

Generalized Principal Component Analysis

Estimation & Segmentation of Hybrid Models

René Vidal (JOHNS HOPKINS UNIVERSITY)

Yi Ma (UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN)

S. Shankar Sastry (UNIVERSITY OF CALIFORNIA AT BERKELEY)

November 30, 2005

Copyright ©2004 Reserved

No parts of this draft may be reproduced without written permission from the authors.

+ This is page ii
Printer: Opaque this

To be written (R.V.)

To be written (Y.M.)

To be written (S.S.S.)

+ This is page vi
Printer: Opaque this

Preface

In the past few years, we have encountered a variety of practical problems in computer vision, image processing, pattern recognition, and systems identification, that can be abstracted to be solutions of a common mathematical problem: Given a set of data points sampled from a “mixture” of unknown geometric or statistical models, how to automatically learn or infer these models? The word “mixture” means that the data points are clustered into different groups, each of which belongs to a different model. In the literature, in different contexts, such data sets are sometimes referred to as “mixed,” or “multi-modal,” or “multi-model,” or “heterogeneous,” or “hybrid.” In this book, we have settled on the usage of the expression “mixed data” and the associated model as a “hybrid model.” Thus, a solution to the problem is to *segment* the data into groups, each belonging to one of the constituent models and to then *estimate* the parameters of each model. In the case that each of the constituent models are linear, the foregoing problem is reduced to one of fitting multiple low-dimensional affine subspaces to the set of sample points in a high-dimensional space.

The main goal of this book is to introduce a new method to study hybrid models, which we refer to as *generalized principal component analysis*, with the acronym GPCA.¹ The general problems that GPCA aims to address represents

¹In the literature, the word “generalized” is sometimes used to indicate different extensions to the classical principal component analysis (PCA) [Jolliffe, 1986]. In our opinion these are indeed extensions rather than a *more extensive generalization* that we propose in this book. Additionally, for the “nonlinear” case when each component is an algebraic variety of higher degree such as a quadratic surface or a more complicated manifold, we may still use the same term GPCA. Other names like “hybrid component analysis” (HCA) have also been suggested and would also be appropriate.

a fairly general class of *unsupervised learning* problems — many data clustering and compression methods in machine learning can be viewed as special cases of this method.²

A major difficulty associated with estimation of a hybrid model is that, without knowing which subset of sample points belong to which constituent model it is not possible to determine the model that this group belongs to. Thus, there is seemingly a “*chicken-and-egg*” relationship between data segmentation and model estimation: If the segmentation of the data was known, one could easily fit a single model to each subset of samples using classical model estimation techniques; and conversely, if the models were known, one could easily find the subset of samples that best fit each model. This relationship has been the intuitive and heuristic justification for many iterative multi-model estimation techniques, such as the well-known expectation maximization (EM) algorithm. This book aims to provide a non-iterative and general solution to the problem of simultaneously grouping and model fitting the data, based on a repertoire of new tools drawn from a novel and somewhat unconventional source in the statistics literature: algebraic geometry – mainly polynomial and linear algebra. The immediate reaction of a statistician to this statement is that the methods of classical algebraic geometry are extremely sensitive to noise and modeling uncertainties and are therefore not robust in problems of estimation and model fitting. However, we will show how to combine these new tools with traditional statistical techniques so as to obtain robust simultaneous data segmentation and model estimation algorithms.

There are several reasons why we felt that it was the right time to write a book on estimation of mixed models. First, although the classical model estimation techniques, for a single model, have been well studied in systems theory, signal processing, and pattern recognition, techniques for hybrid models, even for hybrid linear models, has not been well-understood and thoroughly developed. Designing a working algorithm, for many practical problems of this nature is currently a matter of intuition and clever heuristics: almost a work of art from application from application! Of course, as a result it is difficult to abstract the lessons from one context and use it in another.

Second the conventional single-model paradigm has shown to have significant limitations in numerous important emerging applications. For example, in image processing, it is well known that to achieve a more economic (compressed) representation of images, different images or different regions of the same image, it is best to represent either the regions of a single image or a class of images *adaptively* using different sets of linear bases. However, the traditional image processing doctrine advocates the use of a prefixed or prechosen set of bases (for e.g., the discrete cosine transform or a wavelet transform). This approach is largely because segmentation of the image and identification of adaptive bases are difficult problems – they belong to a hybrid model. Yet another example arises

²Classical clustering analysis can be viewed as the specialization of GPCA to the case that each affine subspace is of dimension 0.

from systems theory: classical system identification theory and techniques are of limited use for a system that switches among multiple control systems, called a *hybrid* linear system. New identification theory and algorithms need to be developed for such systems.

Finally in our opinion even though attempts have been made in the past to study and solve many special cases of estimating mixed models, for instance in multi-modal statistics, machine learning, and pattern recognition, there has never been an attempt to unify and truly generalize the results in a unified framework that is able to encompass all aspects of the problem. For instance, both the so called Expectation Maximization or EM method and the K-means method have been proposed to resolve the “chicken-and-egg” difficulty between data segmentation and model estimation. However, these methods resort primarily to an *incremental and iterative* scheme that starts from a random initialization. They are therefore prone to converge to local minima. Only recently has GPCA, a new algebro-geometric approach, been developed that offers a *global and non-iterative* solution to this problem. Not only does this new approach lead to simple and effective algorithms that do not have the same difficulty as the existing methods, but it also offers new insights to modeling data with a hybrid model. This book gives an introduction to such new methods and algorithms.

The primary goal of this book is to introduce the fundamental geometric and statistical concepts and methods associated with modeling and estimation for the hybrid models, especially the hybrid linear models. The topics covered in this book are of relevance and importance to researchers and practitioners in the areas of machine learning, pattern recognition, computer vision, image and signal processing, and systems identification (in many fields including control theory, econometrics, and finance). The applications given in this book highlight our own experiences which are certainly relevant for researchers who work on practical areas such as image segmentation & compression, motion segmentation & estimation from video data, and hybrid system identification. However, even a cursory examination of the literature shows that the number of other application domains is virtually limitless. Although, traditionally, data modeling and model estimation has been primarily a topic of study in statistics, this book requires surprising little by way of background in statistical theory. Our analysis and techniques will be based mainly on linear algebra and polynomial algebra, largely complementary and indeed compatible with existing statistical methods.

Organization of the Book.

Part I of this book develops the fundamental theory and basic algorithms for the identification and estimation of hybrid linear models. The chapters in this part systematically extend classical principal component analysis (PCA) for a single linear subspace, also known as the Karhunen-Loëve (KL) expansion, to the case of a subspace arrangement. We begin after an introduction in Chapter 1, with a review of Principal Component Analysis: from a geometric, statistical and robustness standpoint and its extensions in Chapter 2. In Chapter 3, we start to study the

problem of modeling data with subspace arrangements. The focus is primarily on existing iterative subspace segmentation methods such as K-Means and EM. These methods are based on either geometric intuition or statistical inference. In Chapter 4 we develop a non-iterative algebraic method for subspace segmentation, i.e., the Generalized Principal Component Analysis (GPCA) algorithm. In Chapter 5, we further study important algebraic concepts and properties of subspace arrangements that are behind the GPCA algorithm. Statistical considerations and robustness issues which provide the link between the algebraic techniques and some traditional statistical methods are given in Chapter 6. Nonlinear extensions of the basic GPCA algorithm to arrangements of quadratic surfaces and finally to more general manifolds are investigated in Chapter 7.

Parts II-IV of this book provide a few case studies of real-world problems. The problems studied in Part II are from image processing and computer vision. They include image representation & segmentation (Chapter 8) and 2-D or 3-D motion segmentation (Chapter 9 and 10). Part III consists of the identification of hybrid linear systems (Chapter 11-12). Part IV includes applications in systems biology (Chapter 13). Some of the case studies are straightforward application of the proposed algorithms, while others require certain more elaborate justification and special domain knowledge. We hope that these case studies will inspire the reader to discover new applications for the general concepts and methods introduced in this book.

To make the book self-contained, we have summarized useful notations, concepts and results of algebra and statistics in Appendix A and B, respectively. They may come by handy for readers who are not so familiar with certain subjects involved in the book, especially for the early chapters of the book.

René Vidal, Baltimore, Maryland
Yi Ma, Champaign, Illinois
Shankar Sastry, Berkeley, California
Spring, 2005

Acknowledgments

The seeds of Generalized Principal Component Analysis (GPCA) can be traced back to year 2001 when René and Yi started to work together on a polynomial-based method for solving the motion segmentation problem in computer vision. We soon realized that the important concepts and ideas behind this method can be extended and applied to solving many other problems with a similar nature. That observation has encouraged us to work very actively for the past few years to develop a complete theoretical and algorithmic paradigm for this method. René has summarized in his PhD thesis at Berkeley much of the work as of May 2003. About one year later, on the day after Yi's wedding reception, we decided to formalize our findings with a manuscript and sketched an outline of this book at Cafè Kopi in downtown Champaign.

Following René's PhD thesis and our earlier papers, many of our graduate students have studied and extended GPCA to many new problems in computer vision, image processing, and system identification. We especially thank Jacopo Piazzoli of Johns Hopkins, Kun Huang now at the Biomedical and Informatics Department of Ohio State University, Yang Yang, Shankar Rao, and Andrew Wagner of UIUC. Their research projects have led to many exciting theories, applications, and examples presented in this book.

Contents

Preface	vii
Acknowledgments	xi
1 Introduction	2
1.1 Modeling Data with a Parametric Model	3
1.1.1 The Choice of a Model Class	3
1.1.2 Statistical Models versus Geometric Models	4
1.2 Modeling Mixed Data with a Hybrid Model	6
1.2.1 Examples of Mixed Data Modeling	7
1.2.2 Mathematical Representations of Hybrid Models . . .	11
1.2.3 Hybrid Model Selection for Noisy Data	15
1.3 Bibliographic Notes	16
I Theory, Analysis, and Algorithms	19
2 Data Modeling with a Single Subspace	21
2.1 Principal Component Analysis (PCA)	21
2.1.1 A Geometric Approach to PCA	22
2.1.2 A Statistical View of PCA	24
2.1.3 Determining the Number of Principal Components .	26
2.2 Robustness Issues for PCA	27
2.2.1 Outliers	27

2.2.2	Incomplete Data Points	29
2.3	Extensions to PCA	30
2.3.1	Nonlinear PCA	30
2.3.2	Kernel PCA	32
2.4	Bibliographic Notes	33
3	Iterative Methods for Multiple-Subspace Segmentation	35
3.1	Unsupervised Learning Methods for Data Clustering	35
3.1.1	K-Means	37
3.1.2	Expectation Maximization (EM)	39
3.2	Problem Formulation of Subspace Segmentation	45
3.2.1	Projectivization of Affine Subspaces	46
3.2.2	Subspace Projection and Minimum Representation	47
3.3	Subspace-Segmentation Algorithms	49
3.3.1	K-Subspaces	49
3.3.2	Expectation Maximization for Subspaces	51
3.3.3	Relationships between K-Subspaces and EM	54
3.4	Bibliographic Notes	56
4	Algebraic Methods for Multiple-Subspace Segmentation	57
4.1	Introductory Cases of Subspace Segmentation	58
4.1.1	Segmenting Points on a Line	58
4.1.2	Segmenting Lines on a Plane	61
4.1.3	Segmenting Hyperplanes	63
4.2	Knowing the Number of Subspaces	66
4.2.1	An Introductory Example	66
4.2.2	Fitting Polynomials to Subspaces	68
4.2.3	Subspaces from Polynomial Differentiation	70
4.2.4	Point Selection via Polynomial Division	71
4.2.5	The Basic Generalized PCA Algorithm	75
4.3	Not Knowing the Number of Subspaces	76
4.3.1	Introductory Examples	76
4.3.2	Segmenting Subspaces of Equal Dimension	77
4.3.3	Segmenting Subspaces of Different Dimensions	79
4.3.4	The Recursive GPCA Algorithm	81
4.4	Relationships between GPCA, K-Subspaces, and EM	82
4.5	Bibliographic Notes	83
4.6	Exercises	84
5	Statistical Techniques and Robustness Issues	85
5.1	Discriminant Analysis	86
5.1.1	Fisher Linear Discriminant Analysis (LDA)	87
5.1.2	Fisher Discriminant Analysis for Subspaces	88
5.1.3	Simulation Results	90
5.2	Voting Techniques	94

5.2.1	Stacks of Bases and Counters	94
5.2.2	A Voting Scheme for Subspaces	95
5.2.3	Simulation Results	96
5.3	Model-Selection Methods	97
5.3.1	Minimum Effective Dimension	98
5.3.2	Hilbert Function for Model Selection	101
5.4	Robust Statistical Techniques	105
5.4.1	The Sample Influence Function	106
5.4.2	First Order Approximation of the Sample Influence .	107
5.4.3	Multivariate Trimming	108
5.4.4	Simulation Comparison	109
5.5	Bibliographic Notes	113
II	Applications in Image Processing & Computer Vision	115
6	Image Representation, Segmentation & Classification	117
6.1	Lossy Image Representation	117
6.1.1	A Hybrid Linear Model	120
6.1.2	Multi-Scale Hybrid Linear Models	126
6.1.3	Experiments and Comparisons	130
6.1.4	Limitations	133
6.2	Multi-Scale Hybrid Linear Models in Wavelet Domain	134
6.2.1	Imagery Data Vectors in Wavelet Domain	134
6.2.2	Estimation of Hybrid Linear Models in Wavelet Domain	136
6.2.3	Comparison with Other Lossy Representations	137
6.2.4	Limitations	138
6.3	Image Segmentation	139
6.3.1	Hybrid Linear Models for Image Segmentation	139
6.3.2	Dimension and Size Reduction	142
6.3.3	Experiments	143
6.4	Image Classification	143
6.5	Bibliographic Notes	145
7	2-D Motion Segmentation from Image Partial Derivatives	146
7.1	An Algebraic Approach to Motion Segmentation	147
7.2	The Multibody Brightness Constancy Constraint	148
7.3	Segmentation of 2-D Translational Motion Models	151
7.3.1	The Multibody Optical Flow	151
7.3.2	Computing the 2-D Translational Model Parameters .	151
7.4	Segmentation of 2-D Affine Motion Models	152
7.4.1	The Multibody Affine Matrix	153
7.4.2	Computing the Number of 2-D Affine Motion Models	154
7.4.3	Computing the 2-D Affine Motion Model Parameters	156
7.5	Segmentation of Motions Models of Different Type	157

7.5.1	The Multibody Motion Matrix	158
7.5.2	Computing the Number of 2-D Motion Models	158
7.5.3	Computing the Type of 2-D Motion at Each Pixel	160
7.5.4	Computing the 2-D Motion Model Parameters	161
7.6	Algorithm Summary	162
7.7	Experimental Results	162
7.7.1	Simulation Results	164
7.7.2	2-D Translational Motions	165
7.7.3	2-D Affine Motions	166
7.7.4	2-D Translational and 2-D Affine Motions	166
7.8	Bibliographic Notes	169
7.9	Exercises	173
8	3-D Motion Segmentation from Point Correspondences	175
8.1	The Motion Estimation Problem	176
8.1.1	Rigid-Body Motions and Camera Projection Models	176
8.1.2	The Fundamental Matrix	177
8.1.3	The Homography Matrix	178
8.1.4	The Trifocal Tensor	179
8.2	The Motion Segmentation Problem	180
8.3	Segmentation of Linear Motion Models	181
8.3.1	The Affine Motion Subspaces	181
8.3.2	Segmentation of Motion Affine Subspaces	181
8.4	Segmentation of Bilinear Motion Models	183
8.4.1	Segmentation of Fundamental Matrices	183
8.4.2	Segmentation of Homography Matrices	185
8.5	Segmentation of Trilinear Motion Models	189
8.5.1	The Multibody Trifocal Tensor	191
8.5.2	Segmentation of Trifocal Tensors	193
8.6	Experimental Results	195
8.6.1	Segmentation of Affine Motion Subspaces	195
8.6.2	Segmentation of Fundamental Matrices	198
8.6.3	Segmentation of Homography Matrices	200
8.7	Bibliographical Notes	201
8.8	Exercises	202
9	Dynamical Texture and Video Segmentation	206
III	Extensions to Arrangements of Dynamical Systems and Nonlinear Varieties	207
10	Switched ARX Systems	209
10.1	Problem Statement	211
10.2	Identification of a Single ARX System	212

10.3	Identification of Hybrid ARX Systems	215
10.3.1	The Hybrid Decoupling Polynomial	216
10.3.2	Identifying the Hybrid Decoupling Polynomial	217
10.3.3	Identifying System Parameters and Discrete States	220
10.3.4	The Basic Algorithm and its Extensions	222
10.4	Simulations and Experiments	223
10.4.1	Error in the Estimation of the Model Parameters	224
10.4.2	Error as a Function of the Model Orders	224
10.4.3	Error as a Function of Noise	225
10.4.4	Experimental Results on Test Datasets	226
10.5	Bibliographic Notes	227
11	Switched ARMA Models	229
12	Extensions to Arrangements of Nonlinear Surfaces	230
12.1	Arrangements of Quadratic Surfaces	230
12.1.1	Problem Formulation	231
12.1.2	Properties of the Fitting Polynomials	232
12.1.3	Generalized Principal Surface Analysis (GPSA)	235
12.1.4	Variations to the Basic GPSA Algorithm	238
12.1.5	Experiments on Synthetic Data	240
12.2	Other Nonlinear Extensions	241
12.3	Bibliographic Notes	241
IV	Appendices	243
A	Basic Facts from Algebraic Geometry	245
A.1	Polynomial Ring	245
A.2	Ideals and Algebraic Sets	247
A.3	Algebra and Geometry: Hilbert's Nullstellensatz	249
A.4	Algebraic Sampling Theory	250
A.5	Decomposition of Ideals and Algebraic Sets	253
A.6	Hilbert Function, Polynomial, and Series	254
B	Algebraic Properties of Subspace Arrangements	256
B.1	Ideals of Subspace Arrangements	257
B.2	Subspace Embedding and PL-Generated Ideals	258
B.3	Hilbert Function of Subspace Arrangements	261
B.3.1	Hilbert Function and GPCA	261
B.3.2	Special Cases of Hilbert Functions	264
B.3.3	Computation of Hilbert Function	266
B.4	Bibliographic Notes	268
C	Basic Facts from Mathematical Statistics	270

C.1	Estimation of Parametric Models	270
C.1.1	Uniformly Minimum Variance Unbiased Estimates .	272
C.1.2	Maximum Likelihood Estimates	273
C.1.3	Estimates from a Large Number of Samples	274
C.2	Expectation Maximization	277
C.3	Mixture Models	279
C.3.1	Minimax Estimates	279
C.3.2	Minimum Entropy-Product Estimates	280
C.4	Model Selection Criteria	282
C.5	Robust Statistical Methods	284
C.5.1	Sample Influence Function	284
C.5.2	Multivariate Trimming	284
C.5.3	Random Sample Consensus	284
References		285

Chapter 1

Introduction

The primary goal of this book is to study how to model a data set that consists of multiple subsets with each drawn from a different primitive model. In different contexts, such a data set is sometimes referred to as “mixed,” “multi-modal,” “multi-model,” “piecewise,” “heterogeneous,” or “hybrid.” To unify the terminology, in this book, we will refer to such data as “mixed data” and the model used to fit the data as a “hybrid model.” Thus, a hybrid model typically consists of multiple constituent (primitive) models. Modeling mixed data with a hybrid model implies grouping the data into multiple (disjoint) subsets and fitting each subset with one of the constituent models. In the literature, the words “group,” “cluster,” “decompose,” or “segment” are often used interchangeably. However, in this book, we will use the words “group” or “cluster” primarily for data points,¹ and use the words “decompose” or “segment” for models.²

An ever growing number of problems that arise today in computer vision, image processing, pattern recognition, system identification or system biology requires us to model mixed data. The techniques in this book are most evolved from studying the specific case of modeling data with an arrangement of subspaces,³ also called hybrid (piecewise) linear models, with algorithms that we refer to as

¹For instance, we may say “group (or cluster) the data into multiple subsets,” or “a group (or a cluster) of sample points.”

²For instance, we may say “decompose (or segment) a hybrid model into its constituent primitive models.”

³In this book, we will use interchangeably “a mixture,” “a collection,” “a union,” or “an arrangement” of subspaces or models. But be aware that, in the case of subspaces, the formal terminology in Algebraic Geometry is “an arrangement of subspaces.”

generalized principal component analysis (GPCA), we also discuss the development of techniques for hybrid quadratic models and more general hybrid models. We also discuss extensions to hierarchical and recursive approaches to finding hybrid (sub)models inside an initial grouping of data according to an initial hybrid model. Taken together these techniques provide a powerful set of general techniques which are computationally efficient and statistically robust. In this chapter we give a brief introduction to some basic concepts associated with data modeling along with some motivating examples for modeling mixed data along with a brief account of a few related approaches.

1.1 Modeling Data with a Parametric Model

In scientific studies and engineering practice, one is frequently called upon to infer (or learn) a quantitative model M of a given set of sample points, denoted as $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N\} \subset \mathbb{R}^D$. For instance, Figure 1.1 shows a simple example in which one is given a set of four sample points on a two dimensional plane. Obviously, these points can be fitted perfectly by a (one-dimensional) straight line. The line can then be called a “model” for the given points. The reason for

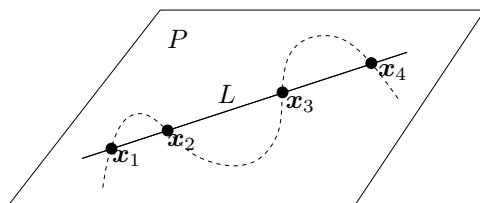


Figure 1.1. Four sample points on a plane are fitted by a straight line. However, they can also be fitted by many other smooth curves, for example the one indicated by the dashed curve.

inferring such a model is because it serves many useful purposes: It can reveal the information encoded in the data or the underlying mechanisms from which the data were generated; Or it may simplify significantly the representation of the given data set or it help to predict effectively future samples.

1.1.1 The Choice of a Model Class

However, inferring the “correct” model of a given data set is an elusive, if not impossible, task. A fundamental difficulty is that, if we are not specific about what we mean by a “correct” model, there could easily be many different models that fit the given data set “equally well.” For instance, for the example shown in Figure 1.1, any smooth curve that passes through the sample points would seem to be an as valid model as the straight line. Furthermore, if there were noise in

the given sample points, then any curve, including the line, passing through the points exactly would unlikely be the “ground truth.”

The question now is: in what sense then can we say a model is correct for a given data set? Firstly, to make the model-inference a well-posed problem, we need to impose additional assumptions or restrictions on the class of models considered. This is to say we should not be looking for any model that can describe the data. Instead, we seek a model M^* that is the *best* among a *restricted* class of models $\mathcal{M} = \{M\}$.⁴ In fact, the well-known No Free Lunch Theorem in statistical learning stipulates that in the absence of prior information or assumptions about the models, there is no reason to prefer one learning algorithm over another [Duda et al., 2000]. Secondly, we need to specify how restricted the class of models need to be. A common strategy is to try to get away with the simplest possible class of models that is *just necessary* to describe the data or solve the problem at hand – known as the principle of Occam’s Razor. More precisely, the model class should be rich enough that it contains at least one model that can fit the data to a desired accuracy and yet be simple enough so as to make the inference of the best model for the given data tractable.

In engineering practice, a popular strategy is to start from the simplest class of models, and only increase the complexity of the models when the simpler models become inadequate. For instance, to fit a set of sample points, one may try first the simplest class of models, namely linear models, then hybrid (piecewise) linear models, followed by (piecewise) quadratic models, and eventually by general topological manifolds. One of the goals of this book is to demonstrate that between them piecewise linear and quadratic models can already achieve an excellent balance between expressiveness and simplicity for many important practical problems.

1.1.2 Statistical Models versus Geometric Models

Roughly speaking, without any training samples whose group membership are known *a priori*, the problem of modeling mixed data falls into the category of *unsupervised learning*. In the literature, almost all unsupervised-learning methods fall into one of two categories. The first category of methods model the data as random samples from a probabilistic distribution and try to learn the distribution from the data. We call such models as statistical models. The second category of methods model the overall geometric shape of the data set as smooth manifolds or topological spaces.⁵ We call such models as geometric models.

⁴Or equivalently, we may impose a non-uniform prior distribution over all models.

⁵Roughly speaking, a smooth manifold is a special topological space that is locally smooth – Euclidean space-like, and has the same dimension everywhere. A general topological space may have singularities and consist of components with different dimensions.

Statistical Learning.

In the statistical paradigm, one assumes that the points \mathbf{x}_i in the data set \mathbf{X} are drawn independently from a common probability distribution $p(\mathbf{x})$, then the task of learning a model from the data becomes one of inferring the most likely probability distribution within a family of distributions of interest (for example Gaussian distributions). Normally the family of distributions is parameterized and denoted as $\mathcal{M} = \{p(\mathbf{x}|\theta) \mid \theta \in \Theta\}$. Consequently, the optimal model $p(\mathbf{x}|\theta^*)$ is given by the maximum likelihood (ML) estimate⁶

$$\theta_{ML}^* \doteq \arg \max_{\theta \in \Theta} \prod_{i=1}^N p(\mathbf{x}_i|\theta). \quad (1.1)$$

If a prior distribution (density) $p(\theta)$ of the parameter θ is also given, then, following the Bayesian rule, the optimal model is given by the maximum a posterior (MAP) estimate

$$\theta_{MAP}^* \doteq \arg \max_{\theta \in \Theta} \prod_{i=1}^N p(\mathbf{x}_i|\theta)p(\theta). \quad (1.2)$$

Many effective methods and algorithms have been developed in the statistics and machine learning literature to infer the optimal distribution $p(\mathbf{x}|\theta^*)$ or a good approximation of it if the exact solution is computationally prohibitive. The estimated probability distribution gives a *generative* description of the samples and can be used to generate new samples or predict the outcomes of new observations.

Geometric Modeling.

However, in many practical scenarios, it is rather difficult to know *a priori* the statistical origins of the data, since we frequently begin with only a few raw sample points, which are insufficient to determine a unique optimal distribution within a large (unrestricted) functional space. Very often, the data points are subject to certain hard geometric constraints, and can only be represented as a singular distribution.⁷ It is very ineffective to learn such a singular distribution [Vapnik, 1995]. Thus, an alternative data-modeling paradigm is to directly learn the overall geometric shape of the given data set \mathbf{X} . Typical methods include fitting one or more geometric primitives such as points⁸, lines, subspaces, surfaces, and manifolds to the data set. For instance, the approach of classical principal component analysis (PCA) is to fit a lower-dimensional subspace, say $S \doteq \text{span}\{\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_d\}$, to a data set in a high-dimensional space, say

⁶If the true distribution from which the data are drawn is $q(\mathbf{x})$, then the maximum likelihood estimate $p(\mathbf{x}|\theta^*)$ minimizes the Kullback-Leibler (KL) divergence: $d(p, q) = \int p \log \frac{p}{q} d\mathbf{x}$ among the given class of distributions.

⁷Mathematically, singular distributions are represented as generalized functions in the Sobolev space. The delta function $\delta(\cdot)$ is one such example.

⁸As the means of multiple clusters.

$\mathbf{X} = \{\mathbf{x}_i\} \subset \mathbb{R}^D$. That is,

$$\mathbf{x}_i = y_{i1}u_1 + y_{i2}u_2 + \cdots + y_{id}u_d + \epsilon_i, \quad \forall \mathbf{x}_i \in \mathbf{X}, \quad (1.3)$$

where $d < D$, $y_{ij} \in \mathbb{R}$, and $u_1, u_2, \dots, u_d \in \mathbb{R}^D$ are unknown parameters and need to be determined – playing the role of the parameters θ in the foregoing statistical model. The line model in Figure 1.1 is an example of PCA for the four points on the plane. In the above equation, the terms $\epsilon_i \in \mathbb{R}^D$ denote possible errors between the samples and the model. PCA minimizes the error $\sum_i \|\epsilon_i\|^2$ for the optimal subspace (see Chapter 2 for details).⁹ In general, a geometric model gives an intuitive description of the samples, and it is often preferred to a statistical one as a “first-cut” description of the given data set. Its main purpose is to capture global geometric, topological, or algebraic characteristics of the data set, such as the number of clusters and their dimensions. A geometric model always gives a more compact representation of the original data set, which makes it useful for data compression and dimension reduction.

As two competing data-modeling paradigms, the statistical modeling techniques in general are more effective in the high-noise (or high-entropy) regime when the generating distribution is (piecewise) smooth and non-singular; and the geometric techniques are more effective in the low-noise (or low-entropy) regime when the underlying geometric space is (piecewise) smooth, at least locally. The two paradigms thus complement each other in many ways. Once the overall geometric shape, the clusters and their dimensions, of the data set are obtained from geometric modeling, one can choose the class of probabilistic distributions more properly for further statistical inference. Since samples normally have noise and sometimes contain outliers or incomplete data, in order to robustly estimate the optimal geometric model, one often resorts to statistical techniques. Therefore, while this book puts more emphasis on geometric or algebraic modeling techniques, we will also thoroughly investigate their connection to and combination with various statistical techniques (see Chapter 5).

1.2 Modeling Mixed Data with a Hybrid Model

However, in practice, many data sets \mathbf{X} cannot be modeled well by any single model M in a pre-chosen or preferred model class \mathcal{M} . Nevertheless, it is often the case that if we *group* such a data set \mathbf{X} into multiple subsets:

$$\mathbf{X} = \mathbf{X}_1 \cup \mathbf{X}_2 \cup \cdots \cup \mathbf{X}_n, \quad (\text{with } \mathbf{X}_l \cap \mathbf{X}_m = \emptyset, \text{ for } l \neq m,) \quad (1.4)$$

then each subset \mathbf{X}_j can be modeled sufficiently well by a *primitive* model in the chosen model class:

$$M_j^* = \arg \min_{M \in \mathcal{M}} [\text{Error}(\mathbf{X}_j, M)], \quad j = 1, 2, \dots, n. \quad (1.5)$$

⁹When the data points \mathbf{x}_i are independent samples drawn from a Gaussian distribution, the geometric formulation of PCA coincides with the classical statistical formulation [Jolliffe, 1986].

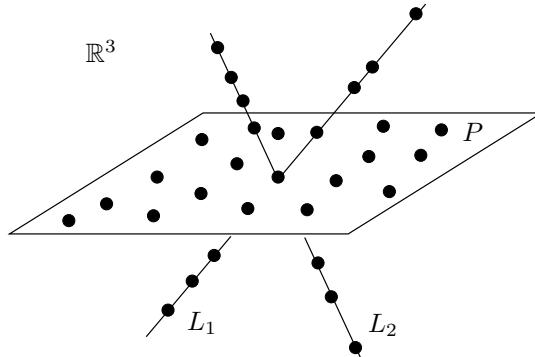


Figure 1.2. A set of sample points in \mathbb{R}^3 are well fitted by a hybrid model with two straight lines and a plane.

Precisely in this sense, we call the data set \mathbf{X} *mixed* (with respect to the chosen model class \mathcal{M}) and call the collection of models $\{M_j^*\}_{j=1}^n$ together a *hybrid model* for \mathbf{X} . For instance, suppose we are given a set of sample points shown in Figure 1.2. These points obviously cannot be fitted well by any single line, plane or smooth surface in \mathbb{R}^3 ; but once they are grouped into three subsets, each subset can be fitted well by a line or a plane (as a standard PCA problem). Note that in this example the topology of the data is indeed “hybrid” – two of the subsets are of dimension one and the other is of dimension two.

1.2.1 Examples of Mixed Data Modeling

In fact, the aforementioned example of mixed data is quite representative of many real data sets that one often encounters in practice. To motivate further the importance of modeling mixed data, we give below a few real-world problems that arise in computer vision and image processing. Most of these problems will be revisited later in this book with more detailed and principled solutions given.

