *ECCV Workshop on Vision and Modeling of Dynamic Scenes*, 2002.

[38] D. Martin, C. Fowlkes, D. Tal, and J. Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *IEEE International Conference on Computer Vision*, volume 2, pages 416–423, July 2001.

[39] A. Ng, Y. Weiss, and M. Jordan. On spectral clustering: analysis and an algorithm. In *Neural Information Processing Systems*, 2001.

[40] P. Perona and W. Freeman. A factorization approach to grouping. In *European Conference on Computer Vision*, pages 655–670, 1998.

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

[42] G. Scott and H. Longuet-Higgins. Feature grouping by relocalisation of eigenvectors of the proximity matrix. In *British Conference on Machine Vision*, pages 731–737, 1990.

[43] A. Shashua and A. Levin. Multi-frame infinitesimal motion model for the reconstruction of (dynamic) scenes with multiple linearly moving objects. In *IEEE International Conference on Computer Vision*, volume 2, pages 592–599, 2001.

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

[45] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.

[46] 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.

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

[48] S. Soatto and R. Brockett. Optimal and suboptimal structure from motion: local ambiguities and global estimates. In *International Conference on Computer Vision and Pattern Recognition*, pages 282–288, 1998.

[49] S. Soatto and P. Perona. Three dimensional transparent structure segmentation and multiple 3D motion estimation from monocular perspective image sequences. In *IEEE Workshop on Motion of Nonrigid and Articulated Objects, Austin*, pages 228–235, 1994.

[50] A. Spoerri and S. Ullman. The early detection of motion boundaries. In *IEEE International Conference on Computer Vision*, pages 209–218, 1987.

[51] R. Szeliski and S. B. Kang. Recovering 3D shape and motion from image streams using non-linear least squares. *Journal of Visual Communication and Image Representation*, 5(1):10–28, 1994.

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

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

[54] D. Titterington, A. Smith, and U. Makov. *Statistical Analysis of Finite Mixture Distributions*. John Wiley & Sons, 1985.

[55] P. Torr, R. Szeliski, and P. Anandan. An integrated Bayesian approach to layer extraction from image sequences. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 23(3):297–303, 2001.

[56] P. H. S. Torr. Geometric motion segmentation and model selection. *Phil. Trans. Royal Society of London A*, 356(1740):1321–1340, 1998.

[57] R. Vidal, Y. Ma, S. Hsu, and S. Sastry. Optimal motion estimation from the multiview normalized epipolar constraint. In *IEEE International Conference on Computer Vision*, volume 1, pages 34–41, Vancouver, Canada, 2001.

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

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

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

[61] R. Vidal and S. Sastry. Optimal segmentation of dynamic scenes from two perspective views. In *International Conference on Computer Vision and Pattern Recognition*, 2003.

[62] 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.

[63] R. Vidal, S. Soatto, and S. Sastry. A factorization method for multibody motion estimation and segmentation. In *Fortieth Annual Allerton Conference on Communication, Control and Computing*, pages 1625–1634, 2002.

[64] R. Vidal, S. Soatto, and S. Sastry. Two-view segmentation of dynamic scenes from the multibody fundamental matrix. Technical Report UCB/ERL M02/02, UC Berkeley, February 2002.

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

[66] Y. Weiss. A unified mixture framework for motion segmentation: incoprporating spatial coherence and estimating the number of models. In *International Conference on Computer Vision and Pattern Recognition*, pages 321–326, 1996.

[67] Y. Weiss. Smoothness in layers: Motion segmentation using nonparametric mixture estimation. In *International Conference on Computer Vision and Pattern Recognition*, pages 520–526, 1997.

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

[69] J. Weng, N. Ahuja, and T. Huang. Optimal motion and structure estimation. *IEEE Transactions PAMI*, 9(2):137–154, 1993.

[70] L. Wolf and A. Shashua. Two-body segmentation from two perspective views. In *International Conference on Computer Vision and Pattern Recognition*, pages 263–270, 2001.