Example One: Vanishing Points in a Perspective Image

The first example is the problem of *vanishing point detection* in computer vision. It is known in computer vision that the perspective images of a group of parallel lines in space all pass through a common point on the image plane which is the so-called vanishing point – a fact already well-known to and extensively exploited by Renaissance artists. Detecting vanishing points is very important for many practical applications such as estimating camera orientation and reconstructing scene structure, especially for man-made environments. A line on the image plane is described as the set of points (x, y) described by an equation $ax + by + c = 0$. For each of the lines passing through the same vanishing point, its coefficient vector $\mathbf{x} = [a, b, c]^T \in \mathbb{R}^3$ must lie on a 2-D subspace, whose normal vector is exactly the vanishing point $\mathbf{v} = [v_x, v_y, 1]^T$, i.e., $\mathbf{v}^T \mathbf{x} = 0$. The vanishing point

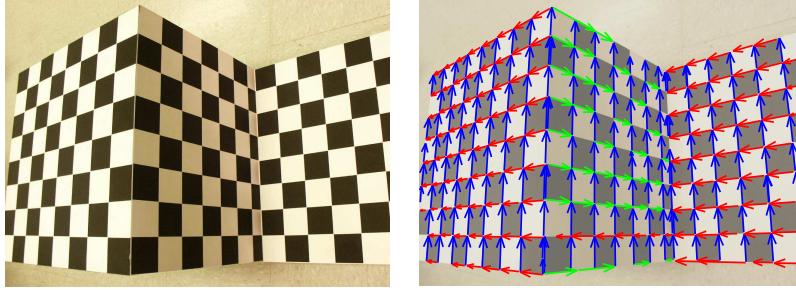


Figure 1.3. Results for clustering edge segments. Each group is marked with a different color. The arrow on each edge segment points to the corresponding vanishing point.

is the point in the plane $(v_x, v_y) \in \mathbb{R}^2$. The extra entry 1 in the vector $v \in \mathbb{R}^3$ may be thought of as representing membership in the plane.

For a scene that consists of multiple sets of parallel lines, as is usually the case for man-made objects and environments, the problem of detecting all the vanishing points from the set of all edge segments is then mathematically equivalent to clustering points into multiple 2-D subspaces in \mathbb{R}^3 . As we will see later that this is a special case of the subspace-segmentation problem addressed by GPCA (see Chapter 4). Figure 1.3 shows the application of the GPCA algorithm to one such example, in which edge segments are correctly grouped to three vanishing points.

Example Two: Motion Segmentation from Two Images

The second example is the so-called *motion segmentation* problem that arises also in the field of computer vision: given a sequence of images of multiple moving objects in a scene, how does one segment the images so that each segment corresponds to only one moving object? This is a very important problem in applications such as motion capture, vision-based navigation, target tracking, and surveillance. If we study the image sequence two images at a time, as it has been known in computer vision, feature points that belong to the same moving object are subject to either linear or quadratic constraints (see Chapter 8), depending on the type of motions and camera models. Therefore, mathematically, the problem of motion segmentation is equivalent to segmentation of points to different linear subspaces and quadratic surfaces. Figure 1.4 shows two images of a moving checker board and cube. The image on the left shows the starting positions of the board and the cube and their directions of motion; and the image on the right shows the final positions. The image on the right also shows the segmentation results obtained using the GPCA algorithm applied to their motion flow for points on the cube and the board. We will describe in detail the motion segmentation method used to achieve the above result in Chapter 8.

Example Three: Image Segmentation and Compression

A third example arises in the context of *image segmentation and compression*. It is commonplace that, in an image, pixels at different regions have significantly

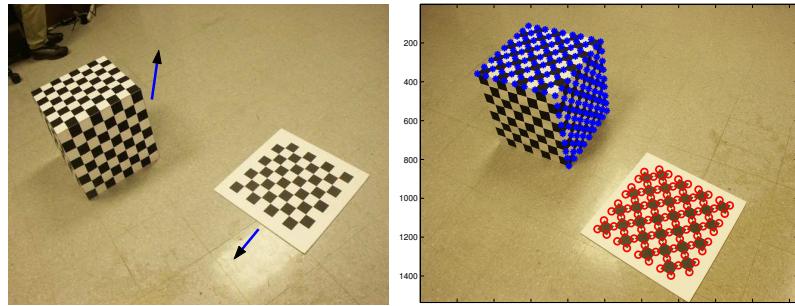


Figure 1.4. Clustering feature points according to different 3-D motions.

different local color/textured profiles (normally an $N \times N$ window around a pixel). Conventional image processing/compression schemes (JPEG, JPEG2000) often ignore such differences and apply the same linear filters or bases (for example., the Fourier transform, discrete cosine transform, wavelets, or curvelets) to the entire set of local profiles. Nevertheless, modeling the set of local profiles as a mixed data set allows us to segment the image into different regions and represent and compress them differently. Each region consists of only those pixels whose local profiles span the same low-dimensional linear subspace.¹⁰ The base vectors of the subspace can be viewed as a bank of adaptive filters for the associated image region. Figure 1.5 shows an example of a segmentation performed using the GPCA algorithm. The so-obtained subspaces (and their bases) normally provide a very

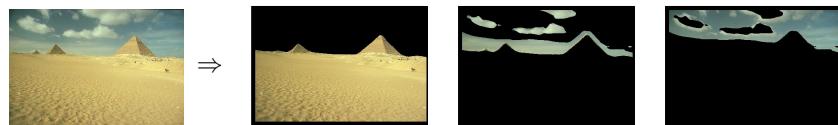


Figure 1.5. Image segmentation based on fitting different linear subspaces (and bases) to regions of different textures. The three segments correspond to the ground, the cloud, and the sky.

compact representation of the image, often more compact than any of the fixed-basis schemes, and therefore very useful for image compression. More details on this image segmentation/compression scheme can be found in Chapter 6.

¹⁰Unlike the previous two examples, there is no rigorous mathematical justification that local profiles from a region of similar texture must span a low-dimensional linear subspace. However, there is strong empirical evidence that a linear subspace is normally a very good approximation.

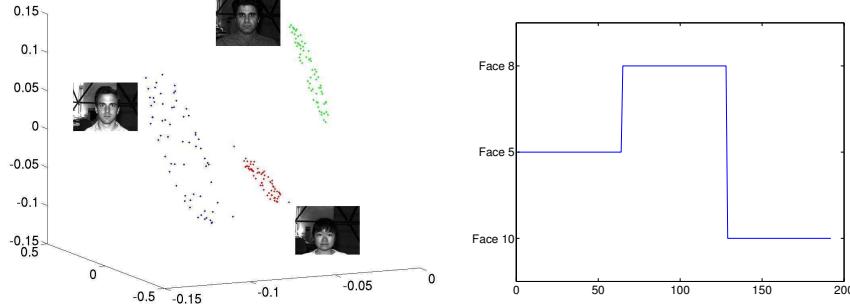


Figure 1.6. Classifying a subset of the Yale Face Database B consisting of 64 frontal views under varying lighting conditions for subjects 2, 5 and 8. Left: Image data projected onto the first three principal components. Right: Classification of the images given by GPCA.

Example Four: Object Classification

The fourth example arises in the context of image-based *object classification*. Given a collection of unlabeled images $\{I_i\}_{i=1}^n$ of several different faces taken under varying illumination, we would like to classify the images corresponding to the face of the same person. For a Lambertian object, it has been shown that the set of all images taken under all lighting conditions forms a cone in the image space, which can be well approximated by a low-dimensional subspace [Ho et al., 2003]. Therefore, we can classify 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. This is obviously another subspace-segmentation problem. In the example shown in Figure 1.6, we use a subset of the Yale Face Database B consisting of $n = 64 \times 3$ frontal views of three faces (subjects 5, 8 and 10) under 64 varying lighting conditions. For computational efficiency, we first down-sample each image to a size of 30×40 pixels. We then project the data onto their first three principal components using PCA, as shown in Figure 1.6 (left).¹¹ We apply GPCA to the projected data in \mathbb{R}^3 and obtain three affine subspaces of dimension 2, 1, and 1, respectively. Despite the series of down-sampling and projection, the subspaces lead to a perfect classification of the images, as shown in Figure 1.6 (right).

Example Five: Video Segmentation and Event Detection

The last example arises in the context of *detecting events from video sequences*. A typical video sequence contains multiple activities or events separated in time. For instance, Figure 1.7 left shows a news sequence where the host is interviewing a guest and the camera is switching between the host, the guest and both of them. Let us assume that all the consecutive frames of the same scene can be modeled as

¹¹The legitimacy of the projection process will be addressed in Chapter 12.

the output from a linear dynamical system and that different scenes correspond to different dynamical systems. Since the image data live in a very high-dimensional space ($\sim 10^5$, the number of pixels), we first project the image data onto a low-dimensional subspace (~ 10) using principal component analysis (PCA) and then apply GPCA to the projected data to identify the different dynamical systems (see Chapter 10). Figure 1.7 shows the segmentation results for two video sequences. In both cases, a perfect segmentation is obtained.

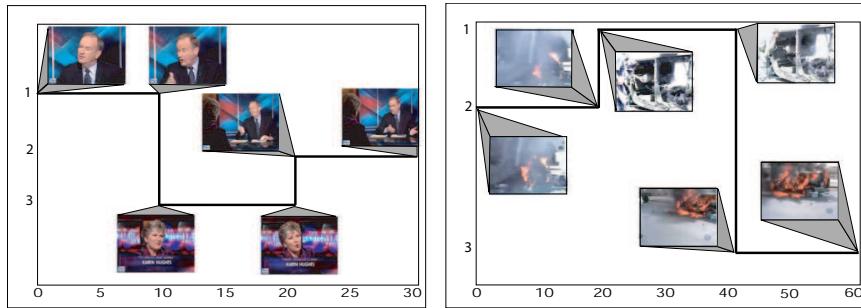


Figure 1.7. Clustering frames of a news video sequence into groups of scenes by modeling each group with a linear dynamical system. Left: 30 frames of a video sequence clustered into 3 groups: host, guest, and both of them. Right: 60 frames of a news video sequence from Iraq clustered into 3 groups: rear of a car with a burning wheel, a burnt car with people, and a burning car.

As we see from the above examples, in some cases, one can rigorously show that a given data set belongs to a collection of linear and quadratic surfaces of the same dimension (example one) or of possibly different dimensions (example two). In many other cases, one can use piecewise linear structures to approximate the data set and obtain a more compact and meaningful geometric representation of the data, including segments, dimensions, and bases (examples three, four, and five). Subspace (or surface) segmentation is a natural abstraction of all these problems and thus merits systematic investigation. From a practical standpoint, the study will lead to many general and powerful modeling tools that are applicable also to many types of data, such as feature points, images, videos, audios, dynamical data, genomic data, proteomic data, and other bio-informatic data sets.

1.2.2 Mathematical Representations of Hybrid Models

Obviously, whether the model associated with a given data set is hybrid or not depends on the class of primitive models considered. In this book, the primitives are normally chosen to be simple classes of smooth manifolds or non-singular distributions. For instance, one may choose the primitive models to be linear subspaces. Then one can use an arrangement of linear subspaces $\{S_i\}_{i=1}^n \subset \mathbb{R}^D$,

$$Z \doteq S_1 \cup S_2 \cup \dots \cup S_n, \quad (1.6)$$

also called a hybrid linear model, to approximate many nonlinear manifolds or piecewise smooth topological spaces. This is the typical model studied by generalized principal component analysis (GPCA). Or as its statistical counterpart, one can assume that the samples points are drawn independently from a mixture of Gaussian distributions $\{p_i(\mathbf{x}), \mathbf{x} \in \mathbb{R}^D\}_{i=1}^n$:

$$q(\mathbf{x}) \doteq \pi_1 p_1(\mathbf{x}) + \pi_2 p_2(\mathbf{x}) + \cdots + \pi_n p_n(\mathbf{x}), \quad (1.7)$$

with $\pi_i > 0$ and $\pi_1 + \pi_2 + \cdots + \pi_n = 1$. This is the typical model studied in mixtures of probabilistic principal component analysis (PPCA) [Tipping and Bishop, 1999a]. In this book, we will study and clarify the similarities and differences between these geometric models and statistical models (see Chapter 3 and 4).

Difficulties with Conventional Data-Modeling Methods.

One may have been wondering why not simply to enlarge the class of primitive models to include such hybrid models so that we can deal with them by the conventional single-model paradigms for learning distribution- or manifold-like models? If this were the case, then there would be no need of developing a theory for hybrid models and thus no need of this book! However, the most compelling reason that we *do* need hybrid models is that smooth manifolds and non-singular distributions *are not rich enough to describe the structure of many commonly observed data*, as we have seen in the examples in the previous section. On one hand, the underlying topological space of a mixed data set may contains multiple manifolds of different dimensions which may probably intersect with each other, as the case with a collection of multiple subspaces. Conventional estimation techniques for manifold-like models do not apply well to this class of spaces. On the other hand, if one represents a hybrid model as a probabilistic distribution, then the distribution will typically not be a (piecewise) smooth function but is singular. Conventional statistical-learning techniques become rather ineffective in inferring such singular distributions [Vapnik, 1995].

An alternative approach to model mixed data is first to segment the data set into coherent subsets and then to model each subset using the classical single-model methods. This is a popular approach adopted by many practitioners in industry. The fundamental difficulty with this approach is that, without knowing which subset of sample points belongs to which constituent model, there is seemingly a “chicken-and-egg” relationship between data segmentation and model estimation: If the segmentation of the data was known, one could fit a model to each subset of samples using classical model estimation techniques; and conversely, if the constituent models were known, one could easily find the subset of samples that best fit each model. This relationship has been the rationale that supports all the *iterative* modeling techniques such as the well-known expectation maximization (EM) algorithm and the K-means method. These iterative methods share several disadvantages:

- The iteration needs to start with a good initial guess of the solution; otherwise the iteration is likely to converge to an irrelevant local minimum.
- Without knowing a-priori the number of models and the dimension of each model, the algorithm may diverge if it starts with a wrong guess on these key parameters.
- There are cases in which it is difficult to solve the grouping problem correctly, yet it is possible to obtain a good estimate of the models. In such cases a direct estimation of the models without grouping seems more appropriate than that based on incorrectly segmented data.

Hybrid Models as Algebraic Sets.

In this book, instead of manifolds or distributions, we will represent hybrid models mainly as algebraic sets.¹² To see the merit of such a representation, let us suppose that data that belong to the i th constituent model can be described as the zero-level set of some polynomials in a prime ideal \mathfrak{p}_i , i.e., an (irreducible) algebraic variety.¹³

$$Z_i \doteq \{\mathbf{x} : p(\mathbf{x}) = 0, p \in \mathfrak{p}_i\} \subset \mathbb{R}^D, \quad i = 1, 2, \dots, n. \quad (1.8)$$

The (mixed) data from a union of n such models then belong to an algebraic set:¹⁴

$$\begin{aligned} Z &\doteq Z_1 \cup Z_2 \cup \dots \cup Z_n \\ &= \{\mathbf{x} : p_1(\mathbf{x})p_2(\mathbf{x}) \dots p_n(\mathbf{x}) = 0, \forall p_i \in \mathfrak{p}_i, i = 1, 2, \dots, n\}. \end{aligned} \quad (1.9)$$

From a number of (random) sample points on the algebraic set $\mathbf{X} \doteq \{\mathbf{x}_j \in Z\}$, one can determine the (radical) ideal of polynomials that vanish on the set Z :¹⁵

$$\mathbf{X} \rightarrow \mathfrak{q}(Z) \doteq \{q(\mathbf{x}_j) = 0, \forall \mathbf{x}_j \in Z\}. \quad (1.10)$$

Obviously, the ideal \mathfrak{q} is no longer a prime ideal. Thus, once the ideal \mathfrak{q} is obtained, the constituent models \mathfrak{p}_i (or Z_i) can be subsequently retrieved by *decomposing* the ideal \mathfrak{q} into irreducible prime ideals:¹⁶

$$\mathfrak{q} \rightarrow \mathfrak{q} = \mathfrak{p}_1 \cap \mathfrak{p}_2 \cap \dots \cap \mathfrak{p}_n. \quad (1.11)$$

¹²Roughly speaking, an algebraic set is the common zero-level set of a family of algebraic equations. For instance, most constraints we encounter in imagery data are given in the form of algebraic equations.

¹³A prime ideal is an ideal that cannot be decomposed further as the intersection of two other ideals. Geometrically, its zero-level set corresponds to an algebraic set that cannot be reduced to multiple algebraic sets. An irreducible algebraic set is called an algebraic variety. A subspace is one such example.

¹⁴Notice that the “union” of algebraic varieties corresponds to the “multiplication” of the polynomials associated with the varieties.

¹⁵According to Hilbert’s Nullstellensatz, there is a one-to-one correspondence between algebraic sets and radical ideals [Eisenbud, 1996].

¹⁶For the special case in which the ideal is generated by a single polynomial, the decomposition is equivalent to factoring the polynomial into irreducible factors.

Clearly, this representation establishes a natural correspondence between common terminologies used in algebraic geometry with the heuristic languages developed in (mixed) data modeling.

Modeling Hybrid Topologies and Degenerate Distributions.

Despite its pure algebraic nature, the above representation is closely related to, as well as complements, the aforementioned two paradigms of data modeling. From the geometric viewpoint, unlike a smooth manifold M which sometimes can be implicitly represented as the level set of a single function, an algebraic set Z is the zero-level set of a family of polynomials. Because of that, algebraic sets Z may have components with different dimensions and singular topologies. From the statistical viewpoint, one can also view the irreducible components $\{Z_i\}$ of Z as the “means” of a collection of probabilistic distributions $\{p_i(\cdot)\}$ and the overall set Z as the “skeleton” of their mixture $q(\cdot)$. For instance, a piecewise linear structure can be viewed as the skeleton of a mixture of Gaussian distributions (see Figure 1.8). Therefore, hybrid models represented by algebraic sets can be interpreted as a special class of *generative models* such that the random variables have very low entropy outside the algebraic sets but high entropy inside.

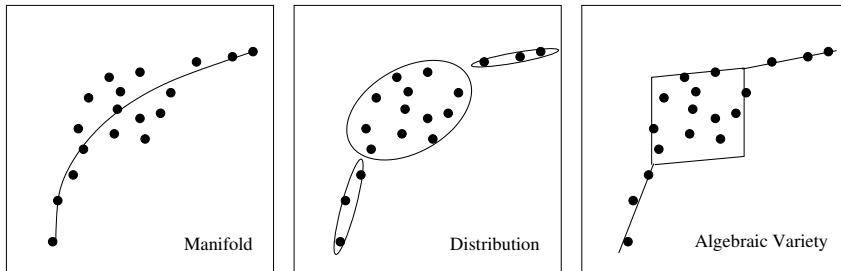


Figure 1.8. Comparison of three representations of the same data set: 1. a (nonlinear) manifold, 2. a (mixed Gaussian) distribution, or 3. a (piecewise linear) algebraic set.

As we will show in this book, if the primitive varieties are simple models such as linear subspaces or quadratic surfaces, then in principle, the problem of segmenting mixed data and estimating a hybrid model can be solved *non-iteratively* (see Chapter 4). As it turns out, the correct number of models and their dimensions can also be correctly determined via purely algebraic means, at least in the noise-free case (see Chapter 12). The algebraic theory of GPCA will be thoroughly developed from Chapter 2 through Chapter 12. The algorithms developed in these chapters are algebraic in their methods. The most common concern that one hears is that algebraic methods are extremely sensitive to noise. We will address this question in considerable detail in this book, but the amazing fact is that even the basic algorithm works extremely well with moderate noise in the data; it can also be applied to high-dimensional data if the data in fact is clustered on fairly low-dimensional structures.

1.2.3 Hybrid Model Selection for Noisy Data

If there is significant noise in the sample points, the problem of finding the “correct” model becomes more challenging. An important observation is that, in the presence of noise, a model is not necessarily the best if it has the highest fidelity to the data. This is especially the case for GPCA since the number of subspaces and their dimensions are not known. In fact, for every point in the data set, one can fit a separate line to it, which results in no modeling error at all. But such a model is not so appealing since it has exactly the same complexity as the original data.

In general, the higher is the model complexity, the smaller the error is.¹⁷ A good model should strike a balance between the complexity of a model M and its fidelity to the data \mathbf{X} .¹⁸ Many general model selection criteria have been proposed in the statistics or machine learning literature, including the minimum description length (MDL), the minimum message length (MML), and the Akaike information criterion (AIC). Despite some small differences, they all tradeoff modeling error for model complexity and minimize an objective of the form:

$$\min_{M \in \mathcal{M}} J(M) \doteq [\alpha \cdot \text{Complexity}(M) + \beta \cdot \text{Error}(\mathbf{X}, M)].$$

In this book, we will introduce a model complexity measure that is specially designed for an arrangement of linear subspaces of different dimensions, namely the *effective dimension* (see Chapter 5).

There is yet another fundamental tradeoff that is often exploited for model selection. When the model complexity is too high, the model tends to over-fit the given data, including the noise in it. Such a model does not generalize well in the sense that it would not predict well the outcome of new samples; when the model complexity is too low, the model under-fits the data and, again, would result in a large error in the prediction. Therefore, a good model should minimize the prediction error. The relationships between modeling error and prediction error as a function of model complexity is plotted in Figure 1.9. Unfortunately, the “optimal” models obtained from trading off modeling error and prediction error can be different, as illustrated in the figure. In such a case, a choice between the two has to be made. In the unsupervised learning setting, it is often difficult to obtain the prediction error curve;¹⁹ and for purposes such as data compression, the prediction error is of less concern than the modeling error. In these cases, we often choose the tradeoff between the modeling error and the model complexity (see Chapter 5).

¹⁷For example, any function can be approximated arbitrarily close by a piecewise linear function with a sufficient number of pieces.

¹⁸For instance, the complexity of a model can be measured as the minimum number of bits needed to fully describe the model and the data fidelity can be measured by the distance from the sample points to the model.

¹⁹Unless one does cross-validation within the given data set itself.

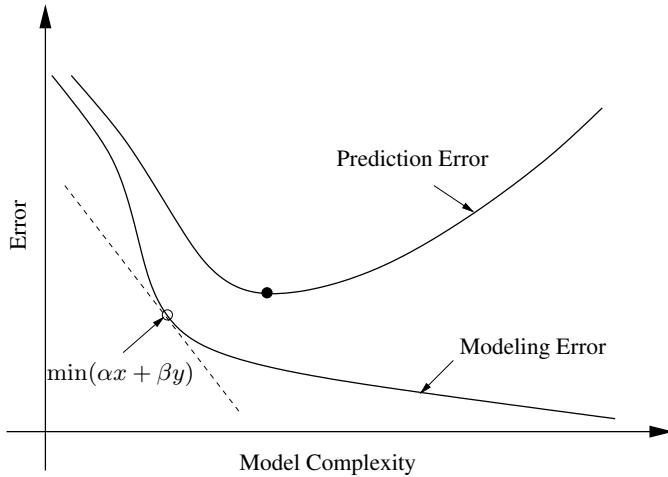


Figure 1.9. Modeling and prediction error versus model complexity.

The algebraic techniques used in the algebraic GPCA algorithm though not very poorly conditioned are by their very nature not tremendously robust to noise.²⁰ Nevertheless, we will show how one can improve the robustness of the algebraic GPCA algorithm to noisy date by incorporating proper model selection criteria such as the one discussed above. Various robust statistical techniques will also be introduced to handle problems with outliers. Finally this robustified GPCA algorithm can be combined with the existing EM and K-means algorithms (for instance for initializing them well) so as to result in much improved robustness, efficiency, and optimality.

1.3 Bibliographic Notes

Yi's notes:

Do we need the historical notes in the introduction or can we put them to the end of each chapter? Do we need to add:

- Applications of PCA in various areas: signal processing, image compression, data analysis in biology, psychometrics...?
- A review of multi-modal statistics, especially its connection to polynomial factorization, and GPCA...?
- The use of mixed models in other areas besides image processing, computer vision, and systems theory...? In particular system biology?

²⁰The techniques include polynomial fitting, factorization, division, and differentiation.

- References on the learning of kernels.
- A review of other related unsupervised learning methods, such as EM, K-means, ICA, multidimensional scaling, local linear embedding, Isomap...?

+ This is page 18
Printer: Opaque this

Part I

Theory, Analysis, and

Algorithms

+ This is page 20
Printer: Opaque this

Chapter 2

Data Modeling with a Single Subspace

In this chapter, we give a brief review of principal component analysis (PCA), i.e., the method for finding a dominant affine subspace to fit a set of data points. The solution to PCA has been well established in the literature and it has become one of the most useful tools for data modeling, compression, and visualization. In this section, we first show that the singular value decomposition (SVD) provides an optimal solution to PCA. Both the geometric and statistical formulation of PCA will be introduced and their equivalence will be established. When the dimension of the subspace is unknown, we introduce some conventional model selection methods to determine the number of principal components. When the samples contain outliers and incomplete data points, we review some robust statistical techniques that help resolve these difficulties. Finally, some nonlinear extensions to PCA such as nonlinear PCA and kernel PCA will also be reviewed.

2.1 Principal Component Analysis (PCA)

Principal component analysis (PCA) refers to the problem of fitting a low-dimensional affine subspace S to a set of points $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N\}$ in a high-dimensional space \mathbb{R}^D , the ambient space. Mathematically, this problem can be formulated as either a statistical problem or a geometric one, and they both lead to the same solution, as we will show in this section.

2.1.1 A Geometric Approach to PCA

We first examine the more intuitive geometric approach to PCA. That is, one tries to find an (affine) subspace that fits the given data points. Let us assume for now that the dimension of the subspace d is known. Then every point \mathbf{x}_i on a d -dimensional affine subspace in \mathbb{R}^D can be represented as

$$\mathbf{x}_i = \mathbf{x}_0 + U_d \mathbf{y}_i, \quad i = 1, \dots, N \quad (2.1)$$

where $\mathbf{x}_0 \in \mathbb{R}^D$ is a(ny) fixed point in the subspace, U_d is a $D \times d$ matrix with d orthonormal column vectors, and $\mathbf{y}_i \in \mathbb{R}^d$ is simply the vector of new coordinates of \mathbf{x}_i in the subspace. Notice that there is some redundancy in the above representation due to the arbitrariness in the choice of \mathbf{x}_0 in the subspace. More precisely, for any $\mathbf{y}_0 \in \mathbb{R}^d$, we can re-represent \mathbf{x}_i as $\mathbf{x}_i = (\mathbf{x}_0 + U_d \mathbf{y}_0) + U_d(\mathbf{y}_i - \mathbf{y}_0)$. Therefore, we need some additional constraints in order to end up with a unique solution to the problem of finding an affine subspace to fit the data. A common constraint is to impose that the mean of \mathbf{y}_i is zero:¹

$$\bar{\mathbf{y}} \doteq \frac{1}{N} \sum_{i=1}^N \mathbf{y}_i = 0. \quad (2.2)$$

In general the given points are imperfect and have noise. We define the “optimal” affine subspace to be the one that minimizes the sum of squared error between \mathbf{x}_i and its projection on the subspace, i.e.,

$$\min_{\mathbf{x}_0, U_d, \{\mathbf{y}_i\}} \sum_{i=1}^N \|\mathbf{x}_i - \mathbf{x}_0 - U_d \mathbf{y}_i\|^2, \quad \text{s.t. } U_d^T U_d = I \text{ and } \bar{\mathbf{y}} = 0. \quad (2.3)$$

Differentiating this function with respect to \mathbf{x}_0 and \mathbf{y}_i (assuming U_d is fixed) and setting the derivatives to be zero,² we obtain the relations:

$$\hat{\mathbf{x}}_0 = \bar{\mathbf{x}} \doteq \frac{1}{N} \sum_{i=1}^N \mathbf{x}_i; \quad \hat{\mathbf{y}}_i = U_d^T (\mathbf{x}_i - \bar{\mathbf{x}}). \quad (2.4)$$

The vector $\hat{\mathbf{y}}_i \in \mathbb{R}^d$ is simply the coordinates of the projection of $\mathbf{x}_i \in \mathbb{R}^D$ in the subspace S . We may call such $\hat{\mathbf{y}}$ the “geometric principal components” of \mathbf{x} .³

Then the original objective becomes one of finding an orthogonal matrix $U_d \in \mathbb{R}^{D \times d}$ that minimizes

$$\min_{U_d} \sum_{i=1}^N \|(\mathbf{x}_i - \bar{\mathbf{x}}) - U_d U_d^T (\mathbf{x}_i - \bar{\mathbf{x}})\|^2. \quad (2.5)$$

¹In the statistical setting, \mathbf{x}_i and \mathbf{y}_i will be samples of two random variables \mathbf{x} and \mathbf{y} , respectively. Then this constraint is equivalent to setting their means to be zero.

²which are the necessary conditions for the minima.

³As we will soon see in the next section, it coincides with the traditional principal components defined in a statistical sense.

Note that this is a restatement of the original problem with the mean $\bar{\mathbf{x}}$ subtracted from each of the sample points. Therefore, from now on, we will consider only the case in which the data points have zero mean. If not, simply subtract the mean from each point and the solution for U_d remains the same. The following theorem gives a constructive solution to the optimal solution \hat{U}_d .

Theorem 2.1 (PCA via SVD). *Let $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N] \in \mathbb{R}^{D \times N}$ be the matrix formed by stacking the (zero-mean) data points as its column vectors. Let $\mathbf{X} = U\Sigma V^T$ be the singular value decomposition (SVD) of the matrix \mathbf{X} . Then for any given $d < D$, a solution to PCA, \hat{U}_d is exactly the first d columns of U ; and $\hat{\mathbf{y}}_i$ is the i th column of the top $d \times N$ submatrix $\Sigma_d V_d^T$ of the matrix ΣV^T .*

Proof. Note that the problem

$$\min_{U_d} \sum_{i=1}^N \|\mathbf{x}_i - U_d U_d^T \mathbf{x}_i\|^2 \quad (2.6)$$

is equivalent to

$$\begin{aligned} & \min_{U_d} \sum_{i=1}^N \text{trace} \left[(\mathbf{x}_i - U_d U_d^T \mathbf{x}_i)(\mathbf{x}_i - U_d U_d^T \mathbf{x}_i)^T \right] \\ \Leftrightarrow & \min_{U_d} \text{trace} \left[(I - U_d U_d^T) \mathbf{X} \mathbf{X}^T \right], \end{aligned}$$

where, for the second equivalence, we use the facts $\text{trace}(AB) = \text{trace}(BA)$, $U_d U_d^T U_d U_d^T = U_d U_d^T$, and $\mathbf{X} \mathbf{X}^T = \sum_{i=1}^N \mathbf{x}_i \mathbf{x}_i^T$ to simplify the expression. Substitute $\mathbf{X} = U \Sigma V^T$ into the above expression, the problem becomes

$$\min_{U_d} \text{trace} \left[(I - U^T U_d U_d^T U) \Sigma^2 \right].$$

Let $\sum_{i=1}^D \sigma_i^2 e_i e_i^T$ be the dyadic decomposition of the diagonal matrix Σ^2 .⁴ Since $U_d^T U$ is an orthogonal matrix, the above minimization is the same as

$$\begin{aligned} & \min_{U_d} \sum_{i=1}^D \text{trace} \left[(\sigma_i e_i - U^T U_d U_d^T U \sigma_i e_i)(\sigma_i e_i - U^T U_d U_d^T U \sigma_i e_i)^T \right] \\ \Leftrightarrow & \min_{U_d} \sum_{i=1}^D \sigma_i^2 \| (I - U^T U_d U_d^T U) e_i \|^2. \end{aligned}$$

Because U_d is an orthogonal matrix of rank d so is $U_d^T U$ so that $I - U^T U_d U_d^T U$ is an idempotent matrix of rank $D - d$, so that the D terms $\| (I - U^T U_d U_d^T U) e_i \|^2$ always sum up to a constant $D - d$, and $\sigma_1^2 \geq \sigma_2^2 \geq \dots \geq \sigma_D^2$ are ordered. Therefore, the minimum is achieved when the d terms associated with the higher weights $\sigma_1^2, \dots, \sigma_d^2$ become zero. This happens only when \hat{U}_d consists of the first d columns of U . The rest of the theorem then easily follows.

⁴Here $e_i \in \mathbb{R}^D$ is the standard i th base vector of \mathbb{R}^D , i.e., its i th entry is 1 and others are 0.

When there are repeated singular values with $\sigma_d = \sigma_{d+1}$, there is a loss of uniqueness of the solution corresponding to the principal components. \square

According to the theorem, the SVD gives an optimal solution to the PCA problem. The resulting matrix \hat{U}_d (together with the mean \bar{x} if the data is not zero-mean) provides a geometric description of the dominant subspace structure for all the points;⁵ and the columns of the matrix $\Sigma_d V_d^T = [\hat{y}_1, \dots, \hat{y}_N] \in \mathbb{R}^{d \times N}$, i.e., the principal components, give a more compact representation for the points $\mathbf{X} = [\mathbf{x}_1, \dots, \mathbf{x}_N] \in \mathbb{R}^{D \times N}$, as d is typically much smaller than D .

2.1.2 A Statistical View of PCA

Historically PCA was first formulated in a statistical setting: to estimate the principal components of a multivariate random variable \mathbf{x} from given sample points $\{\mathbf{x}_i\}$ [Hotelling, 1933]. For a multivariate random variable $\mathbf{x} \in \mathbb{R}^D$ and any $d < D$, the d “principal components” are defined to be d *uncorrelated* linear components of \mathbf{x} :

$$y_i = u_i^T \mathbf{x} \in \mathbb{R}, \quad i = 1, \dots, d \quad (2.7)$$

for some $u_i \in \mathbb{R}^D$ such that the variance of y_i is maximized subject to

$$u_i^T u_i = 1, \quad \text{Var}(y_1) \geq \text{Var}(y_2) \geq \dots \geq \text{Var}(y_d).$$

For example, to find the first principal component, we seek a vector $u_1^* \in \mathbb{R}^D$ such that

$$u_1^* = \arg \max_{u_1 \in \mathbb{R}^D} \text{Var}(u_1^T \mathbf{x}), \quad \text{s.t.} \quad u_1^T u_1 = 1. \quad (2.8)$$

Without loss of generality, we will, in what follows assume \mathbf{x} has zero-mean.

Theorem 2.2 (Principal Components of a Random Variable). *The first d principal components of a multivariate random variable \mathbf{x} are given by the d leading eigenvectors of its covariance matrix $\Sigma_{\mathbf{x}} \doteq E[\mathbf{x}\mathbf{x}^T]$.*

Proof. Notice that for any $u \in \mathbb{R}^D$,

$$\text{Var}(u^T \mathbf{x}) = E[(u^T \mathbf{x})^2] = E[u^T \mathbf{x} \mathbf{x}^T u] = u^T \Sigma_{\mathbf{x}} u.$$

Then to find the first principal component, the above minimization (2.8) is equivalent to

$$\max_{u_1 \in \mathbb{R}^D} u_1^T \Sigma_{\mathbf{x}} u_1, \quad \text{s.t.} \quad u_1^T u_1 = 1. \quad (2.9)$$

Solving the above constrained minimization problem using s Lagrange multiplier method, we obtain the necessary condition for u_1 to be an extrema:

$$\Sigma_{\mathbf{x}} u_1 = \lambda u_1 \quad (2.10)$$

⁵From a statistical standpoint, the column vectors of U_d give the directions in which the data X has the largest variance, hence the name “principal components.”

for some Lagrange multiplier $\lambda \in \mathbb{R}$, and the the associated extremum value is $u_1^T \Sigma_{\mathbf{x}} u_1 = \lambda$. Obviously, the optimal solution u_1^* is exactly the eigenvector associated with the largest eigenvalue of $\Sigma_{\mathbf{x}}$.

To find the remaining principal components, since $u_1^T \mathbf{x}$ and $u_i^T \mathbf{x}$ ($i > 1$) need to be uncorrelated, we have

$$E[(u_1^T \mathbf{x})(u_i^T \mathbf{x})] = E[u_1^T \mathbf{x} \mathbf{x}^T u_i] = u_1^T \Sigma_{\mathbf{x}} u_i = \lambda_1 u_1^T u_i = 0.$$

That is, u_2, \dots, u_d are all orthogonal to u_1 . Following the proof for the optimality of u_1 , u_2 is then the leading eigenvector of $\Sigma_{\mathbf{x}}$ restricted to the orthogonal complement of u_1 .⁶ Overall, u_2 is the second leading eigenvector of $\Sigma_{\mathbf{x}}$. Inductively, one can show for the rest of the principal components. \square

Normally, we do not know $\Sigma_{\mathbf{x}}$ and can only estimate it from the given N samples \mathbf{x}_i . It is known from statistics that

$$\hat{\Sigma}_{\mathbf{x}} \doteq \frac{1}{N} \sum_{i=1}^N \mathbf{x}_i \mathbf{x}_i^T = \frac{1}{N} \mathbf{X} \mathbf{X}^T \quad (2.11)$$

is an asymptotically unbiased estimate of the covariance matrix $\Sigma_{\mathbf{x}}$. The eigenvectors of $\hat{\Sigma}_{\mathbf{x}}$, or equivalently those of $\mathbf{X} \mathbf{X}^T$, lead to the “sample principal components”:

$$\hat{y}_i = \hat{u}_i^T \mathbf{x}, \quad \text{s.t. } \hat{\Sigma}_{\mathbf{x}} \hat{u}_i = \lambda \hat{u}_i \text{ and } \hat{u}_i^T \hat{u}_i = 1. \quad (2.12)$$

One can show that, if \mathbf{x} is Gaussian, then every eigenvector u of $\hat{\Sigma}_{\mathbf{x}}$ is an asymptotically unbiased estimate for the corresponding eigenvector of $\Sigma_{\mathbf{x}}$ [Jolliffe, 1986].

Theorem 2.3 (Equivalence of Geometric and Sample Principal Components). *Let $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N] \in \mathbb{R}^{D \times N}$ be the data matrix (with $\bar{\mathbf{x}} = 0$). The vectors $\hat{u}_1, \hat{u}_2, \dots, \hat{u}_d \in \mathbb{R}^D$ associated with the d sample principal components for \mathbf{X} are exactly the columns of the matrix $\hat{U}_d \in \mathbb{R}^{D \times d}$ that minimizes the least-squares error (2.6).*

Proof. The proof is simple. Notice that if \mathbf{X} has the singular value decomposition $\mathbf{X} = U \Sigma V^T$, then $\mathbf{X} \mathbf{X}^T = U \Sigma^2 U^T$ is the eigenvalue decomposition of \mathbf{X} . If Σ is ordered, then the first d columns of U are exactly the leading d eigenvectors of $\mathbf{X} \mathbf{X}^T$, which give the d sample principal components. \square

Therefore, both the geometric and statistical formulation of PCA lead to exactly the same solutions/estimates of the principal components. The geometric formulation allows us to apply PCA to data even if the statistical nature of the data is unclear; the statistical formulation allows to quantitatively evaluate the quality of the estimates. For instance, for Gaussian random variables, one can derive explicit formulae for the mean and covariance of the estimated principal components. For

⁶The reason for this is that both u_1 and its orthogonal complement u_1^\perp are invariant subspaces of $\Sigma_{\mathbf{x}}$.

a more thorough analysis of the statistical properties of PCA, we refer the reader to the classical book [Jolliffe, 1986].

2.1.3 Determining the Number of Principal Components

Notice that SVD of the noisy data matrix \mathbf{X} does not only give a solution to PCA for a particular d , but also the solutions to all $d = 1, 2, \dots, D$. This has an important side-benefit: if the dimension d of the subspace S , or equivalently the rank of the matrix \mathbf{X} , is *not* known or specified a priori, one may have to look at the entire spectrum of solutions to decide on the “best” estimate \hat{d} for the dimension and hence the subspace S for the given data.

As we have discussed in the introduction of the book, the conventional wisdom is to strike a good balance between the complexity of the chosen model and the data fidelity (to the model). The dimension d of the subspace S can be viewed as a natural measure of the complexity of the model; and the sum of squares of the remaining singular values $\sum_{i=d+1}^D \sigma_i^2$ is exactly the modeling error $\sum_{i=1}^N \|\mathbf{x}_i - \hat{\mathbf{x}}_i\|^2$. The leading singular value σ_{d+1}^2 of the remaining ones is a good index of the modeling error. Therefore, one can seek for a model that balances between d and σ_{d+1}^2 by minimizing an objective function of the form:

$$J_{PCA}(d) \doteq \alpha \cdot \sigma_{d+1}^2 + \beta \cdot d \quad (2.13)$$

for some proper positive weights $\alpha, \beta > 0$. Another somewhat similar and popular objective function that people often use to determine the rank d of the noise matrix \mathbf{X} from its singular values is (e.g., [Kanatani and Matsunaga, 2002a]):

$$J_{rank}(d) \doteq \frac{\sigma_{d+1}^2}{\sum_{i=1}^d \sigma_i^2} + \kappa d. \quad (2.14)$$

In this book, unless stated otherwise, this will be the criterion of choice when we try to determine the rank of a matrix corrupted by noise.

In general, the ordered singular values of the data matrix \mathbf{X} versus the dimension d of the subspace resemble a plot as in Figure 2.1. In the statistics literature, this is known as the “Scree graph.” We will see a significant drop in the singular value right after the “correct” dimension \hat{d} , which is sometimes called the “knee” or “elbow” point of the plot. Obviously, such a point is a stable minimum as it optimizes the above objective function (2.13) for a range of values for α and β , or (2.14) for a range of κ .

A model can also be selected from the Scree graph in another way. If, instead of the dimension d , a tolerance τ for the modeling error is specified, one can easily use the plot to identify the model that has the lowest dimension and satisfies the given tolerance, as indicated in the figure.

There are many other methods for determining the dimension for PCA. Interested readers may find more references in [Jolliffe, 1986].

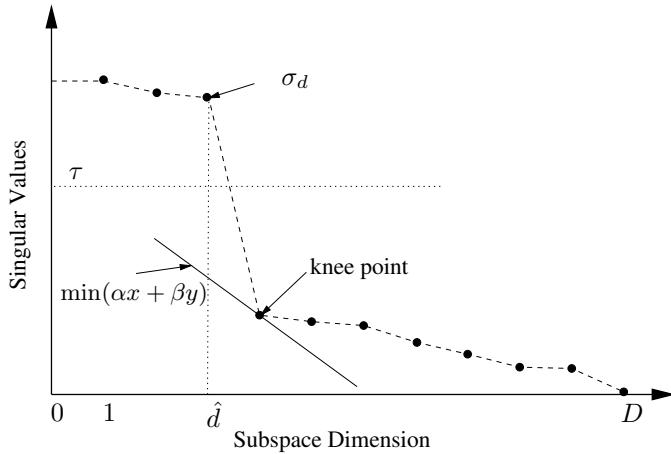


Figure 2.1. Singular value as a function of the dimension of the subspace.

2.2 Robustness Issues for PCA

In the above discussions, we have assumed that all the sample points can be fit with the same geometric or statistical model. In this section, we discuss various robustness issues for PCA. More specifically, we study how to resolve the difficulties with outliers and incomplete data points.

2.2.1 Outliers

In practice, it is often the case that a small subset of the data points do not fit well the same model as the rest of the data. Such points are called *outliers*. The true nature of outliers can be very elusive. There is really no unanimous definition for what is an outlier. Outliers can be perfectly valid samples from the same distribution as the rest of the data and it just happens so that they are *small-probability* instances; or they are not samples drawn from the same model at all and therefore they will likely *not be consistent* with the model derived from the rest of the data; or they are atypical samples that have an unusually *large influence* on the estimated model parameters. In principle however, there is no way that one can really tell which case is really true for a certain “outlier” sample point. But these different interpretations of outliers lead to different approaches to “detect” (and subsequently eliminate) outliers.

Error-Based Outlier Detection

The first approach is to first fit a model to *all* the sample points, including potential outliers, and then detect the outliers as the ones that, with respect to the identified model, correspond to small-probability events or result in too large modeling errors. In PCA, if we assume the samples are drawn from a Gaussian distribution, the probability of a sample is approximately inversely proportional to its (squared)

geometric distance to the identified subspace. Hence using either the probability or the geometric error ends up similar criteria for outlier detection. For instance, from the SVD $\mathbf{X} = U\Sigma V^T$, we know

$$y_i = u_i^T \mathbf{x}, \quad i = 1, \dots, D \quad (2.15)$$

are the new coordinates of each point \mathbf{x} with respect to a new orthonormal coordinate frame specified by U . The first d terms y_1, \dots, y_d are the principal components (the new coordinates of $\hat{\mathbf{x}}$ in the principal subspace); and the last $D - d$ terms y_{d+1}, \dots, y_D are the new coordinates of $\mathbf{x} - \hat{\mathbf{x}}$ in the orthogonal complement of the principal subspace. Clearly the geometric modeling error is

$$\|\mathbf{x} - \hat{\mathbf{x}}\|^2 = y_{d+1}^2 + y_{d+2}^2 + \dots + y_D^2. \quad (2.16)$$

Then one may simply call a point \mathbf{x} an “outlier” if

$$C(\mathbf{x}) \doteq y_{d+1}^2 + y_{d+2}^2 + \dots + y_D^2 \geq \tau \quad (2.17)$$

for some chosen threshold $\tau > 0$. However, this criterion does not take into account how the rest of the points are distributed away from the principal subspace. To fix this oversight, notice that as a random variable, y_i has the (estimated) variance σ_i^2 . Therefore, if we assume $\mathbf{x} - \hat{\mathbf{x}}$ has a Gaussian distribution (and so are y_i), then we have

$$-\log p(\mathbf{x} - \hat{\mathbf{x}}) \propto y_{d+1}^2/\sigma_{d+1}^2 + y_{d+2}^2/\sigma_{d+2}^2 + \dots + y_D^2/\sigma_D^2. \quad (2.18)$$

Therefore, we should use the following criterion

$$C_n(\mathbf{x}) \doteq y_{d+1}^2/\sigma_{d+1}^2 + y_{d+2}^2/\sigma_{d+2}^2 + \dots + y_D^2/\sigma_D^2 \geq \tau \quad (2.19)$$

to detect outliers. The left hand is nothing but the geometric error *normalized* by the respective variance of \mathbf{x} in the direction of each eigenvector. When the variances $\sigma_{d+1}^2, \dots, \sigma_D^2$ are approximately the same, the two criteria (2.17) and (2.19) coincide. In practice, we find either criterion (2.17) or (2.19) can be very effective in detecting outliers, depending on the nature of the data.

Consensus-Based Outlier Detection

The second approach assumes that the outliers are not drawn from the same model as the rest of the data. Hence it makes sense to try to avoid the outliers when we infer a model from the data. However, without knowing which points are outliers beforehand, how can we avoid them? One idea is to fit a model, instead of to all the data points at the same time, only to a *proper subset* of the data. This is possible when the number of data points required for a unique solution for model estimation can be *much* smaller than that of the given data set. Of course, one should *not* expect that a randomly chosen subset will have no outliers and always lead to a correct model. Thus, one should try on *many different subsets*:

$$\mathbf{X}_1, \mathbf{X}_2, \dots, \mathbf{X}_n \subset \mathbf{X}. \quad (2.20)$$

The rationale is that if the percentage of outliers is relatively small, one of the trial subsets, say \mathbf{X}_i , likely contains few or no outliers and hence the resulting

model would be the most consistent with the rest of the data points. For instance, for PCA we may claim a subset \mathbf{X}_i gives a consistent estimate of the subspace $\hat{U}_d(\mathbf{X}_i)$ if the following criterion is satisfied:

$$\#\{\mathbf{x} \in \mathbf{X} : \|\mathbf{x} - \hat{U}_d(\mathbf{X}_i)\| \leq \tau\} \geq N \cdot \delta\%, \quad (2.21)$$

for some error threshold $\tau > 0$ and some percentage threshold δ (normally larger than 50 percent). This scheme is typically called *Random Sample Consensus* (RANSAC), and it normally improves the robustness of the model selection/estimation to outliers. As a word of caution, in practice, in order to design a successful RANSAC algorithm, one needs to carefully choose a few key parameters: the size of each subset, the number of subsets, and the consensus criterion.⁷ There is a vast amount of literature on RANSAC-type algorithms, and for a more thorough introduction, we refer interested readers to [Fischler and Bolles, 1981].

Influence-Based Outlier Detection

The third approach relies on the assumption that an outlier is an atypical sample which has an unusually large influence on the estimated model parameters. This leads to an outlier detection scheme which to some extent combines the characteristics of the previous two approaches: it determines the influence of a sample by comparing the difference between the model estimated with and without this sample. For instance, for PCA one may use a *sample influence function* to measure the difference:

$$I(\mathbf{x}_i, U_d) \doteq d(\hat{U}_d, \hat{U}_{d(i)}), \quad (2.22)$$

where $d(\cdot, \cdot)$ is a proper distance measure between the subspace basis estimated with the i th sample \hat{U}_d and that without the i th sample $\hat{U}_{d(i)}$.⁸ The larger the difference is, the larger is the influence and more likely is the sample \mathbf{x}_i an outlier. Thus, we may eliminate a sample \mathbf{x} as an outlier if

$$I(\mathbf{x}, U_d) \geq \tau \quad (2.23)$$

for some threshold $\tau > 0$. However, this method does not come without an extra cost. We need to compute the principal components for N times: one time for all the samples together and another $N - 1$ times with one sample eliminated from each time. There is also a vast amount of literature on sample influence of PCA, we refer interested readers to [Jolliffe, 2002].

2.2.2 Incomplete Data Points

Another issue that we often encounter in practice is that some of the given data points are “incomplete.” For an incomplete data point $\mathbf{x} = [x_1, x_2, \dots, x_D]^T$, we

⁷That is, the criterion that verifies whether each point is consistent with the model derived from the subset.

⁸One can choose either the largest subspace angle between the two bases or the sum of squares of all the subspace angles.

mean that some of its entry or entries are missing or unspecified. For instance, if the x_i -entry is missing from \mathbf{x} , it means that we know \mathbf{x} only up to a line in \mathbb{R}^D :

$$\mathbf{x} \in L \doteq \{[x_1, \dots, x_{i-1}, t, x_{i+1}, \dots, x_D]^T, t \in \mathbb{R}\}. \quad (2.24)$$

One should be aware that an incomplete data point is in nature rather different from a noisy data point or an outlier.⁹ In general, such incomplete data points can contain useful information about the model, and in the case of PCA, the principal subspace. For instance, if the principal subspace happens to contain the line L , then knowing enough number of such lines, the principal subspace can be uniquely determined. In general, the line L may or may not lie in the principal subspace. We therefore should handle incomplete data points with more care. A key observation here is that the incomplete data point \mathbf{x} is just as good as any point on the line L . Hence it is natural to choose a representative $\hat{\mathbf{x}} \in L$ that is the closest to the principal subspace. Let us denote $B_d \doteq I - U_d U_d^T$, then the closest point $\mathbf{x}^* = [x_1, \dots, x_{i-1}, t^*, x_{i+1}, \dots, x_D]^T$ on L to the principal subspace can be found by minimizing the following quadratic function in t :

$$t^* = \arg \min_t (\mathbf{x}^T B^T B \mathbf{x}). \quad (2.25)$$

This problem has a unique solution as long as the line L is not parallel to the principal subspace, i.e., $e_i \notin \text{span}(U_d)$.

In essence, the above process of finding \mathbf{x}^* on the principal subspace is to give a rank- d approximation of the entire data set containing both complete and incomplete data points. Mathematically, PCA with incomplete data is equivalent to find a rank- d approximation/factorization of the data matrix \mathbf{X} with incomplete data entries (in a least-squares sense). In numerical linear algebra, *power factorization* is especially designed to solve this problem. We refer the interested readers to [Vidal and Hartley, 2004].

2.3 Extensions to PCA

Although PCA offers a rather useful tool to model the linear structure of a given data set, it however becomes less effective when the data actually has some significant nonlinearity, e.g., belonging to some nonlinear manifold. In this section, we introduce some basic extensions to PCA which can, to some extent, handle the difficulty with nonlinearity.

2.3.1 Nonlinear PCA

For nonlinear data, the basic rationale is not to apply PCA directly to the given data, but rather to a transformed version of the data. More precisely, we seek a

⁹One can view incomplete data points as a very special type of noisy data points which have infinite uncertainty only in certain directions.

nonlinear transformation (more precisely, usually an embedding):

$$\begin{aligned}\phi(\cdot) : \mathbb{R}^D &\rightarrow \mathbb{R}^M, \\ \mathbf{x} &\mapsto \phi(\mathbf{x}),\end{aligned}$$

such that the structure of the resulting data $\{\phi(\mathbf{x}_i)\}$ becomes (significantly more) linear. In machine learning, $\phi(\mathbf{x})$ is called the “feature” of the data point \mathbf{x} , and \mathbb{R}^M is called the “feature space.”

Define the matrix $\Phi \doteq [\phi(\mathbf{x}_1), \dots, \phi(\mathbf{x}_N)] \in \mathbb{R}^{M \times N}$. The principal components in the feature space are given by the eigenvectors of the sample covariance matrix¹⁰

$$\Sigma_{\phi(\mathbf{x})} \doteq \frac{1}{N} \sum_{i=1}^N \phi(\mathbf{x}_i) \phi(\mathbf{x}_i)^T = \frac{1}{N} \Phi \Phi^T \in \mathbb{R}^{M \times M}.$$

Let $v_i \in \mathbb{R}^M$ to be the eigenvectors:

$$\Sigma_{\phi(\mathbf{x})} v_i = \lambda_i v_i, \quad i = 1, \dots, M. \quad (2.26)$$

Then the d “nonlinear principal components” of every data point \mathbf{x} are given by

$$y_i \doteq v_i^T \phi(\mathbf{x}) \in \mathbb{R}, \quad i = 1, \dots, d. \quad (2.27)$$

In general, we do not expect that the map $\phi(\cdot)$ is given together with the data. In many cases, searching for the proper map is a difficult task, and the use of nonlinear PCA is therefore limited. However, in some practical applications, good candidates for the map $\phi(\cdot)$ can be found from the nature of the problem. In such cases, the map, together with PCA, can be very effective in extracting the overall geometric structure of the data.

Example 2.4 (Veronese Map for Mixtures of Subspaces). As we will see later in this book, if the data points belong to multiple subspaces, then a natural choice of the transformation $\phi(\cdot)$ is the Veronese map:

$$\begin{aligned}\nu_n(\cdot) : \mathbf{x} &\mapsto \nu_n(\mathbf{x}), \\ (x_1, \dots, x_D) &\mapsto (x_1^n, x_1^{n-1} x_2, \dots, x_D^n)\end{aligned}$$

where the monomials are ordered in the degree-lexicographic order. Under such a mapping, the multiple low-dimensional subspaces are mapped into a single subspace in the feature space, which can then be identified via PCA for the features. ■

NLPCA in a High-dimensional Feature Space.

There is a potential difficulty associated with nonlinear PCA. The dimension of the feature space, depending on the map $\phi(\cdot)$, can be very high and it may be computationally prohibitive to compute the principal components in the feature

¹⁰In principle, we should use the notation $\hat{\Sigma}_{\phi(\mathbf{x})}$ to indicate that it is the estimate of the actual covariance matrix. But for simplicity, we will drop the hat in the sequel and simply use $\Sigma_{\phi(\mathbf{x})}$. The same goes for the eigenvectors and the principal components.

space. For instance, if we try to search for a Veronese map of the proper degree n , the dimension of the feature space M grows exponentially with the degree. When M exceeds N , the eigenvalue decomposition of $\Phi\Phi^T \in \mathbb{R}^{M \times M}$ becomes more costly than that of $\Phi^T\Phi \in \mathbb{R}^{N \times N}$, although the two matrices have the same eigenvalues.

This motivates us to examine whether computation of PCA in the feature space can be reduced to computation with the lower-dimensional matrix $\Phi^T\Phi$. The answer is actually yes. The key is to notice that, despite the dimension of the feature space, every eigenvector $v \in \mathbb{R}^M$ of $\Phi\Phi^T$ associated with a non-zero eigenvalue is always in the span of the matrix Φ :¹¹

$$\Phi\Phi^T v = \lambda v \Leftrightarrow v = \Phi(\lambda^{-1}\Phi^T v) \in \text{range}(\Phi). \quad (2.28)$$

We define the vector $w \doteq \lambda^{-1}\Phi^T v \in \mathbb{R}^N$. Obviously $\|w\|^2 = \lambda^{-1}$. It is straightforward to check that w is an eigenvector of $\Phi^T\Phi$ for the same eigenvalue λ . Once such a w is computed from $\Phi^T\Phi$, we can recover the corresponding v in the feature space as:

$$v = \Phi w. \quad (2.29)$$

Therefore the i th nonlinear principal component of \mathbf{x} under the map $\phi(\cdot)$ can be computed as:

$$y_i \doteq v_i^T \phi(\mathbf{x}) = w_i^T \Phi^T \phi(\mathbf{x}) \in \mathbb{R}, \quad (2.30)$$

where $w_i \in \mathbb{R}^M$ is the i th leading eigenvector of $\Phi^T\Phi$.

2.3.2 Kernel PCA

One should notice a very interesting feature about the above NLPCA method. Entries of both the matrix $\Phi^T\Phi$ and the vector $\Phi^T\phi(\mathbf{x})$ (in the expression for y_i) are all inner products of two features, i.e., of the form $\phi(\mathbf{x})^T\phi(\mathbf{y})$. In other words, computation of the principal components involves only inner products of the features. In the machine learning literature, one defines the ‘‘kernel function’’ of two vectors $\mathbf{x}, \mathbf{y} \in \mathbb{R}^D$ to be the inner product of their features

$$k(\mathbf{x}, \mathbf{y}) \doteq \phi(\mathbf{x})^T\phi(\mathbf{y}) \in \mathbb{R}. \quad (2.31)$$

The so-defined function $k(\cdot, \cdot)$ is a symmetric semi-positive definite function in \mathbf{x} and \mathbf{y} .¹² The entries of the matrix $\Phi^T\Phi$ are nothing but $k(\mathbf{x}_i, \mathbf{x}_j)$.

As a consequence of our discussion above, one can perform nonlinear principal component analysis as long as a (semi-positive definite) kernel function is given. One does not have to explicitly define and evaluate the map $\phi(\cdot)$. In fact, given any (positive-definite) kernel function, according to a fundamental result in functional analysis, one can in principle decompose the kernel and recover the associated map $\phi(\cdot)$ if one wishes to:

¹¹The remaining $M - N$ eigenvectors of $\Phi\Phi^T$ are associated with the eigenvalue zero.

¹²A function $k(\mathbf{x}, \mathbf{y})$ is semi-positive definite if $\int_{\mathbb{R}^D} f(\mathbf{x})k(\mathbf{x}, \mathbf{y})f(\mathbf{y}) d\mathbf{x}d\mathbf{y} \geq 0$

Theorem 2.5 (Mercer's Theorem). *Given a symmetric function $k(\mathbf{x}, \mathbf{y})$ with $|k(\cdot, \cdot)| \leq K$ for some K , if the linear operator $\mathcal{L} : L^2(\mathbb{R}^D) \rightarrow L^2(\mathbb{R}^D)$:*

$$\mathcal{L}(f)(\mathbf{x}) \doteq \int_{\mathbb{R}^D} k(\mathbf{x}, \mathbf{y}) f(\mathbf{y}) d\mathbf{y} \quad (2.32)$$

is semi-positive definite, then:

- *The operator \mathcal{L} has an eigenvalue-eigenfunction decomposition $\{(\lambda_i, \phi_i(\cdot))\}$ such that $\sum_i |\lambda_i| < \infty$ and $|\phi_i(\cdot)| < K_i$ for some K_i .*
- *The kernel $k(\mathbf{x}, \mathbf{y}) = \sum_i \lambda_i \phi_i(\mathbf{x}) \phi_i(\mathbf{y})$ for almost all (\mathbf{x}, \mathbf{y}) .¹³*

The interested readers may refer to [?] for a proof of the theorem.

One important reason for computing with the kernel function is because when the dimension of the feature space is high (sometimes even infinite), the computation of features and their inner products is expensive. But for many popular choices of embedding, the evaluation of the kernel function can be much simpler.

Example 2.6 (Examples of Kernels). There are several popular choices for the nonlinear kernel function:

$$k_1(\mathbf{x}, \mathbf{y}) = (\mathbf{x}^T \mathbf{y})^n, \quad k_2(\mathbf{x}, \mathbf{y}) = \exp\left(-\frac{\|\mathbf{x} - \mathbf{y}\|^2}{2}\right). \quad (2.33)$$

Evaluation of such functions only involves the inner product or the difference between two vectors in the original space \mathbb{R}^D . This is much more efficient than evaluating the inner product in the associated feature space, whose dimension for the first kernel grows exponentially with the degree n and for the second kernel is infinite. ■

We summarize our discussion in this section as Algorithm 2.1.

2.4 Bibliographic Notes

As a matrix decomposition tool, SVD was initially developed independently from PCA in the numerical linear algebra literature, also known as the Eckart and Young decomposition [Eckart and Young, 1936, Hubert et al., 2000]. The result regarding the least-squares optimality of SVD given in Theorem 2.1 can be traced back to [Householder and Young, 1938, Gabriel, 1978]. While principal components were initially defined exclusively in a statistical sense [Pearson, 1901, Hotelling, 1933], one can show that the algebraic solution given by SVD gives asymptotically unbiased estimates of the true parameters in the case of Gaussian distributions. A more detailed analysis of the statistical properties of PCA can be found in [Jolliffe, 2002].

Note that PCA only infers the principal subspace (or components), but not a probabilistic distribution of the data in the subspace. Probabilistic PCA was developed to infer an explicit probabilistic distribution from the data

¹³“Almost all” means except for a zero-measure set.

Algorithm 2.1 (Nonlinear Kernel PCA).

For a given set of data points $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N] \in \mathbb{R}^{D \times N}$, and a given map $\phi(\mathbf{x})$ or a kernel function $k(\mathbf{x}, \mathbf{y})$:

1. Compute the inner product matrix

$$\Phi^T \Phi = (\phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)) \text{ or } (k(\mathbf{x}_i, \mathbf{x}_j)) \in \mathbb{R}^{N \times N}; \quad (2.34)$$

2. Compute the eigenvectors $w_i \in \mathbb{R}^N$ of $\Phi^T \Phi$:

$$\Phi^T \Phi w_i = \lambda_i w_i, \quad (2.35)$$

and normalize $\|w_i\|^2 = \lambda_i^{-1}$;

3. For any data point \mathbf{x} , its i th nonlinear principal component is given by

$$y_i = w_i^T \Phi^T \phi(\mathbf{x}) \text{ or } w_i^T [k(\mathbf{x}_1, \mathbf{x}), \dots, k(\mathbf{x}_N, \mathbf{x})]^T, \quad (2.36)$$

for $i = 1, \dots, d$.

[Tipping and Bishop, 1999b]. The data is 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 or a maximum-a-posteriori sense. When the underlying noise distribution is Gaussian, the geometric and probabilistic interpretations of PCA coincide [Collins et al., 2001]. However, when the underlying distribution is non Gaussian, the optimal solution to PPCA may no longer be linear. For example, in [Collins et al., 2001] PCA is generalized to arbitrary distributions in the exponential family.

PCA is obviously not applicable to data whose underlying structure is nonlinear. PCA was generalized to principal curves and surfaces by [Hastie, 1984] and [Hastie and Stuetzle, 1989]. A more general approach however is to find a nonlinear embedding map, or equivalently a kernel function, such that the embedded data would lie on a linear subspace. Such methods are referred to as nonlinear kernel PCA [Scholkopf et al., 1998]. Finding such nonlinear maps or kernels are by no means simple problems. Learning kernels is still an active research topic in the machine learning community.

Chapter 3

Iterative Methods for Multiple-Subspace Segmentation

In this chapter, we consider a generalization of PCA in which the given sample points are drawn from an unknown arrangement of subspaces of unknown and possibly different dimensions. To some extent, this problem can be cast as a variation to the problem of unsupervised data clustering, that has been widely studied in pattern recognition and machine learning. We will first review some basic concepts and approaches for unsupervised data clustering. We then give a clear formulation of the problem in which the clusters are subspaces and introduce the basic notation for representing both linear and affine subspaces. We then customize two existing iterative algorithms from unsupervised learning, K-means and Expectation Maximization, for segmenting a known number of subspaces with known dimensions. We point out the advantages and disadvantages of these algorithms, particularly their sensitivity to initialization.

3.1 Unsupervised Learning Methods for Data Clustering

In clustering analysis, the basic assumption is that the given data points $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^N \subset \mathbb{R}^D$ are grouped into a number of clusters $n \leq N$ such that the “distance” (or “dissimilarity”) among points in the same group is significantly smaller than those between clusters. Thus the outcome of clustering analysis is a map:

$$c(\cdot) : i \in \{1, 2, \dots, N\} \mapsto j = c(i) \in \{1, 2, \dots, n\} \quad (3.1)$$

that assigns each point \mathbf{x}_i to one of the n clusters. Obviously, the outcome of the clustering very much depends on what the chosen measure of distance is. If the notion of distance is not clearly specified, the clustering problem can be ill-defined. The following example shows some of the reasons.

Example 3.1 (No Invariant Clustering by the Euclidean Distance). If we always choose the Euclidean distance, then clustering result *cannot* be invariant under an arbitrary linear transformation of the data points – usually representing a change of coordinates. That is, if we replace \mathbf{x}_i with $\mathbf{x}'_i = A\mathbf{x}_i$ for some non-singular matrix $A \in \mathbb{R}^{D \times D}$, then the clustering of $\{\mathbf{x}_i\}$ and $\{\mathbf{x}'_i\}$ will in general be different. This is easy to see with a simple example. Suppose we need to cluster the $N = 4$ points in \mathbb{R}^2 as follows

$$\mathbf{x}_1 = [1, 10]^T, \quad \mathbf{x}_2 = [-1, 10]^T, \quad \mathbf{x}_3 = [1, -10]^T, \quad \mathbf{x}_4 = [-1, -10]^T$$

into $n = 2$ clusters. The two clusters are obviously $\{\mathbf{x}_1, \mathbf{x}_2\}$ and $\{\mathbf{x}_3, \mathbf{x}_4\}$. Now consider two linear transformations A_1 and $A_2 \in \mathbb{R}^{2 \times 2}$:

$$A_1 = \begin{bmatrix} 100 & 0 \\ 0 & 1 \end{bmatrix}, \quad A_2 = \begin{bmatrix} 10 & 0 \\ 0 & 10 \end{bmatrix} \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & -10 \\ 10 & 0 \end{bmatrix}.$$

Applying the two maps to the original set of points, we obtain two new sets of points $\{\mathbf{x}'_i = A_1\mathbf{x}_i\}$ and $\{\mathbf{x}''_i = A_2\mathbf{x}_i\}$, respectively:

$$\begin{aligned} \mathbf{x}'_1 &= [100, 10]^T, \quad \mathbf{x}'_2 = [-100, 10]^T, \quad \mathbf{x}'_3 = [100, -10]^T, \quad \mathbf{x}'_4 = [-100, -10]^T; \\ \mathbf{x}''_1 &= [-100, 10]^T, \quad \mathbf{x}''_2 = [-100, -10]^T, \quad \mathbf{x}''_3 = [100, 10]^T, \quad \mathbf{x}''_4 = [100, -10]^T. \end{aligned}$$

As a set $\{\mathbf{x}'_i\}$ is the same as $\{\mathbf{x}''_i\}$. However, the two clusters are $\{\mathbf{x}'_1, \mathbf{x}'_3\}$ and $\{\mathbf{x}'_2, \mathbf{x}'_4\}$ for the first set; and $\{\mathbf{x}''_1, \mathbf{x}''_2\}$ and $\{\mathbf{x}''_3, \mathbf{x}''_4\}$ for the latter. In fact, regardless of the choice of objective or method, it is always the case that the clustering result for one of the two new sets will be different from that for the original set. ■

From the above example, we see that in order for the clustering result to be invariant under a linear transformation, instead of always using the distance in the original representation, one should properly adjust the distance measure after any linear transformation of the data. To be more precise, let the length of a vector $\mathbf{x} \in \mathbb{R}^D$ is measured by

$$\|\mathbf{x}\|_{\Sigma}^2 = \mathbf{x}^T \Sigma^{-1} \mathbf{x} \tag{3.2}$$

for some positive-definite symmetric matrix $\Sigma \in \mathbb{R}^{D \times D}$. Notice that $\Sigma = I_{D \times D}$ corresponds to the Euclidean length. Then under a linear transformation, $\mathbf{x}' = A\mathbf{x}$ for some $D \times D$ matrix A , the “induced” length of \mathbf{x}' is defined to be

$$\|\mathbf{x}'\|_{\Sigma'}^2 = (\mathbf{x}')^T (\Sigma')^{-1} \mathbf{x}' = (\mathbf{x}')^T (A\Sigma A^T)^{-1} \mathbf{x}' = \mathbf{x}^T \Sigma^{-1} \mathbf{x}. \tag{3.3}$$

Thus, the induced length remains the same after the transformation.

Notice that the relationship between Σ and $\Sigma' = A\Sigma A^T$ is just like that between the covariance matrices of two random vectors related by a linear transformation A . Thus, the change of distance measure is largely equivalent to the assumption that the original data $\{\mathbf{x}_i\}$ are drawn from some probabilistic distribution. In the context of clustering analysis, it is natural to further assume that the distribution itself is a mixture of n (Gaussian) distributions with different means and

covariances:¹

$$p_j(\mathbf{x}) \sim \mathcal{N}(\boldsymbol{\mu}_j, \Sigma_j), \quad j = 1, 2, \dots, n. \quad (3.4)$$

Thus, the clustering problem becomes a statistical model estimation problem and can be solved by principled statistical methods. We introduce below two such methods that based on two different estimation (and optimization) paradigms: 1. Minimax estimate; 2. Maximum-likelihood estimate. In the rest of this section, we illustrate the basic ideas using mixtures of Gaussians; and a discussion of the general case can be found in Appendix C.

3.1.1 K-Means

With respect to the above statistical model, a natural measure of the distance between a sample point and the mean of a cluster is given by the log-likelihood of the point with respect to the distribution of the cluster:

$$d(\mathbf{x}_i, \boldsymbol{\mu}_j) \doteq -\log p_j(\mathbf{x}_i) = \|\mathbf{x}_i - \boldsymbol{\mu}_j\|_{\Sigma_j}^2. \quad (3.5)$$

The map $c^*(\cdot)$ that represents an optimal clustering of the data $\{\mathbf{x}_i\}$ minimizes the following “within-cluster scatter”:

$$\min_{c(\cdot)} w(c) \doteq \frac{1}{N} \sum_{j=1}^n \sum_{c(i)=j} \|\mathbf{x}_i - \boldsymbol{\mu}_j\|_{\Sigma_j}^2. \quad (3.6)$$

That is, $w(c)$ is a measure of the average distance of all the sample points to their respective cluster means. Notice that the minimum value of $w(c)$ decreases with the increase of the number n of clusters. In the extreme case $n = N$, i.e., each point is a cluster itself, we have $w(c) = 0$. Therefore, before conducting clustering analysis, it is very important to know the correct value of n . We will discuss methods to determine n in later chapters; and in this chapter, we always assume the correct cluster number n is known.

In the above objective $w(c)$ (3.6), $c(\cdot)$, $\{\boldsymbol{\mu}_j\}$, and $\{\Sigma_j\}$ are all unknown. The problem is how to find the optimal $c^*(\cdot)$, $\boldsymbol{\mu}_j^*$ and Σ_j^* so that $w(c)$ is minimized. Unfortunately, there is no closed-form solution to the optimal solution. The main difficulty is that the objective (3.6) is hybrid – it is a combination of minimization on the continuous variables $\{\boldsymbol{\mu}_j, \Sigma_j\}$ and the discrete variable $c(i)$. Conventional nonlinear optimization techniques, such as gradient descent, do not directly apply to this case either. Hence special optimization schemes have to be developed.

Notice that for $w(c)$ to be minimum, it is necessary that each point \mathbf{x}_i is assigned to the cluster whose mean is the closest to \mathbf{x}_i . That is, given $\{\boldsymbol{\mu}_j, \Sigma_j\}$, we have

$$c(i) = \arg \min_j \|\mathbf{x}_i - \boldsymbol{\mu}_j\|_{\Sigma_j}^2. \quad (3.7)$$

¹From the viewpoint of subspaces, here we try to fit the data with *multiple* zero-dimensional affine spaces – one mean for each cluster. Later in this Chapter, we will see how to generalize the cluster means from points to arbitrary (affine) subspaces.

Also, from the samples that belong to each cluster, we can obtain unbiased estimates of the mean and covariance of the cluster:

$$\hat{\boldsymbol{\mu}}_j \doteq \frac{1}{N_j} \sum_{c(i)=j} \mathbf{x}_i \in \mathbb{R}^D, \quad \hat{\Sigma}_j \doteq \frac{1}{N_j - 1} \sum_{c(i)=j} (\mathbf{x}_i - \hat{\boldsymbol{\mu}}_j)(\mathbf{x}_i - \hat{\boldsymbol{\mu}}_j)^T \in \mathbb{R}^{D \times D}, \quad (3.8)$$

where N_j is the number of points that are assigned to cluster j by the map $c(\cdot)$.

The above discussions have suggested the following two-step iterative process for minimizing $w(c)$.

Suppose that some initial estimates $\{\hat{\boldsymbol{\mu}}_j^{(0)}, \hat{\Sigma}_j^{(0)}\}$ of the means are available. Then we can easily minimize the objective (3.6) for $c(i)$. That is, for each cluster with the mean $\hat{\boldsymbol{\mu}}_j^{(0)}$ and covariance $\hat{\Sigma}_j^{(0)}$, we obtain the subset of points $\mathbf{X}_j^{(0)}$ that are closer to $\hat{\boldsymbol{\mu}}_j^{(0)}$ than to any other means. The data set \mathbf{X} is therefore segmented into n clusters

$$\mathbf{X} = \mathbf{X}_1^{(0)} \cup \mathbf{X}_2^{(0)} \cup \cdots \cup \mathbf{X}_n^{(0)}, \quad (3.9)$$

and we further require $\mathbf{X}_j^{(0)} \cap \mathbf{X}_{j'}^{(0)} = \emptyset$ for $j \neq j'$.² In this way we obtain an estimate of the map $c^{(0)}(\cdot)$.

Knowing the membership of each point \mathbf{x}_i from the above segmentation, the objective (3.6) can be rewritten as:

$$\sum_{j=1}^n \left(\min_{\boldsymbol{\mu}_j, \Sigma_j} \sum_{c^{(0)}(i)=j} \|\mathbf{x}_i - \boldsymbol{\mu}_j\|_{\Sigma_j}^2 \right). \quad (3.10)$$

Notice that the solution to the minimization inside the bracket is a new set of estimates of the mean and covariance:

$$\hat{\boldsymbol{\mu}}_j^{(1)} = \frac{1}{N_j} \sum_{c^{(0)}(i)=j} \mathbf{x}_i, \quad \hat{\Sigma}_j^{(1)} = \frac{1}{N_j - 1} \sum_{c^{(0)}(i)=j} (\mathbf{x}_i - \hat{\boldsymbol{\mu}}_j^{(1)})(\mathbf{x}_i - \hat{\boldsymbol{\mu}}_j^{(1)})^T.$$

These new means and covariances give a new value of the objective no larger than that given by the initial estimates $\{\hat{\boldsymbol{\mu}}_j^{(0)}, \hat{\Sigma}_j^{(0)}\}$.

We can further reduce the objective by re-classifying each data point \mathbf{x}_i to its closest mean according to the new estimates $\{\hat{\boldsymbol{\mu}}_j^{(1)}, \hat{\Sigma}_j^{(1)}\}$. In this way, we obtain a new segmentation $\mathbf{X} = \mathbf{X}_1^{(1)} \cup \mathbf{X}_2^{(1)} \cup \cdots \cup \mathbf{X}_n^{(1)}$. If we keep iterating between the above two steps, the objective will keep decreasing until its value stabilizes to a (local) equilibrium and the segmentation no longer changes. This minimization process is referred to as the K-means algorithm in the statistical-learning literature. We summarize the algorithm as Algorithm 3.1.

Notice that Algorithm 3.1 can be significantly simplified if the Gaussian distributions are all *isotropic*, i.e., $\Sigma_j = \sigma_j^2 I$ for some $\sigma_j^2 \in \mathbb{R}_+$, or all covariance

²If a point $\mathbf{x} \in \mathbf{X}$ has the same minimal distance to more than one cluster, then we assign it arbitrarily to one of them.

Algorithm 3.1 (K-Means).

Given a set of sample points $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^N$, the number of clusters n , initialize the means and covariances of the clusters with a set of initial values $\hat{\boldsymbol{\mu}}_j^{(0)} \in \mathbb{R}^D$, $\hat{\Sigma}_j^{(0)} \in \mathbb{R}^{D \times D}$, $j = 1, 2, \dots, n$.

Let $m = 0$.

1. **Segmentation:** For each point $\mathbf{x}_i \in \mathbf{X}$, assign it to $\mathbf{X}_j^{(m)}$ if

$$j = c(i) = \operatorname{argmin}_{\ell=1,2,\dots,n} \|\mathbf{x}_i - \hat{\boldsymbol{\mu}}_\ell^{(m)}\|_{\hat{\Sigma}_\ell^{(m)}}^2. \quad (3.11)$$

If the above cost function is minimized by more than one mean, assign the point arbitrarily to one of them.

2. **Estimation:** Obtain new estimates for the n cluster means and covariances:

$$\begin{aligned} \hat{\boldsymbol{\mu}}_j^{(m+1)} &= \frac{1}{N_j} \sum_{c^{(m)}(i)=j} \mathbf{x}_i, \\ \hat{\Sigma}_j^{(m+1)} &= \frac{1}{N_j - 1} \sum_{c^{(m)}(i)=j} (\mathbf{x}_i - \hat{\boldsymbol{\mu}}_j^{(m+1)}) (\mathbf{x}_i - \hat{\boldsymbol{\mu}}_j^{(m+1)})^T \end{aligned} \quad (3.12)$$

Let $m \leftarrow m + 1$, and repeat Steps 1 and 2 until the segmentation does not change.

matrices are equal to the identity matrix $\Sigma_j \equiv I$. In the latter case, one essentially adopts the Euclidean distance between the sample points and the cluster means. This special case is often referred to also as the “K-means” algorithm in the literature.

3.1.2 Expectation Maximization (EM)

The K-means algorithm essentially relies on the minimax estimation paradigm in statistics (see Appendix C) and it does not need to assume how exactly the n component distributions are mixed. The Expectation Maximization (EM) algorithm [Dempster et al., 1977a] to be introduced below, however, relies on the maximum-likelihood estimation paradigm (see Appendix C) and it does need an explicit model for the mixed distribution. Instead of minimizing the modeling error in a least-distance sense, the EM algorithm estimates the model parameters and the segmentation of the data in a maximum-likelihood (ML) sense. As we shall soon see, the EM algorithm, though derived from a different set of assumptions, principles, and objectives, has an overall structure that resembles very much that of the K-means algorithm.³

³This resemblance however should not be mistaken as excuses to confuse these two algorithms. The solutions given by these two algorithms will be close but different in general.

A Probabilistic Model for a Mixed Distribution

The EM algorithm is based on the assumption that the given data points $\{\mathbf{x}_i\}_{i=1}^N$ are independent samples from a (mixed) probabilistic distribution. In the context of clustering analysis, it is reasonable to assume that \mathbf{x}_i are samples drawn from multiple “component” distributions and each component distribution is centered around a mean. To model from which component distribution a sample \mathbf{x} is actually drawn, we can associate a latent discrete random variable $z \in \mathbb{R}$ to each data point \mathbf{x} , such that each discrete random variable $z_i = j$ if the point \mathbf{x}_i is drawn from the j th component, $i = 1, 2, \dots, N$. Then the random vector

$$(\mathbf{x}, z) \in \mathbb{R}^D \times \mathbb{Z}_+ \quad (3.13)$$

completely describes the random event that the point \mathbf{x} is drawn from a component distribution indicated by the value of z .

Typically, one assumes that the random variable z is subject to a multinomial (marginal) distribution, i.e.,

$$p(z = j) = \pi_j \geq 0, \quad \text{s.t. } \pi_1 + \pi_2 + \dots + \pi_n = 1. \quad (3.14)$$

Each component distribution is then modeled as a conditional distribution $p(\mathbf{x}|z)$ of \mathbf{x} given z . A popular choice for the component distribution is a multivariate Gaussian distribution: $p(\mathbf{x}|z = j) \sim \mathcal{N}(\boldsymbol{\mu}_j, \Sigma_j)$, in which $\boldsymbol{\mu}_j$ is the mean and Σ_j is the covariance of the j th cluster.

The Maximum-Likelihood Estimation

In the model, the parameters $\theta \doteq \{\boldsymbol{\mu}_j, \Sigma_j, \pi_j\}_{j=1}^n$ are unknown and they need to be inferred from the samples of \mathbf{x} . The marginal distribution of \mathbf{x} given the parameters, the likelihood function, is

$$p(\mathbf{x}|\theta) = \sum_{z=1}^n p(\mathbf{x}|z, \theta)p(z|\theta) = \sum_{j=1}^n \pi_j p(\mathbf{x}|z = j, \theta). \quad (3.15)$$

Notice that $p(\mathbf{x}|\theta)$ is a “mixture” of n distributions $p(\mathbf{x}|z = j, \theta)$, $j = 1, 2, \dots, n$ that is exactly of the form (1.7) introduced in Chapter 1.

Given N *i.i.d.* samples $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^N$ from the distribution, the optimal estimates of the parameters $\hat{\theta}_{ML}$ are given by maximizing the log-likelihood function

$$l(\mathbf{X}; \theta) \doteq \sum_{i=1}^N \log p(\mathbf{x}_i|\theta). \quad (3.16)$$

In the statistical learning literature, this objective is often referred to as the *incomplete log-likelihood function* – “incomplete” compared to the complete log-likelihood function to be introduced later. However, maximizing the incomplete log-likelihood with respect to the parameters θ is typically very difficult, because this is a very high-dimensional nonlinear optimization problem. This is the motivation for the *expectation maximization* (EM) process which utilizes the latent

random variable z introduced earlier to attempt to simplify the maximization process.

Derivation of the Expectation and Maximization

First notice $p(\mathbf{x}|\theta) = p(\mathbf{x}, z|\theta)/p(z|\mathbf{x}, \theta)$ and $\sum_j p(z = j|\mathbf{x}, \theta) = 1$. We can rewrite the (incomplete) log-likelihood function as

$$l(\mathbf{X}; \theta) = \sum_{i=1}^N \sum_{j=1}^n p(z_i = j|\mathbf{x}_i, \theta) \log \frac{p(\mathbf{x}_i, z_i = j|\theta)}{p(z_i = j|\mathbf{x}_i, \theta)} \quad (3.17)$$

$$= \sum_{i=1}^N \sum_{j=1}^n p(z_i = j|\mathbf{x}_i, \theta) \log p(\mathbf{x}_i, z_i = j|\theta) \quad (3.18)$$

$$- \sum_{i=1}^N \sum_{j=1}^n p(z_i = j|\mathbf{x}_i, \theta) \log p(z_i = j|\mathbf{x}_i, \theta). \quad (3.19)$$

The first term (3.18) is what is called the *expected complete log-likelihood function* in the machine learning literature;⁴ and the second term (3.19) is the *conditional entropy*⁵ of z_i given \mathbf{x}_i and θ . Hence, the maximum-likelihood estimation is equivalent to maximizing the expected log-likelihood and at the same time minimizing the conditional entropy of z_i .

Given each \mathbf{x}_i , we can define a new function $w_{ij}(\theta) \doteq p(z_i = j|\mathbf{x}_i, \theta)$. By replacing $\mathbf{w}(\theta) = \{w_{ij}(\theta)\}$ into the incomplete log-likelihood, we can view $l(\mathbf{X}; \theta)$ as a new function

$$l(\mathbf{X}; \theta) \doteq g(\mathbf{w}(\theta), \theta). \quad (3.20)$$

Instead of directly maximizing the $l(\mathbf{X}; \theta)$ with respect to θ , we may maximize $g(\mathbf{w}(\theta), \theta)$ in a “hill-climbing” style by iterating between the following two steps:

Step 1. partially maximizing $g(\mathbf{w}(\theta), \theta)$ with respect to $\mathbf{w}(\theta)$ with the second θ fixed;

Step 2. partially maximizing $g(\mathbf{w}(\theta), \theta)$ with respect to the second θ with $\mathbf{w}(\theta)$ fixed (to the value obtained from Step 1.)

Notice that at each step the value of $g(\mathbf{w}(\theta), \theta)$ does not decrease, so does that of $l(\mathbf{X}; \theta)$. When the iteration converges to a stationary point θ^* , it must be a (local) extremum for the function $l(\mathbf{X}; \theta)$. To see this, examine the equation

$$\frac{dl(\mathbf{X}; \theta)}{d\theta} = \frac{\partial g(\mathbf{w}, \theta)}{\partial \mathbf{w}} \frac{\partial \mathbf{w}(\theta)}{\partial \theta} + \frac{\partial g(\mathbf{w}, \theta)}{\partial \theta}. \quad (3.21)$$

⁴That is, it is the expected value of the complete log-likelihood $\log p(\mathbf{x}, z|\theta)$ of the “complete” random vector (\mathbf{x}, z) with respect to the distribution of $(z|\mathbf{x}, \theta)$.

⁵The entropy of a (discrete) random variable z is defined to be $H(z) \doteq \sum_j p(z = j) \log p(z = j)$.

Since θ^* must be a stationary point for each step, we have $\frac{\partial g(\mathbf{w}, \theta)}{\partial \mathbf{w}} \Big|_{\theta^*} = 0$ and $\frac{\partial g(\mathbf{w}, \theta)}{\partial \theta} \Big|_{\theta^*} = 0$. Therefore, $\frac{dl(\mathbf{X}; \theta)}{d\theta} \Big|_{\theta^*} = 0$.

As we have alluded to earlier, the main reason for choosing this alternative maximization is that, for the log-likelihood function of a mixture of distributions, each of these two steps of maximizing g are much easier to compute than directly maximizing the original log-likelihood function. In fact, for Gaussian distributions, one can often find closed-form solutions to each step.

E-Step: Expected Membership of Samples. To find the optimal $\hat{\mathbf{w}} = \{\hat{w}_{ij}\}$ that maximize $g(\mathbf{w}, \theta)$, we need to minimize the function

$$\max_{\mathbf{w}} g(\mathbf{w}, \theta) = \sum_{i=1}^N \sum_{j=1}^n w_{ij} \log p(\mathbf{x}_i, z_i = j | \theta) - \sum_{i=1}^N \sum_{j=1}^n w_{ij} \log w_{ij} \quad (3.22)$$

with respect to \mathbf{w} subject to the constraints $\sum_j w_{ij} = 1$ for every i . For this purpose, we have the following statement.

Proposition 3.2 (Expected Membership). *The optimal $\hat{\mathbf{w}}$ that partially minimizes $g(\mathbf{w}, \theta)$ is given by:*

$$\hat{w}_{ij} = \frac{\pi_j p(\mathbf{x}_i | z_i = j, \theta)}{\sum_{\ell=1}^n \pi_\ell p(\mathbf{x}_i | z_i = \ell, \theta)}. \quad (3.23)$$

Proof. Using the Lagrange multipliers method, we differentiate the objective function

$$\sum_{i=1}^N \sum_{j=1}^n \left(w_{ij} \log p(\mathbf{x}_i, z_i = j | \theta) - w_{ij} \log w_{ij} \right) + \sum_{i=1}^N \lambda_i \left(\sum_{j=1}^n w_{ij} - 1 \right). \quad (3.24)$$

with respect to w_{ij} and set the derivatives to zero. We obtain the necessary conditions for extrema:

$$\log p(\mathbf{x}_i, z_i = j | \theta) - \log w_{ij} - 1 + \lambda_i = 0 \quad (3.25)$$

for every i and j . Solving for w_{ij} from this equation, we obtain:

$$w_{ij} = e^{\lambda_i - 1} p(\mathbf{x}_i, z_i = j | \theta). \quad (3.26)$$

Since $\sum_j w_{ij} = 1$, we have $e^{\lambda_i - 1} = (\sum_{\ell} p(\mathbf{x}_i, z_i = \ell | \theta))^{-1}$. In addition,

$$p(\mathbf{x}_i, z_i = j | \theta) = p(\mathbf{x}_i | z_i = j, \theta) p(z_i = j | \theta) = \pi_j p(\mathbf{x}_i | z_i = j, \theta).$$

We hence have the claim of the proposition. \square

M-Step: Maximize the Expected Complete Log-Likelihood. Now we consider the second step in which we fix \mathbf{w} in $g(\mathbf{w}, \theta)$ and we maximize it with respect to θ . This means we fix $w_{ij} = p(z_i = j | \mathbf{x}_i, \theta)$ in the expression of $l(\mathbf{X}; \theta)$. The second term (3.19) of $l(\mathbf{X}; \theta)$ is therefore fixed as far as this step is concerned. Hence it is equivalent to maximizing the first term (3.18), the so-called expected

complete log-likelihood:

$$L(\mathbf{X}; \theta) \doteq \sum_{i=1}^N \sum_{j=1}^n w_{ij} \log (\pi_j p(\mathbf{x}_i | z_i = j, \theta)). \quad (3.27)$$

For many common choices of the distributions $p(\mathbf{x}|z = j, \theta)$, we can find closed-form solutions to the maximum of $L(\mathbf{X}; \theta)$.

For simplicity, in the clustering analysis, we may assume that each cluster is an isotropic normal distribution, i.e., $p(\mathbf{x}|z = j, \theta) = \mathcal{N}(\boldsymbol{\mu}_j, \sigma_j^2 I)$. Minimizing $L(\mathbf{X}; \theta)$ is then equivalent to minimizing the function

$$\sum_{i=1}^N \sum_{j=1}^n w_{ij} \left(\log \pi_j - D \log \sigma_j - \frac{\|\mathbf{x}_i - \boldsymbol{\mu}_j\|^2}{\sigma_j^2} \right), \quad (3.28)$$

where we have omitted terms that depend on only the fixed w_{ij} and constants. The goal of maximization is to find the parameters $\hat{\theta} = \{(\hat{\boldsymbol{\mu}}_j, \hat{\sigma}_j, \hat{\pi}_j)\}_{j=1}^n$ that maximize the above expression. Since $\sum_{j=1}^n \pi_j = 1$, this is a constrained optimization problem, which can be solved in closed-form using Lagrange-multiplier method:

$$\hat{\boldsymbol{\mu}}_j = \frac{\sum_{i=1}^N w_{ij} \mathbf{x}_i}{\sum_{i=1}^N w_{ij}}, \quad \hat{\sigma}_j^2 = \frac{\sum_{i=1}^N w_{ij} \|\mathbf{x}_i - \hat{\boldsymbol{\mu}}_j\|^2}{D \sum_{i=1}^N w_{ij}}, \quad \hat{\pi}_j = \frac{\sum_{i=1}^N w_{ij}}{N}. \quad (3.29)$$

We summarize the above results as Algorithm 3.2.

In comparison with K-means (Algorithm 3.1), the EM algorithm assigns each point \mathbf{x}_i , instead by a deterministic map $j = c(i)$, “softly” to each cluster j according to a probability w_{ij} (that are subject to $\sum_{j=1}^n w_{ij} = 1$). Subsequently, the number of points N_j in the j th cluster is expected to be $\sum_{i=1}^N w_{ij}$; the ratio $\frac{N_j}{N}$ is expected as $\frac{\sum_{i=1}^N w_{ij}}{N}$; and the means $\boldsymbol{\mu}_j$ in (3.12) are replaced by an expected version in (3.31). In general, if the variances σ_j are significantly smaller than the distances between the means $\boldsymbol{\mu}_j$, the K-means and EM algorithms give similar clustering results.

From the above derivation, each step of the EM algorithm increases the log-likelihood function $l(\mathbf{X}; \theta)$. However, beware that a stationary value θ^* that the algorithm converges to is not necessarily the global maximum (if the global maximum exists at all). Furthermore, for distributions as simple as a mixture of Gaussian distributions, the global maximum may not even exist! We illustrate this via the following example.

Example 3.3 (ML Estimate of Two Mixed Gaussians [Vapnik, 1995]). Consider a distribution $p(x)$, $x \in \mathbb{R}$ that is a mixture of two Gaussian (normal) distributions:

$$p(x, \mu, \sigma) = \frac{1}{2\sigma\sqrt{2\pi}} \exp \left\{ -\frac{(x - \mu)^2}{2\sigma^2} \right\} + \frac{1}{2\sqrt{2\pi}} \exp \left\{ -\frac{x^2}{2} \right\}, \quad (3.32)$$

where $\theta = (\mu, \sigma)$ are unknown.

Then for any data $\mathbf{X} = \{x_1, x_2, \dots, x_N\}$ and for any given constant $A > 0$, there exists a small σ_0 such that for $\mu = x_1$ the log-likelihood will exceed A (regardless of the

Algorithm 3.2 (Expectation Maximization).

Given a set of sample points $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^N \subset \mathbb{R}^D$ drawn from n (isotropic) Gaussian clusters $\mathcal{N}(\boldsymbol{\mu}_j, \sigma_j^2 I), j = 1, 2, \dots, n$, initialize the parameters $\theta = \{\boldsymbol{\mu}_j, \sigma_j, \pi_j\}$ with a set of vectors $\hat{\boldsymbol{\mu}}_j^{(0)} \in \mathbb{R}^D$ and scalars $\hat{\sigma}_j^{(0)}, \hat{\pi}_j^{(0)} \in \mathbb{R}$. Let $m = 0$.

1. **Expectation:** Using the current estimate for the parameters $\hat{\theta}^{(m)} = \{\hat{\boldsymbol{\mu}}_j^{(m)}, \hat{\sigma}_j^{(m)}, \hat{\pi}_j^{(m)}\}$, compute the estimate of w_{ij} as

$$w_{ij}^{(m)} = p(z_i = j | \mathbf{x}_i, \hat{\theta}^{(m)}) = \frac{\hat{\pi}_j^{(m)} p(\mathbf{x}_i | z_i = j, \hat{\theta}^{(m)})}{\sum_{\ell=1}^n \hat{\pi}_\ell^{(m)} p(\mathbf{x}_i | z_i = \ell, \hat{\theta}^{(m)})}, \quad (3.30)$$

where $p(\mathbf{x}|z = j, \theta)$ is given in (3.43).

2. **Maximization:** Using the estimated $w_{ij}^{(m)}$, update the estimates for the parameters $\hat{\boldsymbol{\mu}}_j, \hat{\sigma}_j$ as:

$$\hat{\boldsymbol{\mu}}_j^{(m+1)} = \frac{\sum_{i=1}^N w_{ij}^{(m)} \mathbf{x}_i}{\sum_{i=1}^N w_{ij}^{(m)}}, \quad (\hat{\sigma}_j^{(m+1)})^2 = \frac{\sum_{i=1}^N w_{ij}^{(m)} \|\mathbf{x}_i - \hat{\boldsymbol{\mu}}_j^{(m+1)}\|^2}{D \sum_{i=1}^N w_{ij}^{(m)}}, \quad (3.31)$$

and update $\hat{\pi}_j$ as $\hat{\pi}_j^{(m+1)} = \frac{\sum_{i=1}^N w_{ij}^{(m)}}{N}$.

Let $m \leftarrow m + 1$, and repeat Steps 1 and 2 until the update in the parameters is small enough.

true μ, σ):

$$\begin{aligned} l(\mathbf{X}; \theta) \Big|_{\mu=x_1, \sigma=\sigma_0} &= \sum_{i=1}^N \ln p(x_i | \mu = x_1, \sigma = \sigma_0) \\ &> \ln \left(\frac{1}{2\sigma_0 \sqrt{2\pi}} \right) + \sum_{i=2}^N \ln \left(\frac{1}{2\sqrt{2\pi}} \exp \left\{ -\frac{x_i^2}{2} \right\} \right) \\ &= -\ln \sigma_0 - \sum_{i=1}^N \frac{x_i^2}{2} - N \ln 2\sqrt{2\pi} > A. \end{aligned}$$

Therefore, the maximum of the log-likelihood does not exist, and the ML objective does not provide a solution to estimating the unknown parameters. In fact, in this case, the true parameter corresponds to the largest (finite) local maximum of the log-likelihood. ■

From the simple example, we can conclude that the ML method only applies to very restrictive set of densities.⁶ If we insist using it for mixtures of Gaussians, we have to rule out the situations that the variance can be arbitrarily small, i.e., $\sigma_0 \rightarrow 0$. Fortunately, in practice, the EM algorithm typically tends to avoid such singular directions and is able to converge to a local minimum that represents the true

⁶For instance, a class of density functions that are bounded by a common finite value from above.

parameters if a reasonable initialization is given. However, this leads to another potential problem: What if the distributions to be estimated are indeed close to being singular? This is unfortunately the case with subspace-like distributions.⁷ Thus, singular distributions like subspaces require special treatment.

Also notice that the above K-means and EM algorithms are derived mainly for isotropic Gaussian distributions. In practice, a cluster is rarely isotropic. For instance, as we have seen in PCA, a cluster can be a set of points sampled from a principal subspace. For the above reasons, in the next two sections of this chapter (Section 3.2 and 3.3), we will extend the basic ideas of K-means and EM to the case in which clusters are subspaces.

3.2 Problem Formulation of Subspace Segmentation

In mathematics (especially in algebraic geometry), a collection of subspaces is formally known as a subspace arrangement:

Definition 3.4 (Subspace Arrangement). *A subspace arrangement is defined as a finite collection of n linear subspaces in R^D : $\mathcal{A} \doteq \{S_1, \dots, S_n\}$. The union of the subspaces is denoted as $Z_{\mathcal{A}} \doteq S_1 \cup S_2 \cup \dots \cup S_n$.*

For simplicity, we will call both \mathcal{A} and $Z_{\mathcal{A}}$ “subspace arrangement.”

Imagine that we are given a set of sample points drawn from an arrangement of unknown number of subspaces which have unknown and possibly different dimensions. Our goal is to simultaneously estimate these subspaces and segment the points into their corresponding subspaces. Versions of this problem are known in the literature as *subspace clustering*, *multiple eigenspaces* [Leonardis et al., 2002], or *mixtures of principal component analyzers* [Tipping and Bishop, 1999a], etc. To be precise, we will first state the problem that we will study in this book, which we refer to as “multiple-subspace segmentation,” or simply as “subspace segmentation,” to be suggestive of the problem of fitting multiple (principal) subspaces to the data.

Notice that in the foregoing problem statement, we have not yet specified the objective for what is an “optimal” solution. We will leave the interpretation of that open for now and will delay the definition until the context is more specific. Although the problem seems to be stated in a purely geometric fashion, it is easy to re-formulate it in a statistical fashion. For instance, we have assumed here that the subspaces do not have to be orthogonal to each other. In a statistical setting, this is essentially equivalent to assuming that these subspaces are not necessarily uncorrelated. Within each subspace, one can also relate all the geometric and statistical notions associated with “principal components” in the classical PCA: The orthonormal basis chosen for each subspace usually corresponds to a decompo-

⁷A subspace-like distribution is one that has large variance inside the subspace but very small (close to singular) variance in directions orthogonal to the subspace.

Problem 3.1 (Multiple-Subspace Segmentation).

Given a set of sample points $\mathbf{X} = \{\mathbf{x}_i \in \mathbb{R}^D\}_{i=1}^N$ drawn from $n \geq 1$ distinct linear subspaces $S_j \subset \mathbb{R}^D$ of dimensions $d_j < D$, $j = 1, 2, \dots, n$, identify each subspace S_j without knowing which sample points belong to which subspace.⁸ More specifically, by identifying the subspaces we mean the following:

1. Identifying the number of subspaces n and their dimensions $d_j = \dim(S_j)$;
2. Identifying an orthonormal basis for each subspace S_j (or equivalently S_j^\perp);⁹
3. Clustering the N points into the subspaces to which they belong.

sition of the random variable into uncorrelated principal components *conditioned on the subspace*. In Section 3.3, a detailed analysis and comparison will be given for both points of view.

3.2.1 Projectivization of Affine Subspaces

Note that a linear subspace always passes through the origin but an affine subspace does not. So, would the above problem statement lose any generality by restricting it only to linear subspaces? The answer to this question is no. In fact every proper affine subspace in \mathbb{R}^D can be converted to a proper linear subspace in \mathbb{R}^{D+1} by lifting every point of it through the so-called homogeneous coordinates:

Definition 3.5 (Homogeneous Coordinates). *The homogeneous coordinates of a point $\mathbf{x} = [x_1, x_2, \dots, x_D]^T \in \mathbb{R}^D$ are defined to as $[x_1, x_2, \dots, x_D, 1]^T$.*

Given a set of points in an affine subspace, it is easy to prove that their homogeneous coordinates span a linear subspace. More precisely:

Fact 3.6 (Homogeneous Representation of Affine Subspaces). *The homogeneous coordinates of points on a k -dimensional affine subspace in \mathbb{R}^D span a $(d + 1)$ -dimensional linear subspace in \mathbb{R}^{D+1} . This representation is one-to-one.*

Figure 3.1 shows an example of the homogeneous representation of three lines in \mathbb{R}^2 . The points on these lines span three linear subspaces in \mathbb{R}^3 which pass through the origin.

Definition 3.7 (Central Subspace Arrangements). *We call an arrangement of subspaces is central if every subspace passes through the origin, i.e., every subspace is a linear subspace.*

According to this definition, the homogeneous representation of any (affine) subspace arrangement in \mathbb{R}^D gives a central subspace arrangement in \mathbb{R}^{D+1} . Therefore, Problem 3.1 does not loss any generality. From now on, we may assume that our data set is drawn from a central subspace arrangement, in which all

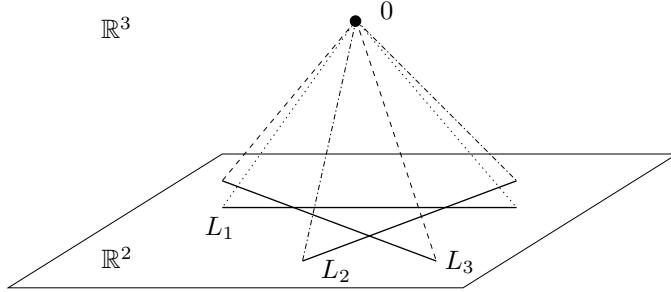


Figure 3.1. Lifting of three (affine) lines in \mathbb{R}^2 to three linear subspaces in \mathbb{R}^3 via the homogeneous representation.

subspaces are linear, not affine, subspaces, unless otherwise stated. In a statistical setting, this is equivalent to assuming that each subset of samples has zero mean.

3.2.2 Subspace Projection and Minimum Representation

There are many cases in which the given data points live in a very high dimensional space. For instance, in many computer vision problems the dimension of the ambient space D is the number of pixels in an image, which is normally in the range 10^6 . In such cases, the complexity of any subspace segmentation solution become computationally prohibitive. It is therefore important for us to seek situations in which the dimension of the ambient space can be significantly reduced.

Fortunately, in most practical applications, we are interested in modeling the data by subspaces of relatively small dimensions ($d \ll D$), thus one can avoid dealing with high-dimensional data sets by first projecting them onto a lower-dimensional (sub)space. 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 P . In this case, segmenting the two lines in the three-dimensional space \mathbb{R}^3 is equivalent to segmenting the two projected lines in the two-dimensional plane P .

In general, we will distinguish between two different kinds of “projections.” The first kind corresponds to the case in which the span of all the subspaces is a proper subspace of the ambient space, i.e., $\text{span}(\cup_{j=1}^n S_j) \subset \mathbb{R}^D$. In this case, one may simply apply PCA (Chapter 2) to eliminate the redundant dimensions. The second kind corresponds to the case in which the largest dimension of the subspaces, denoted by d_{\max} , is strictly less than $D - 1$. When d_{\max} is known,¹⁰ one may choose a $(d_{\max}+1)$ -dimensional subspace P such that, by projecting \mathbb{R}^D onto this subspace:

$$\pi_P : \mathbf{x} \in \mathbb{R}^D \mapsto \mathbf{x}' = \pi_P(\mathbf{x}) \in P, \quad (3.33)$$

¹⁰For example, in 3-D motion segmentation from affine cameras, it is known that the subspaces have dimension at most four [Costeira and Kanade, 1998, Kanatani, 2001, Vidal and Hartley, 2004].

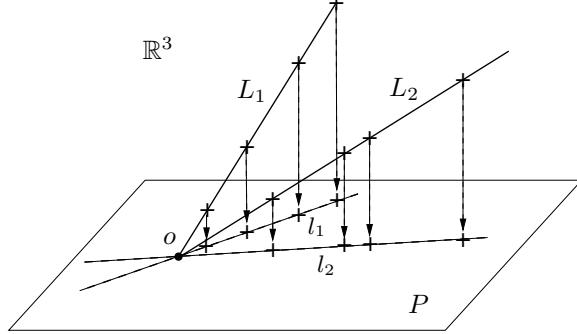


Figure 3.2. Samples on two 1-dimensional subspaces L_1, L_2 in \mathbb{R}^3 projected onto a 2-dimensional plane P . The number and separation of the lines is preserved by the projection.

the dimension of each original subspace S_j is preserved,¹¹ and there is a one-to-one correspondence between S_j and its projection – no reduction in the number of subspaces n ,¹² as stated in the following theorem.

Theorem 3.8 (Segmentation-Preserving Projections). *If a set of vectors $\{\mathbf{x}_i\}$ all lie in n linear subspaces of dimensions $\{d_j\}_{j=1}^n$ in \mathbb{R}^D , and if π_P represents a linear projection onto a subspace P of dimension D' , then the points $\{\pi_P(\mathbf{x}_i)\}$ lie in at most n linear subspaces of P of dimensions $\{d'_j \leq d_j\}_{j=1}^n$. Furthermore, if $D > D' > d_{\max}$, then there is an open and dense set of projections that preserve the separation and dimensions of the subspaces.*

Thanks to Theorem 3.8, if we are given a data set \mathbf{X} drawn from an arrangement of low-dimensional subspaces in a high-dimensional space, we can first project \mathbf{X} onto a generic subspace of dimension $D' = d_{\max} + 1$ and then model the data with a subspace arrangement in the projected subspace, as illustrated by the following sequence of steps:

$$\mathbf{X} \subset \mathbb{R}^D \xrightarrow{\pi_P} \mathbf{X}' \subset P \longrightarrow \bigcup_{j=1}^n \pi_P(S_j) \xrightarrow{\pi_P^{-1}} \bigcup_{j=1}^n S_j. \quad (3.34)$$

However, even though the set of $(d_{\max} + 1)$ -dimensional subspaces $P \subset \mathbb{R}^D$ that preserve the separation and dimension of the subspaces is an open and dense set, it remains unclear as to what a “good” choice for P is, especially when there is noise in the data. For simplicity, one may randomly select a few projections and choose the one that results in the smallest fitting error. Another alternative

¹¹This requires that P be transversal to each S_j^\perp , i.e., $\text{span}\{P, S_j^\perp\} = \mathbb{R}^D$ for every $j = 1, 2, \dots, n$. Since n is finite, this transversality condition can be easily satisfied. Furthermore, the set of positions for P which violate the transversality condition is only a zero-measure closed set [Hirsch, 1976].

¹²This requires that all $\pi_P(S_j)$ be transversal to each other in P , which is guaranteed if we require P to be transversal to $S_j^\perp \cap S_{j'}^\perp$ for $j, j' = 1, 2, \dots, n$. All P 's which violate this condition form again only a zero-measure set.

is to apply PCA regardless and project the data onto the $(d_{\max} + 1)$ -dimensional principal subspace.

One solution for choosing P is attributed to [Broomhead and Kirby, 2000]. The technique was originally designed for dimension reduction of differential manifolds.¹³ We here adopt it for subspace arrangements. Instead of directly using the original data matrix \mathbf{X} , we gather the vectors (also called “secants”) defined by every pair of points $\mathbf{x}_i, \mathbf{x}_j \in \mathbf{X}$

$$\mathbf{y}_{ij} \doteq \mathbf{x}_i - \mathbf{x}_j \quad \in \mathbb{R}^D, \quad (3.35)$$

and construct a matrix consisting of \mathbf{y}_{ij} as columns:

$$\mathbf{Y} \doteq [\mathbf{y}_{12}, \mathbf{y}_{13}, \dots, \mathbf{y}_{(N-1)N}] \quad \in \mathbb{R}^{D \times M}, \quad (3.36)$$

where $M = (N-1)N/2$. Then the principal components of \mathbf{Y} span the subspace in which the distance (and hence the separateness) between the projected points is preserved the most. Therefore, the optimal subspace that maximizes the separateness of the projected points is given by the $d_{\max} + 1$ principal components of \mathbf{Y} . More precisely, if $\mathbf{Y} = \mathbf{U}\Sigma\mathbf{V}^T$ is the SVD of \mathbf{Y} , then the optimal subspace P is given by the first $d_{\max} + 1$ columns of \mathbf{U} .

3.3 Subspace-Segmentation Algorithms

In this section, we generalize the K-means and EM algorithms to estimate arrangements of principal subspaces and cluster points into subspaces. They can both be viewed as certain extension of PCA to multiple principal subspaces. Both algorithms assume that the number of subspaces n and their dimensions $d_j, j = 1, 2, \dots, n$ are known. They estimate a basis for each subspace and the segmentation of the data by optimizing certain objective functions, namely the least-squares error in the geometric setting or the log-likelihood in the statistical setting. Since the optimal solution is normally not available in closed-form, the optimization problem is solved by iterating between the segmentation of the data points and the estimation of the subspace bases, starting from an initial guess for the subspace bases.

The following sections give a detailed description of both algorithms tailored to Problem 3.1. The goal is to reveal the similarity and difference between these two algorithms as well as their advantages and disadvantages.

3.3.1 K-Subspaces

If the number of subspaces n and their dimensions $d_j, j = 1, 2, \dots, n$ are known, then the problem of fitting multiple subspaces to the data is to find orthogonal

¹³That is essentially based on Whitney's classic proof of the fact any differential manifold can be embedded in a Euclidean space.

matrices $U_j, j = 1, 2, \dots, n$ of the dimension $D \times d_j$ such that

$$\forall i \exists j \text{ such that } \mathbf{x}_i = U_j \mathbf{y}_i, \quad (3.37)$$

where $i \in \{1, 2, \dots, N\}$ and $j \in \{1, 2, \dots, n\}$. Once the assignment map $c(i) = j$ is found for each point \mathbf{x}_i , \mathbf{y}_i is simply given by $\mathbf{y}_i = U_{c(i)}^T \mathbf{x}_i$. When \mathbf{x}_i is at the intersection of two subspaces, the solution for $c(i)$ and therefore \mathbf{y}_i is not unique. In this case, we arbitrarily choose one of the possible solutions.

In case the given points are corrupted by noise, we expect that the model parameters be found in a least-squares sense by minimizing the modeling error between \mathbf{x}_i and its closest projection onto the subspaces:

$$\min_{\{U_j\}} \sum_{i=1}^N \min_j \| \mathbf{x}_i - U_j U_j^T \mathbf{x}_i \|^2, \quad (3.38)$$

where U_j is a $D \times d_j$ orthogonal matrix that represents a basis for the j th subspace $S_j, j = 1, 2, \dots, n$. Unfortunately, unlike PCA, there is no constructive solution to the above minimization problem. The main difficulty is that the foregoing objective of (3.38) is hybrid – it is a combination of minimization on the continuous variables $\{U_j\}$ and the discrete variable j . Conventional nonlinear optimization techniques, such as gradient descent, do not directly apply to this case. Hence special optimization schemes have to be developed. For that purpose, we need to examine more closely the relationships between the two minimizations in the above objective function:

Suppose that some initial estimates $\hat{U}_1^{(0)}, \hat{U}_2^{(0)}, \dots, \hat{U}_n^{(0)}$ of the subspaces are available. Then we can easily minimize the objective (3.38) for j . That is, for each subspace S_j defined by $\hat{U}_j^{(0)}$, we obtain the subset of points $\mathbf{X}_j^{(0)}$ that are closer to S_j than to any other subspace. The data set \mathbf{X} is therefore segmented into n groups

$$\mathbf{X} = \mathbf{X}_1^{(0)} \cup \mathbf{X}_2^{(0)} \cup \dots \cup \mathbf{X}_n^{(0)}, \quad (3.39)$$

and we further require $\mathbf{X}_i^{(0)} \cap \mathbf{X}_j^{(0)} = \emptyset$ for $i \neq j$.¹⁴

Knowing the membership of each point \mathbf{x}_i from the above segmentation, the objective (3.38) can be rewritten as:

$$\sum_{j=1}^n \left(\min_{U_j} \sum_{\mathbf{x}_i \in \mathbf{X}_j^{(0)}} \| \mathbf{x}_i - U_j U_j^T \mathbf{x}_i \|^2 \right). \quad (3.40)$$

Notice that the minimization inside the bracket is exactly the same as the minimization in (2.6). Consequently, we have solved this problem in Theorem 2.1 for PCA. We can therefore apply PCA to each group of points $\{\mathbf{X}_j^{(0)}\}$ to obtain new

¹⁴If a point $\mathbf{x} \in \mathbf{X}$ has the same minimal distance to more than one subspace, then we assign it to an arbitrary subspace.

estimates for the bases $\{\hat{U}_j^{(1)}\}$. Such estimates give a modeling error no larger than the error given by the initial estimates $\{\hat{U}_j^{(0)}\}$.

We can further reduce the modeling error by re-classifying each data point \mathbf{x}_i to its closest subspace according to the new estimates $\{\hat{U}_j^{(1)}\}$. In this way, we obtain a new segmentation $\mathbf{X} = \mathbf{X}_1^{(1)} \cup \mathbf{X}_2^{(1)} \cup \dots \cup \mathbf{X}_n^{(1)}$. If we keep iterating between the above two steps, the modeling error will keep decreasing until its value stabilizes to a (local) equilibrium and the segmentation no longer changes. This minimization process is in essence an extension of the K-means algorithm to subspaces. We summarize the algorithm as Algorithm 3.3.

Algorithm 3.3 (K-Subspaces: K-Means for Subspace Segmentation).

Given a set of noisy sample points $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^N$ drawn from n subspaces with the dimensions $d_j, j = 1, 2, \dots, n$, initialize the bases of the subspaces with a set of orthogonal matrices $\hat{U}_j^{(0)} \in \mathbb{R}^{D \times d_j}$.

Let $m = 0$.

1. **Segmentation:** For each point $\mathbf{x}_i \in \mathbf{X}$, assign it to $\mathbf{X}_j^{(m)}$ if

$$j = \arg \min_{\ell=1, \dots, n} \|\mathbf{x}_i - \hat{U}_\ell^{(m)} (\hat{U}_\ell^{(m)})^T \mathbf{x}_i\|^2.$$

If the above cost function is minimized by more than one subspace, assign the point arbitrarily to one of them.

2. **Estimation:** Apply PCA to each subset $\mathbf{X}_j^{(m)}$ using Theorem 2.1 and obtain new estimates for the subspace bases

$$\hat{U}_j^{(m+1)} = \text{PCA}(\mathbf{X}_j^{(m)}), \quad j = 1, 2, \dots, n.$$

Let $m \leftarrow m + 1$, and repeat Steps 1 and 2 until the segmentation does not change.

3.3.2 Expectation Maximization for Subspaces

To apply the EM method in Section 3.1.2 to subspaces, we need to assume a statistical model for the data. Following the general setting in Section 3.1.2, it is reasonable to assume that the data points $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^N$ are samples drawn from multiple component distributions and each component distribution is centered around a subspace. To model from which component distribution a sample \mathbf{x} is actually drawn, we again associate a latent discrete random variable $z \in \mathbb{R}$ to every data point \mathbf{x} , where each discrete random variable $z_i = j$ if the point \mathbf{x}_i is drawn from the j th component, $i = 1, 2, \dots, N$.

To model the fact that each component distribution has a principal subspace, say spanned by the columns of an orthogonal matrix $U_j \in \mathbb{R}^{D \times d_j}$, we may assume that the j th component distribution is a special Gaussian distribution

determined by the following equation:

$$\mathbf{x} = U_j \mathbf{y} + B_j \mathbf{s}, \quad (3.41)$$

where the orthogonal matrix $B_j \in \mathbb{R}^{D \times (D-d_j)}$ is the orthogonal complement to the orthogonal matrix $U_j \in \mathbb{R}^{D \times d_j}$, and $\mathbf{y} \sim \mathcal{N}(0, \sigma_y^2 I)$ and $\mathbf{s} \sim \mathcal{N}(0, \sigma_s^2 I)$. If we further assume that \mathbf{y} and \mathbf{s} are independent random variables, then we have

$$\Sigma_j^{-1} = \sigma_y^{-2} U_j U_j^T + \sigma_s^{-2} B_j B_j^T. \quad (3.42)$$

The term $B_j \mathbf{s}$ models the projection error of \mathbf{x} onto the subspace spanned by U_j . For \mathbf{x} to be close to the subspace, one may assume $\sigma_s^2 \ll \sigma_y^2$. Therefore, when $\sigma_y^2 \rightarrow \infty$, we have $\Sigma_j^{-1} \rightarrow \sigma_s^{-2} B_j B_j^T$. In the limiting case, one essentially assumes a uniform distribution for \mathbf{y} inside the subspace. The uniform assumption suggests that we do not care much about the distribution of the data inside the subspace – it is the subspace itself in which we are interested. Technically, this assumption also helps eliminate additional parameters so that the ML method may better avoid the difficulty shown in Example 3.3. In practice, this assumption is approximately valid as long as the variance of the data inside the subspace is significantly larger than that outside the subspace.

Therefore, in the sequel, we will adopt the limiting case as our probabilistic model for the derivation of the EM algorithm and derive closed-form formulae for the different steps of the EM algorithm. More precisely, we assume the distributions are

$$p(\mathbf{x}|z=j) \doteq \frac{1}{(2\pi\sigma_j^2)^{(D-d_j)/2}} \exp\left(-\frac{\mathbf{x}^T B_j B_j^T \mathbf{x}}{2\sigma_j^2}\right). \quad (3.43)$$

In the model, the parameters $\theta \doteq \{B_j, \sigma_j, \pi_j\}_{j=1}^n$ are unknowns and they need to be inferred from the samples of \mathbf{x} . The marginal distribution of \mathbf{x} given the parameters, the likelihood function, is

$$\begin{aligned} p(\mathbf{x}|\theta) &= \sum_{z=1}^n p(\mathbf{x}|z, \theta) p(z|\theta) \\ &= \sum_{j=1}^n \frac{\pi_j}{(2\pi\sigma_j^2)^{(D-d_j)/2}} \exp\left(-\frac{\mathbf{x}^T B_j B_j^T \mathbf{x}}{2\sigma_j^2}\right). \end{aligned} \quad (3.44)$$

Then given the N samples $\mathbf{X} = \{\mathbf{x}_i\}$, the optimal estimates of the parameters $\hat{\theta}_{ML}$ are given by maximizing the log-likelihood function

$$l(\mathbf{X}; \theta) \doteq \sum_{i=1}^N \log p(\mathbf{x}_i|\theta) \quad (3.45)$$

$$= \sum_{i=1}^N \log \left[\sum_{j=1}^n \frac{\pi_j}{(2\pi\sigma_j^2)^{(D-d_j)/2}} \exp\left(-\frac{\mathbf{x}_i^T B_j B_j^T \mathbf{x}_i}{2\sigma_j^2}\right) \right] \quad (3.46)$$

Again, this is in general a difficult high-dimensional optimization problem. Thus, we can apply the Expectation Maximization method introduced in Section 3.1.2.

All the analysis in Section 3.1.2 directly applies to this new log-likelihood function except that in the M-Step, under the new probabilistic model, the new expected complete log-likelihood $L(\mathbf{X}; \theta)$ becomes

$$\sum_{i=1}^N \sum_{j=1}^n w_{ij} \left(\log \pi_j - (D - d_j) \log \sigma_j - \frac{\|\mathbf{B}_j^T \mathbf{x}_i\|^2}{2\sigma_j^2} \right), \quad (3.47)$$

where, as before, we have omitted terms that depend on only the fixed w_{ij} and constants. The goal now is to find the parameters $\hat{\theta} = \{\hat{B}_j, \hat{\sigma}_j, \hat{\pi}_j\}_{j=1}^n$ that maximize the above expected complete log-likelihood. Since $\mathbf{B}_j^T \mathbf{B}_j = I$ and $\sum_{j=1}^n \pi_j = 1$, this is again a constrained optimization problem, whose solutions are given by the following proposition.

Proposition 3.9 (Maximum of the Expected Complete Log-Likelihood). *The parameters $\hat{\theta} = \{\hat{B}_j, \hat{\sigma}_j, \hat{\pi}_j\}_{j=1}^n$ that maximize the expected complete log-likelihood function are: \hat{B}_j are exactly the eigenvectors associated with the smallest $D - d_j$ eigenvalues of the weighted sample covariance matrix $\hat{\Sigma}_j \doteq \sum_{i=1}^N w_{ij} \mathbf{x}_i \mathbf{x}_i^T$, and $\hat{\pi}_j$ and $\hat{\sigma}_j^2$ are*

$$\hat{\pi}_j = \frac{\sum_{i=1}^N w_{ij}}{N}, \quad \hat{\sigma}_j^2 = \frac{\sum_{i=1}^N w_{ij} \|\hat{B}_j^T \mathbf{x}_i\|^2}{(D - d_j) \sum_{i=1}^N w_{ij}}. \quad (3.48)$$

Proof. The part of objective function associated with the bases $\{\mathbf{B}_j\}$ can be rewritten as

$$\sum_{i=1}^N \sum_{j=1}^n -w_{ij} \frac{\|\mathbf{B}_j^T \mathbf{x}_i\|^2}{2\sigma_j^2} = \sum_{j=1}^n -\text{trace} \left(\frac{\mathbf{B}_j^T \hat{\Sigma}_j \mathbf{B}_j}{2\sigma_j^2} \right), \quad (3.49)$$

where $\hat{\Sigma}_j = \sum_{i=1}^N w_{ij} \mathbf{x}_i \mathbf{x}_i^T$. Differentiate the Lagrangian associated with \mathbf{B}_j , set the derivatives to zero, and we obtain the necessary conditions for extrema:

$$\sum_{j=1}^n -\text{trace} \left(\frac{\mathbf{B}_j^T \hat{\Sigma}_j \mathbf{B}_j}{2\sigma_j^2} \right) + \text{trace} (\Lambda_j (\mathbf{B}_j^T \mathbf{B}_j - I)) \Rightarrow \hat{\Sigma}_j \mathbf{B}_j = 2\sigma_j^2 \mathbf{B}_j \Lambda_j.$$

Since $\mathbf{B}_j^T \mathbf{B}_j = I$, the objective function for \mathbf{B}_j becomes $-\sum_{j=1}^n \text{trace}(\Lambda_j)$. Thus \hat{B}_j can be obtained as the matrix whose columns are the eigenvectors of $\hat{\Sigma}_j$ associated with the $(D - d_j)$ smallest eigenvalues.

From the Lagrangian associated with the mixing proportions $\{\pi_j\}$, we have

$$\min \sum_{i=1}^N \sum_{j=1}^n w_{ij} \log(\pi_j) + \lambda \left(1 - \sum_{j=1}^n \pi_j \right) \Rightarrow \hat{\pi}_j = \frac{\sum_{i=1}^N w_{ij}}{N}. \quad (3.50)$$

Finally, after taking the derivative of the expected log-likelihood with respect to σ_j and setting it to zero, we obtain

$$\hat{\sigma}_j^2 = \frac{\sum_{i=1}^N w_{ij} \|\hat{B}_j^T \mathbf{x}_i\|^2}{(D - d_j) \sum_{i=1}^N w_{ij}}. \quad (3.51)$$

□

We summarize the above results as Algorithm 3.4.

Algorithm 3.4 (EM for Subspace Segmentation).

Given a set of sample points $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^N \subset \mathbb{R}^D$, the number of subspaces n and the dimensions d_j , initialize the parameters $\theta = \{B_j, \sigma_j, \pi_j\}$ with a set of initial orthogonal matrices $\hat{B}_j^{(0)} \in \mathbb{R}^{D \times (D-d_j)}$ and scalars $\hat{\sigma}_j^{(0)}, \hat{\pi}_j^{(0)}, j = 1, 2, \dots, n$. Let $m = 0$.

1. **Expectation:** Using the current estimate for the parameters $\hat{\theta}^{(m)} = \{\hat{B}_j^{(m)}, \hat{\sigma}_j^{(m)}, \hat{\pi}_j^{(m)}\}$, compute the estimate of w_{ij} as

$$w_{ij}^{(m)} = p(z_i = j | \mathbf{x}_i, \hat{\theta}^{(m)}) = \frac{\hat{\pi}_j^{(m)} p(\mathbf{x}_i | z_i = j, \hat{\theta}^{(m)})}{\sum_{\ell=1}^n \hat{\pi}_\ell^{(m)} p(\mathbf{x}_i | z_i = \ell, \hat{\theta}^{(m)})}, \quad (3.52)$$

where $p(\mathbf{x}|z = j, \theta)$ is given in (3.43).

2. **Maximization:** Using the estimated $w_{ij}^{(m)}$, compute $\hat{B}_j^{(m+1)}$ as the eigenvectors associated with the smallest $D - d_j$ eigenvalues of the matrix $\hat{\Sigma}_j^{(m)} \doteq \sum_{i=1}^N w_{ij}^{(m)} \mathbf{x}_i \mathbf{x}_i^T$, and update $\hat{\pi}_j$ and $\hat{\sigma}_j$ as:

$$\hat{\pi}_j^{(m+1)} = \frac{\sum_{i=1}^N w_{ij}^{(m)}}{N}, \quad (\hat{\sigma}_j^{(m+1)})^2 = \frac{\sum_{i=1}^N w_{ij}^{(m)} \|(\hat{B}_j^{(m+1)})^T \mathbf{x}_i\|^2}{(D - d_j) \sum_{i=1}^N w_{ij}^{(m)}}. \quad (3.53)$$

Let $m \leftarrow m + 1$, and repeat Steps 1 and 2 until the update in the parameters is small enough.

3.3.3 Relationships between K-Subspaces and EM

As we have seen in the above, both K-subspaces and EM are algorithms that can be used to analyze arrangements of principal subspaces and fit multiple subspaces to a given set of data points. Both algorithms optimize their objectives via an iterative scheme. The overall structure of the two algorithms is also very much similar: the “Segmentation” step in K-subspaces is replaced by the “Expectation” step in EM; and “Estimation” by “Maximization”.

In addition to the structural similarity, there are also subtle technical relationships between the two steps of K-subspaces and EM. To see this, let us further assume that in the EM algorithm, the noise has the same variance for all the subspaces (i.e., $\sigma = \sigma_1 = \dots = \sigma_n$). According to equation (3.49), the EM algorithm updates the estimates for the subspaces in the “Maximization” step by minimizing

the objective function:

$$\min_{\{B_j\}} \sum_{i=1}^N \sum_{j=1}^n w_{ij} \|B_j^T \mathbf{x}_i\|^2 = \min_{\{U_j\}} \sum_{i=1}^N \sum_{j=1}^n w_{ij} \|\mathbf{x}_i - U_j U_j^T \mathbf{x}_i\|^2, \quad (3.54)$$

where the equality is due to the identity $B_j B_j^T = I - U_j U_j^T$. For EM, the weights w_{ij} are computed from the “Expectation” step as the expected membership of \mathbf{x}_i in the subspaces j according to the equation (3.23), and w_{ij} in general take continuous values between 0 and 1. For K-subspaces, however, w_{ij} is a discrete variable and it is computed in the “Segmentation” step as (see Algorithm 3.3):

$$w_{ij} = \begin{cases} 1 & \text{if } j = \arg \min_{\ell=1,\dots,n} \|B_\ell^T \mathbf{x}_i\|^2, \\ 0 & \text{otherwise.} \end{cases} \quad (3.55)$$

Then the objective function (3.54) can be rewritten as:

$$\min_{\{U_j\}} \sum_{i=1}^N \sum_{j=1}^n w_{ij} \|\mathbf{x}_i - U_j U_j^T \mathbf{x}_i\|^2 = \min_{\{U_j\}} \sum_{i=1}^N \min_j \|\mathbf{x}_i - U_j U_j^T \mathbf{x}_i\|^2, \quad (3.56)$$

which is exactly the same objective function (3.38) for K-subspaces. This is also the reason why both K-subspaces and EM rely on the eigenvalue decomposition (or singular value decomposition) of the sample covariance matrix to estimate the basis for each subspace.

Based on the above analysis, the only conceptual difference between the K-subspaces and EM algorithm is: At each iteration, the K-subspaces algorithm gives a “definite” assignment of every data point into one of the subspaces; but the EM algorithm views the membership as a random variable and uses its expected value to give a “probabilistic” assignment of the data point. Because of this difference, for the same set of data points, the “subspaces” found by using K-subspaces and EM will in general be different, although normally the difference is expected to be small. A precise quantitative characterization of the difference between the solutions by K-subspaces and EM remains an open question. Also because of this difference, the K-subspaces algorithm is less dependent on the correct knowledge in the dimension of each subspace: as long as the initial subspaces may segment the data well enough, both the basis and the dimension of each subspace can be updated at the Estimation step. However, the EM algorithm, at least for the version we presented above, depends explicitly on correct knowledge in both the number of subspaces and their dimensions. In addition, both algorithms require a good initialization so that they are more likely to converge to the optimal solution (e.g., the global maximum of the log likelihood) when the iteration stabilizes. In the next chapter, we will show how these difficulties can be resolved by a new algebraic method for identifying arrangements of principal subspaces.

3.4 Bibliographic Notes

When the data points lie on an arrangement of subspaces, the modeling problem was initially treated as “chicken-and-egg” and tackled with iterative methods, such as the K-means and EM algorithms. The basic ideas of K-means clustering goes back to [Lloyd, 1957, Forgy, 1965, Jancey, 1966, MacQueen, 1967]. Its probabilistic counterpart, the EM algorithm is due to [Dempster et al., 1977b, Neal and Hinton, 1998]. In [Tipping and Bishop, 1999a], mixtures of probabilistic PCA were studied, and the maximum likelihood solution was found via expectation maximization (EM). The classical K-means algorithm was also extended to the case of subspaces, called K-subspace [Ho et al., 2003]. Some other algorithms such as subspace growing and subspace selection [Leonardis et al., 2002] were also proposed. Unfortunately, as we have alluded to above, iterative methods are sensitive to initialization, hence they may not converge to the global optimum [Shi and Malik, 1998, Torr et al., 2001].

Chapter 4

Algebraic Methods for Multiple-Subspace Segmentation

In Chapter 3, we have shown that the subspace segmentation problem can be cast as a special case of an unsupervised learning or clustering problem. We have presented two iterative algorithms (K-means and EM) for segmenting a known number of subspaces with known dimensions. This chapter starts to reveal the main agenda of this book. We examine a different solution to the more general problem of segmenting an unknown number of subspaces of unknown and possibly different dimensions. In order to make the material accessible to a larger audience, in this chapter we focus primarily on the development of a (conceptual) algorithm. We leave a more formal study of subspace arrangements and more rigorous justifications of the derivation of the algorithm to the next chapter (Chapter B).

We first present a series of simple examples that demonstrate that the subspace-segmentation problem can be solved non-iteratively via certain algebraic methods. These solutions lead to a general-purpose algebro-geometric algorithm for subspace segmentation. We conveniently refer to the algorithm as Generalized Principal Component Analysis (GPCA). To better isolate the difficulties in the general problem, we will develop the algorithm in two steps. The first step is to develop a basic GPCA algorithm by assuming a known number of subspaces; and in the second step, we deal with an unknown number of subspaces and develop a recursive GPCA algorithm. The algorithms in this chapter will be derived under ideal noise-free conditions and assume no probabilistic model. Nevertheless, the algebraic techniques involved are numerically well-conditioned and the algorithms are designed to tolerate moderate amounts of noise. Dealing with large amounts of noise, outliers, or even incomplete data will be the subject of Chapter 5.

4.1 Introductory Cases of Subspace Segmentation

Notice that, to apply the K-subspaces and EM algorithms, we need to know three things in advance: the number of subspaces, their dimensions, and initial estimates of the bases of the subspaces. In practice, this may not be the situation and many difficulties may arise. The optimizing process in both algorithms is essentially a local iterative descent scheme. If the initial estimates of the bases of the subspaces are far off from the global optimum, the process is likely to converge to a local minimum. More seriously, if the number of subspaces and their dimensions were wrong, the process might never converge or might converge to meaningless solutions. Furthermore, when the number of subspaces and their dimensions are unknown, model selection becomes a much more elusive problem as we have alluded to earlier in the introduction.

In this and next few chapters, we will systematically address these difficulties and aim to arrive at global non-iterative solutions to subspace segmentation that require less or none of the above initial information. Before we delve into the most general problem, we first examine, in this section, a few important special cases. The reason is two-fold: Firstly, many practical problems fall into these cases already and the simplified solutions can be directly applied; and secondly, the analysis of these special cases offers some insights into a solution to the general case.

4.1.1 Segmenting Points on a Line

Let us begin with an extremely simple clustering problem: clustering a collection of points $\{x_i\}_{i=1}^N$ on the real line \mathbb{R} around a collection of cluster centers $\{\mu_j\}_{j=1}^n$. In spite of its simplicity, this problem shows up in various segmentation problems. For instance, in intensity-based image segmentation, one wants to separate the pixels of an image into different regions, with each region corresponding to a significantly different level of intensity (a one-dimensional quantity). More generally, the point clustering problem is very much at the heart of spectral clustering, a popular technique for clustering data in spaces of any dimension. Furthermore, as we will see throughout this book, the same basic ideas introduced through this simple example can also be applied to clustering points from arrangements of more complex structures such as lines, hyperplanes, subspaces, and even surfaces.

In this sequel, we introduce a not so conventional solution to the point clustering problem. The new formulation that the solution is based on is neither geometric (like K-subspaces) nor statistical (like EM). Instead, the solution is purely *algebraic*.

Let $x \in \mathbb{R}$ be any of the data points. In an ideal situation in which each data point perfectly matches one of the cluster centers, we know that there exists a constant μ_j such that $x = \mu_j$. This means that

$$(x = \mu_1) \vee (x = \mu_2) \vee \cdots \vee (x = \mu_n). \quad (4.1)$$

The “ \vee ” in the preceding equation stands for the logical connective “or.” This problem is equivalent to say that x satisfies the following polynomial equation of degree n in x :

$$p_n(x) \doteq (x - \mu_1)(x - \mu_2) \cdots (x - \mu_n) = \sum_{k=0}^n c_k x^{n-k} = 0. \quad (4.2)$$

Since the polynomial equation $p_n(x) = 0$ must be satisfied by every data point, we have that

$$\mathbf{V}_n \mathbf{c}_n \doteq \begin{bmatrix} x_1^n & x_1^{n-1} & \cdots & x_1 & 1 \\ x_2^n & x_2^{n-1} & \cdots & x_2 & 1 \\ \vdots & \vdots & & \vdots & \vdots \\ x_N^n & x_N^{n-1} & \cdots & x_N & 1 \end{bmatrix} \begin{bmatrix} 1 \\ c_1 \\ \vdots \\ c_n \end{bmatrix} = 0, \quad (4.3)$$

where $\mathbf{V}_n \in \mathbb{R}^{N \times (n+1)}$ is a matrix of embedded data points, and $\mathbf{c}_n \in \mathbb{R}^{n+1}$ is the vector of coefficients of $p_n(x)$.

In order to determine the number of groups n and then the vector of coefficients \mathbf{c}_n from (4.3), notice that for n groups there is a unique polynomial of degree n whose roots are the n cluster centers. Since the coefficients of this polynomial must satisfy equation (4.3), in order to have a unique solution we must have that $\text{rank}(\mathbf{V}_n) = n$. This rank constraint on $\mathbf{V}_n \in \mathbb{R}^{N \times (n+1)}$ enables us to determine the number of groups n as¹

$$n \doteq \min\{j : \text{rank}(\mathbf{V}_j) = j\}. \quad (4.4)$$

Example 4.1 (Two Clusters of Points). The intuition behind this formula is 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 equation of degree one, $p_1(x) = x - \mu$, that is satisfied by *all* the points. Similarly, there are infinitely many polynomial equations of degree 3 or more that are satisfied by all the points, namely any multiple of $p_2(x)$. Thus the degree $n = 2$ is the only one for which there is a unique polynomial that fits all the points. ■

Once the minimum polynomial $p_n(x)$ that fits all the data points is found, we can solve the equation $p_n(x) = 0$ for its n roots. These roots, by definition, are the centers of the clusters. We summarize the overall solution as Algorithm 4.1.

Notice that the above algorithm is described in a purely algebraic fashion and is more of a conceptual than practical algorithm. It does not minimize any geometric errors or maximize any probabilistic likelihood functions. In the presence of noises in the data, one has to implement each step of the algorithm in a numerically more stable and statistically more robust way. For example, with noisy data, the matrix \mathbf{V}_n will most likely be of full rank. In this case, the vector of

¹Notice that the minimum number of points needed is $N \geq n$, which is *linear* in the number of groups. We will see in future chapters that this is no longer the case for more general segmentation problems.

Algorithm 4.1 (Algebraic Point Clustering Algorithm).

Let $\{\mathbf{x}_i\}_{i=1}^N \subset \mathbb{R}$ be a given collection of $N \geq n$ points clustering around an unknown number n of cluster centers $\{\mu_j\}_{j=1}^n$. The number of groups, the cluster centers and the segmentation of the data can be determined as follows:

1. **Number of Groups.** Let $\mathbf{V}_j \in \mathbb{R}^{N \times (j+1)}$ be a matrix containing the last $j + 1$ columns of \mathbf{V}_n . Determine the number of groups as

$$n \doteq \min\{j : \text{rank}(\mathbf{V}_j) = j\}. \quad (4.5)$$

2. **Cluster Centers.** Solve for \mathbf{c}_n from $\mathbf{V}_n \mathbf{c}_n = 0$. Set $p_n(x) = \sum_{k=0}^n c_k x^{n-k}$. Find the cluster centers $\{\mu_j\}_{j=1}^n$ as the n roots of $p_n(x)$.

3. **Segmentation.** Assign point x_i to cluster $j = \arg \min_{l=1,\dots,n} (x_i - \mu_l)^2$.

coefficients \mathbf{c}_n should be solved in a least-squares sense as the singular-vector of \mathbf{V}_n associated with the smallest singular value. It is also possible that the $p_n(x)$ obtained from \mathbf{c}_n may have some complex roots, because the constraint that the polynomial must have real roots is never enforced when solving for the coefficients in the least-squares sense.² In practice, for well-separated clusters with moderate noises, the roots normally give decent estimates of the cluster centers.

Clustering Points in a (Complex) Plane.

Algorithm 4.1 can also be applied to a set of points in the plane $\{\mathbf{x}_i \in \mathbb{R}^2\}_{i=1}^N$ that are distributed around a collection of cluster centers $\{\mu_j \in \mathbb{R}^2\}_{j=1}^n$ by interpreting the data points as complex numbers: $\{z \doteq x + y\sqrt{-1} \in \mathbb{C}\}$. The only difference is that now the polynomial $p_n(z)$ has complex coefficients and complex roots. One can obtain the cluster centers by interpreting the complex roots as points in \mathbb{R}^2 .

Connection of Algebraic Clustering with Spectral Clustering.

Although the problem of clustering data on a line may seem rather simple, it is the key to one of the popular clustering algorithms: *spectral clustering*. In spectral clustering, one is given a set of N data points around n centers and an $N \times N$ pairwise similarity matrix S .³ The points can be clustered into n groups by clustering the entries of one or more eigenvectors of S into n values. If we let $\mathbf{v} = [x_1, x_2, \dots, x_N]^T$ be one such eigenvector, then we can use Algorithm 4.1

²However, in some specially cases, one can show that this would never occur. For example, for $\mathbf{x} = [x_1, \dots, x_N]^T$, if $n = 2$ the least-squares solution for \mathbf{c}_n is $c_2 = \text{Var}[\mathbf{x}]$, $c_1 = E[\mathbf{x}^2]E[\mathbf{x}] - E[\mathbf{x}^3]$ and $c_0 = E[\mathbf{x}^3]E[\mathbf{x}] - E[\mathbf{x}^2]^2 \leq 0$, hence $c_1^2 - 4c_0c_2 \geq 0$ and the roots of the polynomial $c_0x^2 + c_1x + c_2$ are real.

³The entries s_{ij} of S measure the likelihood of two points belonging to the same cluster: $s_{ij} \rightarrow 1$ when points i and j likely belong to the same group and $s_{ij} \rightarrow 0$ when points i and j likely belong to different groups.

to automatically cluster the entries of \mathbf{v} around n cluster centers $\{\mu_j\}_{j=1}^n$. In the case of two eigenvectors \mathbf{v}_1 and \mathbf{v}_2 , we define a complex vector $\mathbf{z} = \mathbf{v}_1 + \mathbf{v}_2\sqrt{-1}$ and apply Algorithm 4.1 to the complex data points.

4.1.2 Segmenting Lines on a Plane

Let us now consider the case of clustering data points to a collection of n lines in \mathbb{R}^2 passing through the origin, as illustrated in Figure 4.1. Each one of the lines can be represented as:

$$L_j \doteq \{\mathbf{x} = [x, y]^T : b_{j1}x + b_{j2}y = 0\}, \quad j = 1, 2, \dots, n. \quad (4.6)$$

Given a point $\mathbf{x} = [x, y]^T$ in one of the lines we must have that

$$(b_{11}x + b_{12}y = 0) \vee \dots \vee (b_{n1}x + b_{n2}y = 0). \quad (4.7)$$

Therefore, even though each individual line is described with one polynomial equation of degree one (a linear equation), an arrangement of n lines can be described with a polynomial of degree n , namely

$$p_n(\mathbf{x}) = (b_{11}x + b_{12}y) \cdots (b_{n1}x + b_{n2}y) = \sum_{k=0}^n c_k x^{n-k} y^k = 0. \quad (4.8)$$

An example is shown in Figure 4.1.

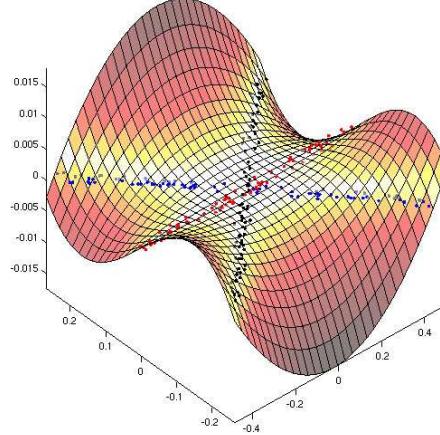


Figure 4.1. A polynomial in two variables whose zero set is three lines in \mathbb{R}^2 .

The polynomial $p_n(\mathbf{x})$ allows us to algebraically eliminate the segmentation of the data at the beginning of the model estimation, because the equation $p_n(\mathbf{x}) = 0$ is satisfied by every data point regardless of whether it belongs to L_1 , L_2 or L_n . Furthermore, even though $p_n(\mathbf{x})$ is nonlinear in each data point $\mathbf{x} = [x, y]^T$, $p_n(\mathbf{x})$ is actually linear in the vector of coefficients $\mathbf{c} = [c_0, \dots, c_n]^T$. Therefore,

given enough data points $\{\mathbf{x}_i = [x_i, y_i]^T\}_{i=1}^N$, one can linearly fit this polynomial to the data. Indeed, if n is known, we can obtain the coefficients of $p_n(\mathbf{x})$ from solving the equation:

$$\mathbf{V}_n \mathbf{c}_n = \begin{bmatrix} x_1^n & x_1^{n-1}y_1 & \cdots & x_1y_1^{n-1} & y_1^n \\ x_2^n & x_2^{n-1}y_2 & \cdots & x_2y_2^{n-1} & y_2^n \\ \vdots & \vdots & & \vdots & \vdots \\ x_N^n & x_N^{n-1}y_N & \cdots & x_Ny_N^{n-1} & y_N^n \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_n \end{bmatrix} = 0. \quad (4.9)$$

Similarly to the case of points in a line, the above linear system has a unique solution if and only if $\text{rank}(\mathbf{V}_n) = n$, hence the number of lines is given by

$$n \doteq \min\{j : \text{rank}(\mathbf{V}_j) = j\}. \quad (4.10)$$

Given the vector of coefficients \mathbf{c}_n , we are now interested in estimating the equations of each line from the associated polynomial $p_n(\mathbf{x})$. We know each line is determined by its normal vector $\mathbf{b}_j = [b_{1j}, b_{2j}]^T$, $j = 1, 2, \dots, n$. For the sake of simplicity, let us consider the case $n = 2$. A simple calculation shows that the derivative of $p_2(\mathbf{x})$ is given by

$$\nabla p_2(\mathbf{x}) = (b_{21}x + b_{22}y)\mathbf{b}_1 + (b_{11}x + b_{12}y)\mathbf{b}_2. \quad (4.11)$$

Therefore, if the point \mathbf{x} belongs to L_1 , then $(b_{11}x + b_{12}y) = 0$ and hence $\nabla p_2(\mathbf{x}) \sim \mathbf{b}_1$. Similarly if \mathbf{x} belongs to L_2 , then $\nabla p_2(\mathbf{x}) \sim \mathbf{b}_2$. This means that given any point \mathbf{x} , without knowing which line contains the point, we can obtain the equation of the line passing through the point by simply evaluating the derivative of $p_2(\mathbf{x})$ at \mathbf{x} . This fact should come at no surprise and is valid for any number of lines n . Therefore, if we are given one point in each line⁴ $\{\mathbf{y}_j \in L_j\}$, we can determine the normal vectors as $\mathbf{b}_j \sim \nabla p_n(\mathbf{y}_j)$. We summarize the overall solution for clustering points to multiple lines as Algorithm 4.2.

Connection between Point Clustering and Line Segmentation.

The reader may have realized that, the problem of clustering points on a line is very much related to the problem of segmenting lines in the plane. In point clustering, for each data point x there exists a cluster center μ_j such that $x - \mu_j = 0$. By working in homogeneous coordinates, one can convert it into a line clustering problem: for each data point $\mathbf{x} = [x, 1]^T$ there is a line $\mathbf{b}_j = [1, -\mu_j]^T$ passing through the point. Figure 4.2 shows an example of how three cluster centers are converted into three lines via homogeneous coordinates. Indeed, notice that if we let $y = 1$ in the matrix \mathbf{V}_n in (4.9), we obtain exactly the matrix \mathbf{V}_n in (4.3). Therefore, the vector of coefficients \mathbf{c}_n is the same for both algorithms and the two polynomials are related as $p_n(x, y) = y^n p_n(x/y)$. Therefore, the point clustering problem can be solved either by polynomial factorization (Algorithm 4.1) or by polynomial differentiation (Algorithm 4.2).

⁴We will discuss how to automatically obtain one point per subspace from the data in the next subsection when we generalize this problem to clustering points on hyperplanes.

Algorithm 4.2 (Algebraic Line Segmentation Algorithm)

Let $\{\mathbf{x}_i \in \mathbb{R}^2\}_{i=1}^N$ be a given collection of $N \geq n$ points clustering around an unknown number n of lines $\{\mathbf{b}_j\}_{j=1}^n$. The number of lines, the normal vectors and the segmentation of the data can be determined as follows:

1. **Number of Lines.** Let \mathbf{V}_j be defined as in (4.9). Determine the number of groups as

$$n \doteq \min\{j : \text{rank}(\mathbf{V}_j) = j\}. \quad (4.12)$$

2. **Normal Vectors.** Solve for \mathbf{c}_n from $\mathbf{V}_n \mathbf{c}_n = 0$ and set $p_n(x, y) = \sum_{k=0}^n c_k x^{n-k} y^k$. Determine the normal vectors as

$$\mathbf{b}_j = \frac{\nabla p_n(\mathbf{y}_j)}{\|\nabla p_n(\mathbf{y}_j)\|} \in \mathbb{R}^2, \quad j = 1, 2, \dots, n, \quad (4.13)$$

where \mathbf{y}_j is a point in the j th line.

3. **Segmentation.** Assign point \mathbf{x}_i to line $j = \arg \min_{\ell=1, \dots, n} (\mathbf{b}_\ell^T \mathbf{x}_i)^2$.

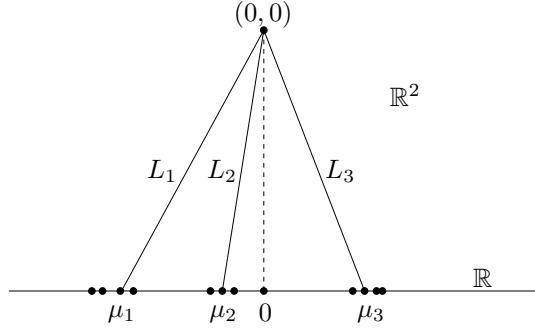


Figure 4.2. Using homogeneous coordinates to convert the point clustering problem into the line segmentation problem.

4.1.3 Segmenting Hyperplanes

In this section, we consider another particular case of Problem 3.1 in which all the subspaces are hyperplanes of equal dimension $d_1 = \dots = d_n = d = D - 1$. This case shows up in a wide variety of segmentation problems in computer vision, including vanishing point detection and motion segmentation. We will discuss these applications in greater detail in later chapters.

We start by noticing that every $(D-1)$ -dimensional subspace $S_j \subset \mathbb{R}^D$ can be defined in terms of a nonzero *normal* vector $\mathbf{b}_j \in \mathbb{R}^D$ as follows:⁵

$$S_j \doteq \{\mathbf{x} \in \mathbb{R}^D : \mathbf{b}_j^T \mathbf{x} \doteq b_{j1}x_1 + b_{j2}x_2 + \cdots + b_{jD}x_D = 0\}. \quad (4.14)$$

Therefore, a point $\mathbf{x} \in \mathbb{R}^D$ lying in one of the hyperplanes S_j must satisfy the formula:

$$(\mathbf{b}_1^T \mathbf{x} = 0) \vee (\mathbf{b}_2^T \mathbf{x} = 0) \vee \cdots \vee (\mathbf{b}_n^T \mathbf{x} = 0), \quad (4.15)$$

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

$$p_n(\mathbf{x}) = \prod_{j=1}^n (\mathbf{b}_j^T \mathbf{x}) = \sum c_{n_1, n_2, \dots, n_D} x_1^{n_1} x_2^{n_2} \cdots x_D^{n_D} = \nu_n(\mathbf{x})^T \mathbf{c}_n = 0, \quad (4.16)$$

where $c_{n_1, \dots, n_D} \in \mathbb{R}$ represents the coefficient of monomial $x_1^{n_1} x_2^{n_2} \cdots x_D^{n_D}$, \mathbf{c}_n is the vector of all coefficients, and $\nu_n(\mathbf{x})$ is the stack of all possible monomials. The number of linearly independent monomials is $M_n \doteq \binom{D+n-1}{D}$, hence \mathbf{c}_n and $\nu_n(\mathbf{x})$ are vectors in \mathbb{R}^{M_n} .

After applying (4.16) to the given collection of N sample points $\{\mathbf{x}_i\}_{i=1}^N$, we obtain the following system of linear equations on the vector of coefficients \mathbf{c}_n

$$\mathbf{V}_n \mathbf{c}_n \doteq \begin{bmatrix} \nu_n(\mathbf{x}_1)^T \\ \nu_n(\mathbf{x}_2)^T \\ \vdots \\ \nu_n(\mathbf{x}_N)^T \end{bmatrix} \mathbf{c}_n = 0 \in \mathbb{R}^N. \quad (4.17)$$

We now study under what conditions we can solve for n and \mathbf{c}_n from equation (4.17). To this end, notice that if the number of hyperplanes n was known, we could immediately recover \mathbf{c}_n as the eigenvector of $\mathbf{V}_n^T \mathbf{V}_n$ associated with its smallest eigenvalue. However, since the above linear system (4.17) depends explicitly on the number of hyperplanes n , we cannot estimate \mathbf{c}_n directly without knowing n in advance. Recall from Example B.14, the vanishing ideal \mathcal{I} of a hyperplane arrangement is always principal, i.e., generated by a single polynomial of degree n . The number of hyperplanes n then coincides with the degree of the first non-trivial homogeneous component \mathcal{I}_n of the vanishing ideal. This leads to the following theorem.

Theorem 4.2 (Number of Hyperplanes). *Assume that a collection of $N \geq M_n - 1$ sample points $\{\mathbf{x}_i\}_{i=1}^N$ on n different $(D-1)$ -dimensional subspaces of \mathbb{R}^D is given. Let $\mathbf{V}_j \in \mathbb{R}^{N \times M_j}$ be the matrix defined in (4.17), but computed with polynomials of degree j . If the sample points are in general position and at least*

⁵Since the subspaces S_j are all different from each other, we assume that the normal vectors $\{\mathbf{b}_j\}_{j=1}^n$ are pairwise linearly independent.

$D - 1$ points correspond to each hyperplane, then:

$$\text{rank}(\mathbf{V}_j) \begin{cases} = M_j & j < n, \\ = M_j - 1 & j = n, \\ < M_j - 1 & j > n. \end{cases} \quad (4.18)$$

Therefore, the number n of hyperplanes is given by:

$$n = \min\{j : \text{rank}(\mathbf{V}_j) = M_j - 1\}. \quad (4.19)$$

In the presence of noise, one cannot directly estimate n from (4.19), because the matrix \mathbf{V}_j is always full rank. In this case, one can use the criterion (2.14) given in Chapter 2 to determine the rank.

Theorem 4.2 and the linear system in equation (4.17) allow us to determine the number of hyperplanes n and the vector of coefficients c_n , respectively, from sample points $\{\mathbf{x}_i\}_{i=1}^N$. The rest of the problem becomes now how to recover the normal vectors $\{\mathbf{b}_j\}_{j=1}^n$ from c_n . Imagine, for the time being, that we were given a set of n points $\{\mathbf{y}_j\}_{j=1}^n$, each one lying in only one of the n hyperplanes, that is $\mathbf{y}_j \in S_j$ for $j = 1, \dots, n$. Now let us consider the derivative of $p_n(\mathbf{x})$ evaluated at each \mathbf{y}_j . We have:

$$\nabla p_n(\mathbf{x}) = \frac{\partial p_n(\mathbf{x})}{\partial \mathbf{x}} = \frac{\partial}{\partial \mathbf{x}} \prod_{j=1}^n (\mathbf{b}_j^T \mathbf{x}) = \sum_{j=1}^n (\mathbf{b}_j) \prod_{\ell \neq j} (\mathbf{b}_\ell^T \mathbf{x}). \quad (4.20)$$

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

$$\mathbf{b}_j = \frac{\nabla p_n(\mathbf{y}_j)}{\|\nabla p_n(\mathbf{y}_j)\|}, \quad j = 1, 2, \dots, n. \quad (4.21)$$

Therefore, if we know one point in each one of the hyperplanes, the hyperplane segmentation problem can be solved analytically by simply evaluating the partial derivatives of $p_n(\mathbf{x})$ at each one of the points with known labels.

Consider now the case in which we do not know the membership of any of the data points. We now show that one can obtain one point per hyperplane by intersecting a random line with each one of the hyperplanes. To this end, consider a random line $L \doteq \{t\mathbf{v} + \mathbf{x}_0, t \in \mathbb{R}\}$ with direction \mathbf{v} and base point \mathbf{x}_0 . We can obtain one point in each hyperplane by intersecting L with the union of all the hyperplanes.⁶ Since at the intersection points we must have $p_n(t\mathbf{v} + \mathbf{x}_0) = 0$, the n points $\{\mathbf{y}_j\}_{j=1}^n$ can be obtained as

$$\mathbf{y}_j = t_j \mathbf{v} + \mathbf{x}_0, \quad j = 1, 2, \dots, n, \quad (4.22)$$

⁶Except when the chosen line is parallel to one of the hyperplanes, which corresponds to a zero-measure set of lines.

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

$$q_n(t) = p_n(t\mathbf{v} + \mathbf{x}_0) = \prod_{j=1}^n (t\mathbf{b}_j^T \mathbf{v} + \mathbf{b}_j^T \mathbf{x}_0) = 0. \quad (4.23)$$

We summarize our discussion so far as Algorithm 4.3 for segmenting hyperplanes.

Algorithm 4.3 (Algebraic Hyperplane Segmentation Algorithm).

Let $\{\mathbf{x}_i \in \mathbb{R}^D\}_{i=1}^N$ be a given collection of points clustering around an unknown number n of planes $\{\mathbf{b}_j\}_{j=1}^n$. The number of planes, the normal vectors, and the segmentation of the data can be determined as follows:

1. **Number of Hyperplanes.** Let \mathbf{V}_j be defined as in (4.17). Determine the number of groups as

$$n \doteq \min\{j : \text{rank}(\mathbf{V}_j) = M_j - 1\}. \quad (4.24)$$

2. **Normal Vectors.** Solve for \mathbf{c}_n from $\mathbf{V}_n \mathbf{c}_n = 0$ and set $p_n(\mathbf{x}) = \mathbf{c}_n^T \nu_n(\mathbf{x})$. Choose \mathbf{x}_0 and \mathbf{v} at random and compute the n roots $t_1, \dots, t_n \in \mathbb{R}$ of the univariate polynomial $q_n(t) = p_n(t\mathbf{v} + \mathbf{x}_0)$. Determine the normal vectors as

$$\mathbf{b}_j = \frac{\nabla p_n(\mathbf{y}_j)}{\|\nabla p_n(\mathbf{y}_j)\|}, \quad j = 1, 2, \dots, n, \quad (4.25)$$

where $\mathbf{y}_j = \mathbf{x}_0 + t_j \mathbf{v}$ is a point in the j th hyperplane.

3. **Segmentation.** Assign point \mathbf{x}_i to hyperplane $j = \arg \min_{l=1, \dots, n} (\mathbf{b}_l^T \mathbf{x}_i)^2$.
-

4.2 Knowing the Number of Subspaces

In this section, we derive a general solution to the subspace-segmentation problem (Problem 3.1) in the case in which the number of subspaces n is *known*. However, unlike the special cases we saw in the previous section, the dimensions of the subspaces can be different. In Section 4.2.1, we illustrate the basic ideas of dealing with subspaces of different dimensions via a simple example. Through Sections 4.2.2-4.2.4, we give detailed derivation and proof for the general solution. The final algorithm is summarized in Section 4.2.5.

4.2.1 An Introductory Example

To motivate and highlight the key ideas, in this section we study a simple example of clustering data points lying in subspaces of different dimensions in \mathbb{R}^3 : a line

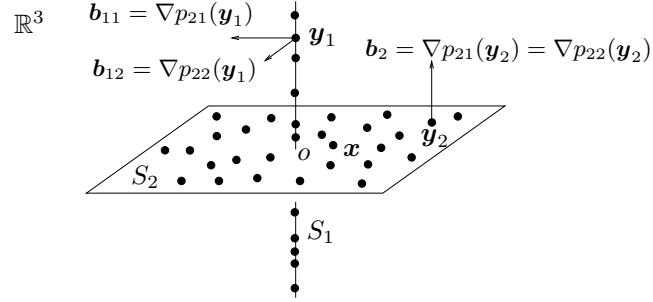


Figure 4.3. Data samples drawn from a union of one plane and one line (through the origin o) in \mathbb{R}^3 . The derivatives of the two vanishing polynomials $p_{21}(\mathbf{x}) = x_1x_2$ and $p_{22}(\mathbf{x}) = x_1x_3$ evaluated at a point \mathbf{y}_1 in the line give two normal vectors to the line. Similarly, the derivatives at a point \mathbf{y}_2 in the plane give the normal vector to the plane.

$S_1 = \{\mathbf{x} : x_1 = x_2 = 0\}$ and a plane $S_2 = \{\mathbf{x} : x_3 = 0\}$, as shown in Figure 4.3.

We can describe the union of these two subspaces as

$$S_1 \cup S_2 = \{\mathbf{x} : (x_1 = x_2 = 0) \vee (x_3 = 0)\} \quad (4.26)$$

$$= \{\mathbf{x} : (x_1x_3 = 0) \wedge (x_2x_3 = 0)\}. \quad (4.27)$$

Therefore, even though each individual subspace is described with polynomials of degree one (linear equations), the union of two subspaces is described with two polynomials of degree two, namely $p_{21}(\mathbf{x}) = x_1x_3$ and $p_{22}(\mathbf{x}) = x_2x_3$. In general, we can represent any two subspaces of \mathbb{R}^3 as the set of points satisfying a set of homogeneous polynomials of the form

$$c_1x_1^2 + c_2x_1x_2 + c_3x_1x_3 + c_4x_2^2 + c_5x_2x_3 + c_6x_3^2 = 0. \quad (4.28)$$

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

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

$$\nabla P_2(\mathbf{x}) = \begin{bmatrix} x_3 & 0 \\ 0 & x_3 \\ x_1 & x_2 \end{bmatrix} \implies \nabla P_2(\mathbf{y}_1) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix} \text{ and } \nabla P_2(\mathbf{y}_2) = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 1 \end{bmatrix}. \quad (4.29)$$

Then the columns of $\nabla P_2(\mathbf{y}_1)$ span the orthogonal complement to the first subspace S_1^\perp and the columns of $\nabla P_2(\mathbf{y}_2)$ span the orthogonal complement to the second subspace S_2^\perp (see Figure 4.3). Thus the dimension of the line is given by $d_1 = 3 - \text{rank}(\nabla P_2(\mathbf{y}_1)) = 1$ and the dimension of the plane is given by $d_2 = 3 - \text{rank}(\nabla P_2(\mathbf{y}_2)) = 2$. Therefore, if we are given one point in each sub-

space, we can obtain the *subspace bases* and their *dimensions* from the *derivatives of the polynomials* at the given points.

The final question is how to choose one representative point per subspace. With perfect data, we may choose a first point as any of the points in the data set. With noisy data, we may first define a distance from any point in \mathbb{R}^3 to the union of the subspaces,⁷ and then choose a point in the data set that minimizes this distance. Say we pick $\mathbf{y}_2 \in S_2$ as such point. We can then compute the normal vector $\mathbf{b}_2 = [0, 0, 1]^T$ to S_2 from $\nabla P(\mathbf{y}_2)$ as above. How do we now pick a second point in S_1 but not in S_2 ? As it turns out, this can be done by *polynomial division*. We can divide the original polynomials by $\mathbf{b}_2^T \mathbf{x}$ to obtain new polynomials of degree $n - 1 = 1$:

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

Since these new polynomials vanish on S_1 but not on S_2 , we can use them to define a new distance to S_1 only,⁸ and then find a point \mathbf{y}_1 in S_1 but not in S_2 as the point in the data set that minimizes this distance.

The next sections shows how this simple example can be systematically generalized to multiple subspaces of unknown and possibly different dimensions by *polynomial fitting* (Section 4.2.2), *differentiation* (Section 4.2.3), and *division* (Section 4.2.4).

4.2.2 Fitting Polynomials to Subspaces

Now consider a subspace arrangement $\mathcal{A} = \{S_1, S_2, \dots, S_n\}$ with $\dim(S_j) = d_j, j = 1, 2, \dots, n$. Let $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^N$ be a sufficiently large number of sample points in general position drawn from $Z_{\mathcal{A}} = S_1 \cup S_2 \cup \dots \cup S_n$. As we will know from Chapter B, the vanishing ideal $\mathcal{I}(Z_{\mathcal{A}})$, i.e., the set of all polynomials that vanish on $Z_{\mathcal{A}}$, is much more complicated than the special cases we studied earlier in this chapter.

Nevertheless, since we assume to know the number of subspaces n , we only have to consider the set of polynomials of degree n that vanish on $Z_{\mathcal{A}}$, i.e., the homogeneous component \mathcal{I}_n of $\mathcal{I}(Z_{\mathcal{A}})$. As we will know from the next chapter, these polynomials uniquely determine $Z_{\mathcal{A}}$.

Using the Veronese map, each polynomial in \mathcal{I}_n can be written as:

$$p_n(\mathbf{x}) = \mathbf{c}_n^T \nu_n(\mathbf{x}) = \sum c_{n_1, \dots, n_D} x_1^{n_1} \cdots x_D^{n_D} = 0, \quad (4.30)$$

where $c_{n_1, \dots, n_D} \in \mathbb{R}$ represents the coefficient of the monomial $\mathbf{x}^n = x_1^{n_1} \cdots x_D^{n_D}$. Although the polynomial equation is nonlinear in each data point \mathbf{x} , it is *linear* in the vector of coefficients \mathbf{c}_n . Indeed, since each polynomial $p_n(\mathbf{x}) = \mathbf{c}_n^T \nu_n(\mathbf{x})$ must be satisfied by every data point, we have $\mathbf{c}_n^T \nu_n(\mathbf{x}_i) = 0$

⁷For example, the squared algebraic distance to $S_1 \cup S_2$ is $p_{21}(\mathbf{x})^2 + p_{22}(\mathbf{x})^2 = (x_1^2 + x_2^2)x_3^2$.

⁸For example, the squared algebraic distance to S_1 is $p_{11}(\mathbf{x})^2 + p_{12}(\mathbf{x})^2 = x_1^2 + x_2^2$.

for all $i = 1, \dots, N$. Therefore, the vector of coefficients \mathbf{c}_n must satisfy the system of linear equations

$$\mathbf{V}_n(D) \mathbf{c}_n \doteq \begin{bmatrix} \nu_n(\mathbf{x}_1)^T \\ \nu_n(\mathbf{x}_2)^T \\ \vdots \\ \nu_n(\mathbf{x}_N)^T \end{bmatrix} \mathbf{c}_n = 0 \in \mathbb{R}^N, \quad (4.31)$$

where $\mathbf{V}_n(D) \in \mathbb{R}^{N \times M_n(D)}$ is called the *embedded data matrix*.

Clearly, the coefficient vector of every polynomial in \mathcal{I}_n is in the null space of the data matrix $\mathbf{V}_n(D)$. For every polynomial obtained from the null space of $\mathbf{V}_n(D)$ to be in \mathcal{I}_n , we need to have

$$\dim(\text{Null}(\mathbf{V}_n(D))) = \dim(\mathcal{I}_n) \doteq \mathcal{D}_n(Z_{\mathcal{A}}).$$

Or equivalently, the rank of the matrix $\mathbf{V}_n(D)$ needs to satisfy

$$\text{rank}(\mathbf{V}_n(D)) = \phi_{Z_{\mathcal{A}}}(n) = M_n(D) - \mathcal{D}_n(Z_{\mathcal{A}}) \quad (4.32)$$

in order that \mathcal{I}_n can be exactly recovered from the null space of $\mathbf{V}_n(D)$. As a result of the Algebraic Sampling Theory in Appendix A, the above rank condition is typically satisfied with $N \geq (M_n(D) - 1)$ data points in general position.⁹ A basis of \mathcal{I}_n ,

$$\mathcal{I}_n = \text{span}\{p_{n\ell}(\mathbf{x}), \ell = 1, \dots, \mathcal{D}_n\}, \quad (4.33)$$

can be computed from the set of \mathcal{D}_n singular vectors of $\mathbf{V}_n(D)$ associated with its \mathcal{D}_n zero singular values.

In the presence of moderate noise, we can still estimate the coefficients of each polynomial from the null space of $\mathbf{V}_n(D)$ using least-squares. However, we may not be able to directly determine the number of polynomials, because $\mathbf{V}_n(D)$ may be full rank. In this case, we can adopt the criterion (2.14) in Chapter 2 to determine the correct rank (and hence the null space) of the matrix $\mathbf{V}_n(D)$.

As discussed in Sections 2.3.1 and 2.3.2, the basic modeling assumption in NLPCA and KPCA is that there exists an embedding of the data into a higher-dimensional feature space \mathbf{F} such that the features live in a linear subspace of \mathbf{F} . However, there is no general methodology for finding the correct embedding for an arbitrary problem. Equation (4.31) shows that the commonly used polynomial embedding ν_n is the right one to use in NLPCA and KPCA when the data lives in an arrangement of subspaces, because the embedded data points $\{\nu_n(\mathbf{x}_i)\}_{i=1}^N$ indeed live in a subspace of $\mathbb{R}^{M_n(D)}$. Indeed, notice that each vector \mathbf{c}_n is simply a normal vector to the embedded subspace, as illustrated in Figure 4.4.

⁹For instance, it requires at least d_j points from each subspace S_j .

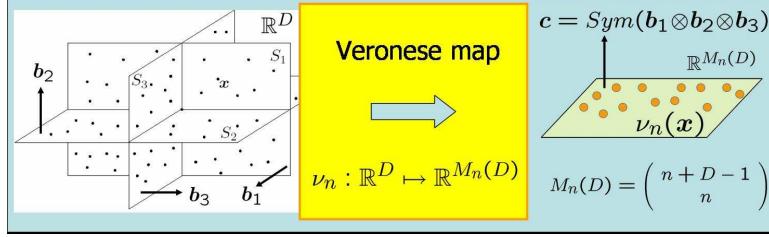


Figure 4.4. The polynomial embedding maps a union of subspaces of \mathbb{R}^D into a single subspace of $\mathbb{R}^{M_n(D)}$ whose normal vectors $\{c_n\}$ are the coefficients of the polynomials $\{p_n\}$ defining the subspaces. The normal vectors to the embedded subspace $\{c_n\}$ are related to the normal vectors to the original subspaces $\{b_j\}$ via the symmetric tensor product.

4.2.3 Subspaces from Polynomial Differentiation

Given a basis for the set of polynomials representing an arrangement of subspaces, we are now interested in determining a basis and the dimension of each subspace. In this section, we show that one can estimate the bases and the dimensions by differentiating all the polynomials $\{p_{n\ell}\}$ obtained from the null space of the embedded data matrix $V_n(D)$.

Consider a subspace S_j of dimension d_j in the subspace arrangement. Let $k_j = D - d_j$. Suppose that

$$B_j \doteq [b_1, b_2, \dots, b_{k_j}] \in \mathbb{R}^{D \times (k_j)}$$

is a set of base vectors for the orthogonal complement S_j^\perp of S_j . Then the vanishing ideal $\mathcal{I}(S_j)$ is generated by the set of linear forms $\{l_i = b_i^T x, i = 1, \dots, k_j = D - d_j\}$.

Let $p_n(x)$ be any polynomial in \mathcal{I}_n . Since $p_n \in \mathcal{I}(Z_A) \subset \mathcal{I}(S_j)$ and $\mathcal{I}(S_j)$ is generated by the linear forms l_i , p_n is of the form

$$p_n = l_1 h_1 + l_2 h_2 + \dots + l_{k_j} h_{k_j} \quad (4.34)$$

for some polynomials h_1, \dots, h_{k_j} . The derivative of p_n is

$$\nabla p_n = \sum_{i=1}^{k_j} h_i \nabla l_i + l_i \nabla h_i = \sum_{i=1}^{k_j} h_i b_i + l_i \nabla h_i. \quad (4.35)$$

Let y_j be a point in the subspace S_j but not in any other subspaces in the arrangement Z_A . Then $l_i(y_j) = 0, i = 1, \dots, k_j$. Thus, the derivative of p_n evaluated at y_j is a superposition of the vectors b_i :

$$\nabla p_n(y_j) = \sum_{i=1}^{k_j} h_i(y_j) b_i \in S_j^\perp. \quad (4.36)$$

This fact should come at no surprise. The zero set of each polynomial p_n is just a surface in \mathbb{R}^D , therefore its derivative at a regular point $y_j \in S_j$, $\nabla p_n(y_j)$, gives a vector orthogonal to the surface. Since an arrangement of subspaces is locally

flat, i.e., in a neighborhood of \mathbf{y}_j the surface is merely the subspace S_j , then the derivative at \mathbf{y}_j lives in the orthogonal complement S_j^\perp of S_j . By evaluating the derivatives of *all* the polynomials in \mathcal{I}_n at the same point \mathbf{y}_j we obtain a set of normal vectors that span the orthogonal complement of S_j . We summarize the above facts as Theorem 4.3. Figure 4.3 illustrates the theorem for the case of a plane and a line described in Section 4.2.1.

Theorem 4.3 (Subspace Bases and Dimensions by Polynomial Differentiation). *If the data set \mathbf{X} is such that $\dim(\text{Null}(\mathbf{V}_n(D))) = \dim(\mathcal{I}_n) = \mathcal{D}_n$ and one generic point \mathbf{y}_j is given for each subspace S_j , then we have*

$$S_j^\perp = \text{span}\left\{\frac{\partial}{\partial \mathbf{x}} \mathbf{c}_n^T \nu_n(\mathbf{x}) \Big|_{\mathbf{x}=\mathbf{y}_j}, \forall \mathbf{c}_n \in \text{Null}(\mathbf{V}_n(D))\right\}. \quad (4.37)$$

Therefore, the dimensions of the subspaces are given by

$$d_j = D - \text{rank}(\nabla P_n(\mathbf{y}_j)) \quad \text{for } j = 1, 2, \dots, n, \quad (4.38)$$

where $P_n(\mathbf{x}) \doteq [p_{n1}(\mathbf{x}), \dots, p_{n\mathcal{D}_n}(\mathbf{x})] \in \mathbb{R}^{1 \times \mathcal{D}_n}$ is a row of linearly independent polynomials in \mathcal{I}_n , and $\nabla P_n(\mathbf{x}) \doteq [\nabla p_{n1}(\mathbf{x}), \dots, \nabla p_{n\mathcal{D}_n}(\mathbf{x})] \in \mathbb{R}^{D \times \mathcal{D}_n}$.

Proof. (Sketch only). The fact that the derivatives span the entire normal space is the consequence of the general dimension theory for algebraic varieties [Bochnak et al., 1998, Harris, 1992, Eisenbud, 1996]. \square

Given \mathbf{c}_n , the computation of the derivative of $p_n(\mathbf{x}) = \mathbf{c}_n^T \nu_n(\mathbf{x})$ can be done algebraically:

$$\nabla p_n(\mathbf{x}) = \mathbf{c}_n^T \nabla \nu_n(\mathbf{x}) = \mathbf{c}_n^T E_n \nu_{n-1}(\mathbf{x}),$$

where $E_n \in \mathbb{R}^{M_n(D) \times M_{n-1}(D)}$ is a constant matrix containing only the exponents of the Veronese map $\nu_n(\mathbf{x})$. Thus, the computation does *not* involve taking derivatives of the (possibly noisy) data.

4.2.4 Point Selection via Polynomial Division

Theorem 4.3 suggests that one can obtain a basis for each S_j^\perp directly from the derivatives of the polynomials representing the union of the subspaces. However, in order to proceed we need to have one point per subspace, i.e., we need to know the vectors $\{\mathbf{y}_j\}_{j=1}^n$. In this section, we show how to select these n points in the *unsupervised learning scenario* in which we do not know the label for any of the data points.

In Section 4.1.3, we showed that in the case of hyperplanes, one can obtain one point per hyperplanes by intersecting a random line L with the union of all hyperplanes.¹⁰ This solution, however, does not generalize to subspaces of arbitrary

¹⁰This can always be done, except when the chosen line is parallel to one of the subspaces, which corresponds to a zero-measure set of lines.

dimensions. For instance, in the case of data lying in a line and a plane shown in Figure 4.3, a randomly chosen line L may not intersect the line. Furthermore, because polynomials in the null space of $\mathbf{V}_n(D)$ are no longer factorizable, their zero set is no longer a union of hyperplanes, hence the points of intersection with L may not lie in any of the subspaces.

In this section we propose an alternative algorithm for choosing one point per subspace. The idea is that we can always choose a point \mathbf{y}_n lying in one of the subspaces, say S_n , by checking that $P_n(\mathbf{y}_n) = 0$. Since we are given a set of data points $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^N$ lying in the subspaces, in principle we can choose \mathbf{y}_n to be any of the data points. However, in the presence of noise and outliers, a random choice of \mathbf{y}_n may be far from the true subspaces. One may be tempted to choose a point in the data set \mathbf{X} that minimizes $\|P_n(\mathbf{x})\|$, as we did in our introductory example in Section 4.2.1. However, such a choice has the following problems:

1. The value $\|P_n(\mathbf{x})\|$ is merely an *algebraic* error, i.e., it does not really represent the *geometric* distance from \mathbf{x} to its closest subspace. In principle, finding the geometric distance from \mathbf{x} to its closest subspace is a hard problem, because we do not know the normal bases $\{B_j\}_{j=1}^n$.
2. Points \mathbf{x} lying close to the intersection of two or more subspaces are more likely to be chosen, because two or more factors in $p_n(\mathbf{x}) = (\mathbf{b}_1^T \mathbf{x}) \cdots (\mathbf{b}_n^T \mathbf{x})$ are approximately zero, which yields a smaller value for $|p_n(\mathbf{x})|$. In fact, we can see from (4.36) that for an arbitrary \mathbf{x} in the intersection, the vector $\nabla p_n(\mathbf{x})$ needs to be a common normal vector to the two or more subspaces. If the subspaces have no common normal vector, then $\|\nabla p_n(\mathbf{x})\| = 0$. Thus, one should avoid choosing points close to the intersection, because they typically give very noisy estimates of the normal vectors.

We could avoid these two problems if we could compute the distance from each point to the subspace passing through it. However, we cannot compute such a distance yet because we do not know the subspace bases. The following lemma shows that we can compute a first order approximation to such a distance from P_n and its derivatives.

Lemma 4.4. *Let $\tilde{\mathbf{x}}$ be the projection of $\mathbf{x} \in \mathbb{R}^D$ onto its closest subspace. The Euclidean distance from \mathbf{x} to $\tilde{\mathbf{x}}$ is given by*

$$\|\mathbf{x} - \tilde{\mathbf{x}}\| = n \sqrt{P_n(\mathbf{x})(\nabla P_n(\mathbf{x})^T \nabla P_n(\mathbf{x}))^\dagger P_n(\mathbf{x})^T + O(\|\mathbf{x} - \tilde{\mathbf{x}}\|^2)},$$

where $P_n(\mathbf{x}) = [p_{n1}(\mathbf{x}), \dots, p_{nD_n}(\mathbf{x})] \in \mathbb{R}^{1 \times D_n}$ is a row vector with all the polynomials, $\nabla P_n(\mathbf{x}) = [\nabla p_{n1}(\mathbf{x}), \dots, \nabla p_{nD_n}(\mathbf{x})] \in \mathbb{R}^{D \times D_n}$, and A^\dagger is the Moore-Penrose inverse of A .

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

$$\min \|\tilde{\mathbf{x}} - \mathbf{x}\|^2, \quad \text{s.t. } p_{n\ell}(\tilde{\mathbf{x}}) = 0, \quad \ell = 1, 2, \dots, D_n. \quad (4.39)$$

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

$$\min_{\tilde{\mathbf{x}}, \lambda} \|\tilde{\mathbf{x}} - \mathbf{x}\|^2 + P_n(\tilde{\mathbf{x}})\lambda. \quad (4.40)$$

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

$$2(\tilde{\mathbf{x}} - \mathbf{x}) + \nabla P_n(\tilde{\mathbf{x}})\lambda = 0. \quad (4.41)$$

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

$$\lambda = 2(\nabla P_n(\tilde{\mathbf{x}})^T \nabla P_n(\tilde{\mathbf{x}}))^\dagger \nabla P_n(\tilde{\mathbf{x}})^T \mathbf{x}, \quad \|\tilde{\mathbf{x}} - \mathbf{x}\|^2 = \frac{1}{2} \mathbf{x}^T \nabla P_n(\tilde{\mathbf{x}}) \lambda, \quad (4.42)$$

where we have used the fact that $(\nabla P_n(\tilde{\mathbf{x}}))^T \tilde{\mathbf{x}} = 0$. After substituting the first equation into the second, we obtain that the squared distance from \mathbf{x} to its closest subspace can be expressed as

$$\|\tilde{\mathbf{x}} - \mathbf{x}\|^2 = \mathbf{x}^T \nabla P_n(\tilde{\mathbf{x}}) (\nabla P_n(\tilde{\mathbf{x}})^T \nabla P_n(\tilde{\mathbf{x}}))^\dagger \nabla P_n(\tilde{\mathbf{x}})^T \mathbf{x}. \quad (4.43)$$

After expanding in Taylor series about $\tilde{\mathbf{x}} = \mathbf{x}$, and noticing that $\nabla P_n(\mathbf{x})^T \mathbf{x} = n P_n(\mathbf{x})^T$ we obtain

$$\|\tilde{\mathbf{x}} - \mathbf{x}\|^2 \approx n^2 P_n(\mathbf{x}) (\nabla P_n(\mathbf{x})^T \nabla P_n(\mathbf{x}))^\dagger P_n(\mathbf{x})^T, \quad (4.44)$$

which completes the proof. \square

Thanks to Lemma 4.4, we can immediately choose a candidate \mathbf{y}_n lying in (close to) one of the subspaces and not in the intersection as

$$\mathbf{y}_n = \arg \min_{\mathbf{x} \in \mathbf{X}: \nabla P_n(\mathbf{x}) \neq 0} P_n(\mathbf{x}) (\nabla P_n(\mathbf{x})^T \nabla P_n(\mathbf{x}))^\dagger P_n(\mathbf{x})^T. \quad (4.45)$$

and compute a basis $B_n \in \mathbb{R}^{D \times (D-d_n)}$ for S_n^\perp by applying PCA to $\nabla P_n(\mathbf{y}_n)$.

In order to find a point \mathbf{y}_{n-1} lying in (close to) one of the remaining $(n-1)$ subspaces but not in (far from) S_n , we could in principle choose \mathbf{y}_{n-1} as in (4.45) after removing the points in S_n from the data set \mathbf{X} . With noisy data, however, this depends on a threshold and is not very robust. Alternatively, we can find a new set of polynomials $\{p_{(n-1)\ell}(\mathbf{x})\}$ defining the algebraic set $\cup_{j=1}^{n-1} S_j$. In the case of hyperplanes, there is only one such polynomial, namely

$$p_{n-1}(\mathbf{x}) \doteq (\mathbf{b}_1 \mathbf{x})(\mathbf{b}_2 \mathbf{x}) \cdots (\mathbf{b}_{n-1}^T \mathbf{x}) = \frac{p_n(\mathbf{x})}{\mathbf{b}_n^T \mathbf{x}} = \mathbf{c}_{n-1}^T \nu_{n-1}(\mathbf{x}).$$

Therefore, we can obtain $p_{n-1}(\mathbf{x})$ by *polynomial division*. Notice that dividing $p_n(\mathbf{x})$ by $\mathbf{b}_n^T \mathbf{x}$ is a *linear problem* of the form

$$R_n(\mathbf{b}_n) \mathbf{c}_{n-1} = \mathbf{c}_n, \quad (4.46)$$

where $R_n(\mathbf{b}_n) \in \mathbb{R}^{M_n(D) \times M_{n-1}(D)}$. This is because solving for the coefficients of $p_{n-1}(\mathbf{x})$ is equivalent to solving the equations $(\mathbf{b}_n^T \mathbf{x})(\mathbf{c}_{n-1}^T \nu_n(\mathbf{x})) = \mathbf{c}_n^T \nu_n(\mathbf{x})$ for all $\mathbf{x} \in \mathbb{R}^D$. These equations are obtained by equating the coefficients, and they are linear in \mathbf{c}_{n-1} , because \mathbf{b}_n and \mathbf{c}_n are already known.

Example 4.5 If $n = 2$ and $\mathbf{b}_2 = [b_1, b_2, b_3]^T$, then the matrix $R_2(\mathbf{b}_2)$ is given by

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

■

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

Theorem 4.6 (Choosing one Point per Subspace by Polynomial Division). *If the data set \mathbf{X} is such that $\dim(\text{null}(\mathbf{V}_n(D))) = \dim(\mathcal{I}_n)$, then the set of homogeneous polynomials of degree $(n - 1)$ associated with the algebraic set $\cup_{j=1}^{n-1} S_j$ is given by $\{\mathbf{c}_{n-1}^T v_{n-1}(\mathbf{x})\}$ where the vectors of coefficients $\mathbf{c}_{n-1} \in \mathbb{R}^{M_{n-1}(D)}$ must satisfy*

$$\mathbf{V}_n(D) R_n(\mathbf{b}_n) \mathbf{c}_{n-1} = 0, \quad \forall \mathbf{b}_n \in S_n^\perp. \quad (4.47)$$

Proof. We first show the necessity. That is, any polynomial of degree $n - 1$, $\mathbf{c}_{n-1}^T \nu_{n-1}(\mathbf{x})$, that vanishes on $\cup_{j=1}^{n-1} S_j$ satisfies the above equation. Since a point \mathbf{x} in the original algebraic set $\cup_{j=1}^n S_j$ belongs to either $\cup_{j=1}^{n-1} S_j$ or S_n , we have $\mathbf{c}_{n-1}^T \nu_{n-1}(\mathbf{x}) = 0$ or $\mathbf{b}_n^T \mathbf{x} = 0$ for all $\mathbf{b}_n \in S_n^\perp$. Hence $p_n(\mathbf{x}) = (\mathbf{c}_{n-1}^T \nu_{n-1}(\mathbf{x}))(\mathbf{b}_n^T \mathbf{x}) = 0$, and $p_n(\mathbf{x})$ must be a linear combination of polynomials in $P_n(\mathbf{x})$. If we denote $p_n(\mathbf{x})$ as $\mathbf{c}_n^T \nu_n(\mathbf{x})$, then the vector of coefficients \mathbf{c}_n must be in the null space of $\mathbf{V}_n(D)$. From $\mathbf{c}_n^T \nu_n(\mathbf{x}) = (\mathbf{c}_{n-1}^T \nu_{n-1}(\mathbf{x}))(\mathbf{b}_n^T \mathbf{x})$, the relationship between \mathbf{c}_n and \mathbf{c}_{n-1} can be written as $R_n(\mathbf{b}_n) \mathbf{c}_{n-1} = \mathbf{c}_n$. Since $\mathbf{V}_n(D) \mathbf{c}_n = 0$, \mathbf{c}_{n-1} needs to satisfy the following linear system of equations $\mathbf{V}_n(D) R_n(\mathbf{b}_n) \mathbf{c}_{n-1} = 0$.

We now show the sufficiency. That is, if \mathbf{c}_{n-1} is a solution to (4.47), then $\mathbf{c}_{n-1}^T \nu_{n-1}(\mathbf{x})$ is a homogeneous polynomial of degree $(n - 1)$ that vanishes on $\cup_{j=1}^{n-1} S_j$. Since \mathbf{c}_{n-1} is a solution to (4.47), then for all $\mathbf{b}_n \in S_n^\perp$ we have that $\mathbf{c}_n = R_n(\mathbf{b}_n) \mathbf{c}_{n-1}$ is in the null space of $\mathbf{V}_n(D)$. Now, from the construction of $R_n(\mathbf{b}_n)$, we also have that $\mathbf{c}_n^T \nu_n(\mathbf{x}) = (\mathbf{c}_{n-1}^T \nu_{n-1}(\mathbf{x}))(\mathbf{b}_n^T \mathbf{x})$. Hence, for every $\mathbf{x} \in \cup_{j=1}^{n-1} S_j$ but not in S_n , we have $\mathbf{c}_{n-1}^T \nu_{n-1}(\mathbf{x}) = 0$, because there is a \mathbf{b}_n such that $\mathbf{b}_n^T \mathbf{x} \neq 0$. Therefore, $\mathbf{c}_{n-1}^T \nu_{n-1}(\mathbf{x})$ is a homogeneous polynomial of degree $(n - 1)$ that vanishes on $\cup_{j=1}^{n-1} S_j$. □

Thanks to Theorem 4.6, we can obtain a basis $\{p_{(n-1)\ell}(\mathbf{x})\}_{\ell=1}^{\mathcal{D}_{n-1}}$ for the polynomials representing $\cup_{j=1}^{n-1} S_j$ from the intersection of the null spaces of $\mathbf{V}_n(D) R_n(\mathbf{b}_n) \in \mathbb{R}^{N \times M_{n-1}(D)}$ for all $\mathbf{b}_n \in S_n^\perp$. By evaluating the derivatives of the polynomials $p_{(n-1)\ell}$ we can obtain normal vectors to S_{n-1} and so on. By repeating these process, we can find a basis for each one of the remaining subspaces.

The overall subspaces estimation and segmentation process involves polynomial fitting, differentiation, and division.

4.2.5 The Basic Generalized PCA Algorithm

We summarize the results of this section with the following Generalized Principal Component Analysis (GPCA) algorithm for segmenting a known number of subspaces of unknown and possibly different dimensions from sample data points $\{\mathbf{x}_i\}_{i=1}^N$.

Algorithm 4.4 (GPCA: Generalized Principal Component Analysis).

```

set  $\mathbf{V}_n(D) \doteq [\nu_n(\mathbf{x}_1), \nu_n(\mathbf{x}_2), \dots, \nu_n(\mathbf{x}_N)]^T \in \mathbb{R}^{N \times M_n(D)}$ ;
for  $j = n : 1$ ,
    solve  $\mathbf{V}_j(D)\mathbf{c} = 0$  to obtain a basis  $\{\mathbf{c}_{j\ell}\}_{\ell=1}^{\mathcal{D}_j}$  of  $\text{null}(\mathbf{V}_j(D))$ , where the
        number of polynomials  $\mathcal{D}_j$  is obtained as in (4.32);
    set  $P_j(\mathbf{x}) = [p_{j1}(\mathbf{x}), \dots, p_{j\mathcal{D}_j}(\mathbf{x})] \in \mathbb{R}^{1 \times \mathcal{D}_j}$ , where  $p_{j\ell}(\mathbf{x}) =$ 
         $\mathbf{c}_{j\ell}^T \nu_j(\mathbf{x})$  for  $\ell = 1, \dots, \mathcal{D}_j$ ;
    compute
         $\mathbf{y}_j = \arg \min_{\mathbf{x} \in \mathbf{X}: \nabla P_j(\mathbf{x}) \neq 0} P_j(\mathbf{x})(\nabla P_j(\mathbf{x})^T \nabla P_j(\mathbf{x}))^\dagger P_j(\mathbf{x})^T;$ 
         $B_j \doteq [\mathbf{b}_{j1}, \mathbf{b}_{j2}, \dots, \mathbf{b}_{j(D-d_j)}] = \text{PCA}(\nabla P_j(\mathbf{y}_j));$ 
         $\mathbf{V}_{j-1}(D) = \mathbf{V}_j(D) [R_j^T(\mathbf{b}_{j1}), \dots, R_j^T(\mathbf{b}_{j(D-d_j)})]^T;$ 
end
for  $i = 1 : N$ ,
    assign point  $\mathbf{x}_i$  to subspace  $S_j$  if  $j = \arg \min_{\ell=1, \dots, n} \|B_\ell^T \mathbf{x}_i\|^2$ ;
end

```

Avoiding Polynomial Division.

In practice, we may avoid computing P_j for $j < n$ by using a heuristic distance function to choose the points $\{\mathbf{y}_j\}_{j=1}^n$ as follows. Since a point in $\cup_{\ell=j}^n S_\ell$ must satisfy $\|B_j^T \mathbf{x}\| \cdots \|B_n^T \mathbf{x}\| = 0$, we can choose a point \mathbf{y}_{j-1} on $\cup_{\ell=1}^{j-1} S_\ell$ as:

$$\mathbf{y}_{j-1} = \arg \min_{\mathbf{x} \in \mathbf{X}: \nabla P_n(\mathbf{x}) \neq 0} \frac{\sqrt{P_n(\mathbf{x})(\nabla P_n(\mathbf{x})^T \nabla P_n(\mathbf{x}))^\dagger P_n(\mathbf{x})^T} + \delta}{\|B_j^T \mathbf{x}\| \cdots \|B_n^T \mathbf{x}\| + \delta}, \quad (4.48)$$

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

4.3 Not Knowing the Number of Subspaces

The solution to the subspace-segmentation problem proposed in Section 4.2.5 assumes prior knowledge of the number of subspaces n . In practice, however, the number of subspaces n may not be known beforehand, hence we cannot estimate the polynomials representing the subspaces directly, because the linear system in (4.31) depends explicitly on n .

Earlier in Section 4.1, we have presented some special cases (e.g., arrangements of hyperplanes) for which one can recover the number of subspaces from data. In this section, we show that by exploiting the algebraic structure of the vanishing ideals of subspace arrangements it is possible to simultaneously recover the number of subspaces, together with their dimensions and their bases. As usual, we first examine some subtlety with determining the number of subspaces via two simple examples in Section 4.3.1 and illustrate the key ideas. Section 4.3.2 considers the case of *perfect subspace arrangements* in which all subspaces are of equal dimension $d = d_1 = \dots = d_n$. We derive a set of rank constraints on the data from which one can estimate the n and d . Section 4.3.3 considers the most general case of subspaces of different dimensions and shows that n and d can be computed in a recursive fashion by first fitting subspaces of larger dimensions and then further segmenting these subspaces into subspaces of smaller dimensions.

4.3.1 Introductory Examples

Imagine we are given a set of points $\mathbf{X} = \{\mathbf{x}_i\}$ lying in two lines in \mathbb{R}^3 , say

$$S_1 = \{\mathbf{x} : x_2 = x_3 = 0\} \quad \text{and} \quad S_2 = \{\mathbf{x} : x_1 = x_3 = 0\}. \quad (4.49)$$

If we form the matrix of embedded data points $\mathbf{V}_n(D)$ for $n = 1$ and $n = 2$, respectively:

$$\mathbf{V}_1(3) = \begin{bmatrix} \vdots & & \vdots \\ x_1 & x_2 & x_3 \\ \vdots & & \vdots \end{bmatrix} \quad \text{and} \quad \mathbf{V}_2(3) = \begin{bmatrix} \vdots & & & & \vdots \\ x_1^2 & x_1x_2 & x_1x_3 & x_2^2 & x_2x_3 & x_3^2 \\ \vdots & & & & & \vdots \end{bmatrix},$$

we obtain $\text{rank}(\mathbf{V}_1(3)) = 2 < 3$ and $\text{rank}(\mathbf{V}_2(3)) = 2 < 6$.¹¹ Therefore, we cannot determine the number of subspaces as the degree n such that the matrix $\mathbf{V}_n(D)$ drops rank (as we did in Section 4.1.3 for the case of hyperplanes), because we would obtain $n = 1$ which is not the correct number of subspaces.

How do we determine the correct number of subspaces in this case? As discussed in Section 3.2.2, a linear projection onto a low-dimensional subspace preserves the number and dimensions of the subspaces. In our example, if we project the data onto the plane $P = \{\mathbf{x} : x_1 + x_2 + x_3 = 0\}$ and then embed the

¹¹The reader is encouraged to verify these facts numerically and do the same for the examples in the rest of this section.

projected data we obtain

$$\mathbf{V}_1(2) = \begin{bmatrix} \vdots & \vdots \\ x_1 & x_2 \\ \vdots & \vdots \end{bmatrix} \quad \text{and} \quad \mathbf{V}_2(2) = \begin{bmatrix} \vdots & & \vdots \\ x_1^2 & x_1 x_2 & x_2^2 \\ \vdots & & \vdots \end{bmatrix}.$$

In this case $\text{rank}(\mathbf{V}_1(2)) = 2 < 2$, but $\text{rank}(\mathbf{V}_2(2)) = 2 < 3$. Therefore, the first time the matrix $\mathbf{V}_n(d+1)$ drops rank is when $n = 2$ and $d = 1$. This suggests that, as we will formally show in Section 4.3.2, when the subspaces are of equal dimension one can determine d and n as the minimum values for which there are a projection onto a $d+1$ -dimensional subspace such that the matrix $\mathbf{V}_n(d+1)$ drops rank.

Unfortunately, the situation is not so simple for subspaces of different dimensions. Imagine now that in addition to the two lines S_1 and S_2 we are also given data points on a plane $S_3 = \{\mathbf{x} : x_1 + x_2 = 0\}$ (so that the overall configuration is similar to that shown in Figure 1.2). In this case we have $\text{rank}(\mathbf{V}_1(3)) = 3 < 3$, $\text{rank}(\mathbf{V}_2(3)) = 5 < 6$, and $\text{rank}(\mathbf{V}_3(3)) = 6 < 10$. Therefore, if we try to determine the number of subspaces as the degree of the embedding for which the embedded data matrix drops rank we would obtain $n = 2$, which is incorrect again. The reason for this is clear: we can either fit the data with one polynomial of degree $n = 2$, which corresponds to the plane S_3 and the plane P spanned by the two lines, or we can fit the data with four polynomials of degree $n = 3$, which vanish precisely on the two lines S_1 , S_2 , and the plane S_3 .

In cases like this, one needs to resort to a more sophisticated algebraic process to identify the correct number of subspaces. As in the previous example, we can first search for the minimum degree n and dimension d such that $\mathbf{V}_n(d+1)$ drops rank. In our example, we obtain $n = 2$ and $d = 2$. By applying the GPCA algorithm to this data set we will partition it into two planes P and S_3 . Once the two planes have been estimated, we can reapply the same process to each plane. The plane P will be separated into two lines S_1 and S_2 , as described in the previous example, while the plane S_3 will remain unchanged. This recursive process stops when every subspace obtained can no longer be separated into lower-dimensional subspaces. We will a more detailed description of this recursive GPCA algorithm in Section 4.3.3.

4.3.2 Segmenting Subspaces of Equal Dimension

In this section, we derive explicit formulae for the number of subspaces n and their dimensions $\{d_j\}$ in the case of subspaces of equal dimension $d = d_1 = d_2 = \dots = d_n$. Notice that this is a generalized version to the two-lines example that we discussed in the previous section. In the literature, arrangements of subspaces of equal dimensions, are called *pure arrangements*. This type of arrangements are important for a wide range of applications, such as motion segmentation in computer vision [?, ?, ?, ?], pattern recognition [?, ?], as well as identification of hybrid linear systems [?, ?].

Theorem 4.7 (Subspaces of Equal Dimension). *Let $\{\mathbf{x}_i\}_{i=1}^N$ be a given collection of $N \geq M_n(d+1) - 1$ sample points lying in n different d -dimensional subspaces of \mathbb{R}^D . Let $\mathbf{V}_j(\ell + 1) \in \mathbb{R}^{N \times M_j(\ell+1)}$ be the embedded data matrix defined in (4.31), but computed with the Veronese map ν_j of degree j applied to the data projected onto a generic $(\ell+1)$ -dimensional subspace of \mathbb{R}^D . If the sample points are in general position and at least d points are drawn from each subspace, then the dimension of the subspaces is given by:*

$$d = \min\{\ell : \exists j \geq 1 \text{ such that } \text{rank}(\mathbf{V}_j(\ell + 1)) < M_j(\ell + 1)\}, \quad (4.50)$$

and the number of subspaces can be obtained as:

$$n = \min\{j : \text{rank}(\mathbf{V}_j(d + 1)) = M_j(d + 1) - 1\}. \quad (4.51)$$

Proof. For simplicity, we divide the proof into the following three cases:

Case 1: d known

Imagine for a moment that d was known, and that we wanted to compute n only. Since d is known, following our analysis in Section 3.2.2, we can first project the data onto a $(d + 1)$ -dimensional space $P \subset \mathbb{R}^D$ so that they become n d -dimensional hyperplanes in P (see Theorem 3.8). Now compute the matrix $\mathbf{V}_j(d + 1)$ as in (4.31) by applying the Veronese map of degree $j = 1, 2, \dots$ to the projected data. From our analysis in Section 4.1.3, there is a unique polynomial of degree n representing the union of the projected subspaces and the coefficients of this polynomial must lie in the null space of $\mathbf{V}_n(d + 1)$. Thus, given $N \geq M_n(d + 1) - 1$ points in general position, with at least d points in each subspace, we have that $\text{rank}(\mathbf{V}_n(d + 1)) = M_n(d + 1) - 1$. Furthermore, there cannot be a polynomial of degree less than n that is satisfied by all the data,¹² hence $\text{rank}(\mathbf{V}_j(d + 1)) = M_j(d + 1)$ for $j < n$. Consequently, if d is known, we can compute n by first projecting the data onto a $(d + 1)$ -dimensional space and then obtain

$$n = \min\{j : \text{rank}(\mathbf{V}_j(d + 1)) = M_j(d + 1) - 1\}. \quad (4.52)$$

Case 2: n known

Consider now the opposite case in which n is known, but d is unknown. Let $\mathbf{V}_n(\ell + 1)$ be defined as in (4.31), but computed from the data projected onto a generic $(\ell + 1)$ -dimensional subspace of \mathbb{R}^D . When $\ell < d$, we have a collection of $(\ell + 1)$ -dimensional subspaces in an $(\ell + 1)$ -dimensional space, which implies that $\mathbf{V}_n(\ell + 1)$ must be full rank. If $\ell = d$, then from equation (4.52) we have that $\text{rank}(\mathbf{V}_n(\ell + 1)) = M_n(\ell + 1) - 1$. When $\ell > d$, then equation (4.31) has more than one solution, thus $\text{rank}(\mathbf{V}_n(\ell + 1)) < M_n(\ell + 1) - 1$. Therefore, if n is known, we can compute d as

$$d = \min\{\ell : \text{rank}(\mathbf{V}_n(\ell + 1)) = M_n(\ell + 1) - 1\}. \quad (4.53)$$

¹²This is guaranteed by the algebraic sampling theorem in Appendix A.

Case 3: n and d unknown

We are left with the case in which both n and d are unknown. As before, if $\ell < d$ then $\mathbf{V}_j(\ell+1)$ is full rank for all j . When $\ell = d$, $\mathbf{V}_j(\ell+1)$ is full rank for $j < n$, drops rank by one if $j = n$ and drops rank by more than one if $j > n$. Thus one can set d to be the smallest integer ℓ for which there exist an j such that $\mathbf{V}_j(\ell+1)$ drops rank, that is

$$d = \min\{\ell : \exists j \geq 1 \text{ such that } \text{rank}(\mathbf{V}_j(\ell+1)) < M_j(\ell+1)\}.$$

Given d one can compute n as in equation (4.52). □

Therefore, in principle, both n and d can be retrieved if sufficient data points are drawn from the subspaces. The subspace-segmentation problem can be subsequently solved by first projecting the data onto a $(d+1)$ -dimensional subspace and then applying the GPCA algorithm (Algorithm 4.4) to the projected data points.

In the presence of noise, one may not be able to estimate d and n from equations (4.50) and (4.51), respectively, because the matrix $\mathbf{V}_j(\ell+1)$ may be full rank for all j and ℓ . As before, we can use the criterion (2.14) of Chapter 2 to determine the rank of $\mathbf{V}_j(\ell+1)$. However, in practice this requires to search for up to possibly $(D-1)$ values for d and $\lceil N/(D-1) \rceil$ values for n . In our experience, the rank conditions work well when either d or n are known. There are still many open issues in the problem of finding a good search strategy and model selection criterion for n and k when both of them are unknown. Some of these issues will be discussed in more detail in Chapter 5

4.3.3 Segmenting Subspaces of Different Dimensions

In this section, we consider the problem of segmenting an unknown number of subspaces of unknown and possibly different dimensions from sample points.

First of all, we notice that the simultaneous recovery of the number and dimensions of the subspaces may be an ill-conditioned problem if we are not clear about what we are looking for. For example, in the extreme cases, one may interpret the sample set \mathbf{X} as N 1-dimensional subspaces, with each subspace spanned by each one of the sample points $\mathbf{x} \in \mathbf{X}$; or one may view the whole \mathbf{X} as belonging to one D -dimensional subspace, i.e., \mathbb{R}^D itself.

Although the above two trivial solutions can be easily rejected by imposing some conditions on the solutions,¹³ other more difficult ambiguities may also arise in cases such as that of Figure 1.2 in which two lines and a plane can also be interpreted as two planes. More generally, when the subspaces are of different dimensions one may not be able to determine the number of subspaces directly

¹³To reject the N -lines solution, one can put a cap on the maximum number of groups n_{max} ; and to reject \mathbb{R}^D as the solution, one can simply require that the maximum dimension of every subspace is strictly less than D .

from the degree of the polynomials fitting the data, because the degree of the polynomial of minimum degree that fits a collection of subspaces is always less than or equal to the number of subspaces.

To resolve the difficulty in determining the number and dimension of subspaces, notice that the *algebraic set* $Z_{\mathcal{A}} = \cup_{j=1}^n S_j$ can be decomposed into irreducible subsets S_j 's – an irreducible algebraic set is also called a *variety*. The decomposition of Z into $\{S_j\}_{j=1}^n$ is always unique. Therefore, as long as we are able to correctly determine from the given sample points the underlying algebraic set $Z_{\mathcal{A}}$ or the associated (radical) ideal $\mathcal{I}(Z_{\mathcal{A}})$, in principle the number of subspaces n and their dimensions $\{d_j\}_{j=1}^n$ can always be uniquely determined in a purely algebraic fashion. In Figure 1.2, for instance, the first interpretation (2 lines and 1 plane) would be the right one and the second one (2 planes) would be incorrect, because the two lines, which span one of the planes, is not an irreducible algebraic set.

Having established that the problem of subspace segmentation is equivalent to decomposing the algebraic ideal associated with the subspaces, we are left with deriving a computable scheme to achieve the goal.

From every homogeneous component \mathcal{I}_i of

$$\mathcal{I}(Z_{\mathcal{A}}) = \mathcal{I}_m \oplus \mathcal{I}_{m+1} \oplus \cdots \oplus \mathcal{I}_n \oplus \cdots,$$

we may compute a subspace arrangement Z_i such that $Z_{\mathcal{A}} \subseteq Z_i$ is a subspace embedding (see Section B.2). For each $i \geq m$, we can evaluate the derivatives of polynomials in \mathcal{I}_i on subspace S_j and denote the collection of derivatives as

$$D_{i,j} \doteq \cup_{\mathbf{x} \in S_j} \{\nabla f |_{\mathbf{x}}, \forall f \in \mathcal{I}_i\}, \quad j = 1, 2, \dots, n. \quad (4.54)$$

Obviously, we have the following relationship:

$$D_{i,j} \subseteq D_{i+1,j} \subseteq S_j^\perp, \quad \forall i \geq m. \quad (4.55)$$

Then for each \mathcal{I}_i , we can define a new subspace arrangement as

$$Z_i \doteq D_{i,1}^\perp \cup D_{i,2}^\perp \cup \cdots \cup D_{i,n}^\perp. \quad (4.56)$$

Notice that it is possible that $D_{i,j} = D_{i,j'}$ for different j and j' and Z_i contains less than n subspaces. We summarize the above derivation as the following theorem.

Theorem 4.8 (A Filtration of Subspace Arrangements). *Let $\mathcal{I}(Z_{\mathcal{A}}) = \mathcal{I}_m \oplus \mathcal{I}_{m+1} \oplus \cdots \oplus \mathcal{I}_n \oplus \cdots$ be the ideal of a subspace arrangement $Z_{\mathcal{A}}$. Let Z_i be the subspace arrangement defined by the derivatives of \mathcal{I}_i , $i \geq m$ as above. Then we obtain a filtration of subspace arrangements:*

$$Z_m \supseteq Z_{m+1} \supseteq \cdots \supseteq Z_n = Z_{\mathcal{A}},$$

and each subspace of $Z_{\mathcal{A}}$ is embedded in one of the subspaces of Z_i .

The above theorem naturally leads to a recursive scheme that allows us to determine the correct number and dimensions of the subspaces in $Z_{\mathcal{A}}$. Specifically, we start with $n = 1$ and increase n until there is at least one polynomial of degree

n fitting all the data, i.e., until the matrix $\mathbf{V}_n(D)$ drops rank for the first time. For such an n , we can use Algorithm 4.4 to separate the data into n subspaces. Then we can further separate each one of these n groups of points using the same procedure. The stopping criterion for the recursion is when all the groups cannot be further separated or the number of groups n reaches some n_{max} .¹⁴

4.3.4 The Recursive GPCA Algorithm

To summarize the above discussions, in principle we can use the following algorithm to recursively identify and segment subspaces in a subspace arrangement Z from a set of samples \mathbf{X} .

```

Function Recursive-GPCA( $\mathbf{X}$ )
 $n = 1;$ 
repeat
    build a data matrix  $\mathbf{V}_n(D) \doteq [\nu_n(\mathbf{x}_1), \dots, \nu_n(\mathbf{x}_N)]^T \in \mathbb{R}^{M_n(D) \times N}$  via the
    Veronese map  $\nu_n$  of degree  $n$ ;
    if  $\text{rank}(\mathbf{V}_n(D)) < M_n(D)$  then
        compute the basis  $\{\mathbf{c}_{n\ell}\}$  of the null space of  $\mathbf{V}_n(D)$ ;
        obtain polynomials  $\{p_{n\ell}(\mathbf{x}) \doteq \mathbf{c}_{n\ell}^T \nu_n(\mathbf{x})\}$ ;
         $\mathbf{Y} = \emptyset$ ;
        for  $j = 1 : n$  do
            select a point  $\mathbf{x}_j$  from  $\mathbf{X} \setminus \mathbf{Y}$ ;
            obtain the subspace  $S_j$  spanned by the derivatives  $\text{span}\{\nabla p_{n\ell}(\mathbf{x}_j)\}$ ;
            find the subset  $\mathbf{X}_j \subset \mathbf{X}$  that belong to the subspace  $S_j$ ;
             $\mathbf{Y} \leftarrow \mathbf{Y} \cup \mathbf{X}_j$ ;
            Recursive-GPCA( $\mathbf{X}_j$ ); (with  $S_j$  now as the ambient space)
        end for
         $n \leftarrow n_{max}$ ;
    else
         $n \leftarrow n + 1$ ;
    end if
until  $n \geq n_{max}$ .

```

Figure 4.5 shows the result of applying the Recursive GPCA algorithm to a set of points sampled from two lines and one plane. The points are corrupted by about 5% noises and the algorithm seems to tolerate them well. The appearance of a third “ghost” line in the final solution clearly illustrates the recursive segmentation process: points at the intersection of the two planes segmented at the highest level get assigned to both planes depending on the distances, which are random due to the noises.

¹⁴For example, the inequality $M_n(D) \leq N$ imposes a constraint on the maximum possible number of groups n_{max} .

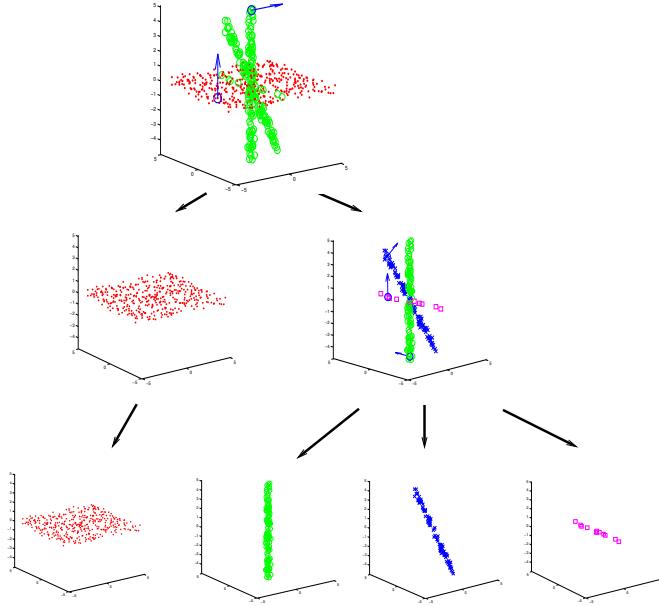


Figure 4.5. Recursive segmentation of the data points on one plane and two lines. Initially the points are partitioned into two planes. Then one of the planes is further partitioned into three lines. Notice that a “ghost” line appears, because points on the intersection of the two planes form a new line. The arrows on each subspace indicate the normal vectors.

4.4 Relationships between GPCA, K-Subspaces, and EM

In Section 3.3.3, we have discussed the relationships between K-subspaces and EM. In this section, we reveal their relationships with GPCA through the special case of hyperplane arrangements. Let \mathbf{b}_j be the normal vectors to an arrangement of hyperplanes S_j , $j = 1, \dots, n$, respectively.

We know from Chapter 3 that, under reasonable assumptions, both the K-subspaces and the EM methods minimize an objective of the form

$$\min_{\{\mathbf{b}_j\}} \sum_{i=1}^N \sum_{j=1}^n w_{ij} \|\mathbf{b}_j^T \mathbf{x}_i\|^2. \quad (4.57)$$

In the case of K-subspaces, w_{ij} is a “hard” assignment of \mathbf{x}_i to the subspaces: $w_{ij} = 1$ only if $\mathbf{x}_i \in S_j$ and 0 otherwise. The above objective function becomes exactly the geometric modeling error. In the case of EM, $w_{ij} \in [0, 1]$ is the probability of the latent random variable $z_i = j$ given \mathbf{x}_i . Then w_{ij} plays the role as a “soft” assignment of \mathbf{x}_i to group j .

Following the same line of reasoning, we can replace w_{ij} with an even “softer” assignment of membership:

$$w_{ij} \doteq \frac{1}{n} \prod_{l \neq j} \|b_l^T \mathbf{x}_i\|^2 \in \mathbb{R}. \quad (4.58)$$

Notice that, in general, the value of w_{ij} is large when \mathbf{x}_i belongs to (or is close to) S_j , and the value is small when \mathbf{x}_i belongs to (or is close to) any other subspace. With this choice of w_{ij} , the objective function becomes

$$\min_{\{\mathbf{b}_j\}} \sum_{i=1}^N \sum_{j=1}^n \left(\frac{1}{n} \prod_{l \neq j} \|b_l^T \mathbf{x}_i\|^2 \right) \|b_j^T \mathbf{x}_i\|^2 = \sum_{i=1}^N \prod_{j=1}^n \|b_j^T \mathbf{x}_i\|^2. \quad (4.59)$$

This is exactly the objective function that all the algebraic methods are based upon. To see this, notice that

$$\sum_{i=1}^N \prod_{j=1}^n \|b_j^T \mathbf{x}_i\|^2 = \sum_{i=1}^N p_n(\mathbf{x}_i)^2 = \sum_{i=1}^N (\mathbf{c}_n^T \nu_n(\mathbf{x}_i))^2. \quad (4.60)$$

Not so surprisingly, we end up with a “least-squares like” formulation in terms of the embedded data $\nu_n(\mathbf{x})$ and the coefficient vector \mathbf{c}_n . Notice that the above objective function can be rewritten as

$$\sum_{i=1}^N (\mathbf{c}_n^T \nu_n(\mathbf{x}_i))^2 = \|\mathbf{V}_n(D) \mathbf{c}_n\|^2. \quad (4.61)$$

The least-squares solution of \mathbf{c}_n is exactly given by the eigenvector associated with the smallest eigenvalue of the matrix $\mathbf{V}_n(D)$.

The K-subspaces or EM methods minimizes its objective iteratively using \mathbf{b}_j computed in the previous iteration. However, one key observation in the GPCA algorithm is that the derivative of the vanishing polynomial $p_n(\mathbf{x}) = \mathbf{c}_n^T \nu_n(\mathbf{x})$ at the sample points provide information about the normal vectors \mathbf{b}_j . Therefore, the GPCA algorithm does not require initialization and iteration but still achieves a goal similar to that of K-subspaces or EM.

4.5 Bibliographic Notes

The difficulty with initialization for the iterative clustering algorithms that we have presented in the previous chapter has motivated the recent development of algebro-geometric approaches to subspace segmentation that do *not* require initialization. [Kanatani, 2001, Boult and Brown, 1991, Costeira and Kanade, 1998] demonstrated that when the subspaces are orthogonal, of equal dimensions, and with trivial intersection, one can use the SVD of the data to define a similarity matrix from which the segmentation of the data can be obtained using spectral clustering techniques. Unfortunately, this method is sensitive to noise in the data,

as pointed out in [Kanatani, 2001, Wu et al., 2001], where various improvements are proposed. When the intersection of the subspaces is nontrivial, the segmentation of the data is usually obtained in an *ad-hoc* fashion again using clustering algorithms such as K-means. A basis for each subspace is then obtained by applying PCA to each group. For the special case of two planes in \mathbb{R}^3 , a geometric solution was developed by [Shizawa and Mase, 1991] in the context of segmentation of 2-D transparent motions. In the case of subspaces of co-dimension one, i.e., *hyperplanes*, an algebraic solution was developed by [Vidal et al., 2003b], where the hyperplane clustering problem is shown to be equivalent to homogeneous polynomial factorization.

The GPCA algorithm for the most general case¹⁵ was later developed in [Vidal et al., 2004]; and the decomposition of the polynomial(s) was based on differentiation, a numerically better-conditioned operation. The GPCA algorithm was successfully applied to solve the motion segmentation problem in computer vision [Vidal and Ma, 2004]. The generalization to arrangements of both linear and quadratic surfaces was first studied by [?], which we will present in Chapter 12. Robustness issues of the algorithm were addressed in the paper of [Huang et al., 2004], and Chapter 5 will discuss them in more detail.

4.6 Exercises

Exercise 4.1 For each $f = 1, \dots, F$, let $\{\mathbf{x}_{fi} \in \mathbb{R}^D\}_{i=1}^N$ be a collection of N points lying in n hyperplanes with normal vectors $\{\mathbf{b}_{fj}\}_{j=1}^n$. Assume that $\mathbf{x}_{1i}, \mathbf{x}_{2i}, \dots, \mathbf{x}_{Fi}$ corresponds to each other, i.e., for each $i = 1, \dots, N$ there is a $j = 1, \dots, n$ such that for all $f = 1, \dots, F$, we have $\mathbf{b}_{fj}^\top \mathbf{x}_{1i} = 0$. Propose an extension of the GPCA algorithm that computes the normal vectors in such a way that $\mathbf{b}_{1j}, \mathbf{b}_{2j}, \dots, \mathbf{b}_{Fj}$ correspond to each other.

Hint: If $p_{fn}(\mathbf{x}) = \mathbf{c}_f^\top \nu_n(\mathbf{x}) = (\mathbf{b}_{f1}^\top \mathbf{x}) \cdots (\mathbf{b}_{fn}^\top \mathbf{x})$ and the i th set of points $\mathbf{x}_{1i}, \mathbf{x}_{2i}, \dots, \mathbf{x}_{Fi}$ corresponds to the j th group of hyperplanes, then $\mathbf{b}_{fj} \sim \nabla p_{fn}(\mathbf{x}_{fi})$.

¹⁵That is, an arbitrary number of subspaces of arbitrary dimensions.

Chapter 5

Statistical Techniques and Robustness Issues

The GPCA algorithms developed in the previous chapter are based on purely (linear) algebraic techniques. In practice, these techniques can tolerate moderate noise in the data or numerical error in the computation. However, they are not designed to deal with large amount of noises or outliers in the data. In addition, many of the statistic properties of the subspace-segmentation problem are not sufficiently harnessed so as to improve the performance of the GPCA algorithm. In this chapter, we switch the gears a little bit and show how to incorporate various statistical techniques with the algebraic GPCA algorithm so as to improve its numerical stability and statistical robustness.

As there are usually many different ways to improve multiple aspects of the GPCA algorithm, it is impossible for us to examine here in detail every possible scenario. Instead, we decide to select a few representative statistical techniques and demonstrate how to use them to improve GPCA:

1. The vanishing polynomials estimated using the least-square fitting in the basic GPCA are not necessarily the optimal ones for segmenting the data points into the subspaces. In Section 5.1, we show how certain techniques from *linear discriminant analysis* can help to improve the polynomials estimated.
2. The normal vectors estimated using the derivatives of the vanishing polynomials at only one point per subspace can certainly be improved if they can be estimated collectively from all the points on the same subspace. However, we must deal with the difficulty that we do not know which points belong to the same subspace. In Section 5.2, we show how this difficulty can be resolved using the *majority voting techniques* in statistics.

3. We have seen in the previous chapter (as well as in Appendix B) that one can determine the number of subspaces and their dimensions from a sufficient number of samples via purely algebraic means. However, in practice, when the data are corrupted with noise, it becomes a rather challenging problem to determine the correct number of subspaces and their dimensions. In Section 5.3, we show how to adopt proper *model-selection criterion* for subspace arrangements.
4. Real data are often corrupted with outliers. In Section 5.4, we introduce some *robust statistical techniques* that have been shown to be effective for the subspace-segmentation problem in the presence of a significant amount (say up to 50%) of outliers in the sample data. In particular, we examine the influence function method, multivariate trimming, and random sample consensus (RANSAC).

5.1 Discriminant Analysis

Just like PCA, GPCA aims to find multiple subspaces that best fit the (embedded) data samples. It is based on the notion that we are able to correctly identify *all* the polynomials that fit the data. However, from a practical viewpoint, identifying all the polynomials simultaneously from noisy data can be a very difficult problem. When the data is noisy, it sometimes can be very difficult even to determine the number of polynomials from the spectrum of the matrix $V_n(D)$. In general, if the number of polynomials were under-estimated, the resulting subspaces would over-fit the data;¹ and if the number of polynomials were over-estimated, the resulting subspaces would under-fit the data.

Obviously, both over-fitting and under-fitting result in incorrect estimates for the subspaces. However, do they necessarily result in equally bad segmentation of the data? The answer is *no*. Between over-fitting and under-fitting, we actually would favor over-fitting. The reason is that, though over-fitting results in subspaces that are larger than the original subspaces, but it is a zero-measure event that any over-estimated subspace contains simultaneously more than one original subspace. Thus, the grouping of the data points may still be correct. For instance, consider the extreme case that we choose only one polynomial that fits the data, then the derivatives of the polynomial, evaluated at one point per subspace, lead to n hyperplanes. Nevertheless, these over-fitting hyperplanes will in general result in a correct grouping of the data points. One can verify this with the introductory example we discussed in Section 4.2.1. Either of the two polynomials $p_{21}(\mathbf{x}) = x_1x_3$ and $p_{22}(\mathbf{x}) = x_2x_3$ leads to two hyperplanes that segment the line and the plane correctly.

However, the GPCA algorithm is not designed for finding the polynomial(s) that are best at *discriminating* between noisy data samples that belong to different

¹That is, the dimensions of some of the subspaces would be larger than the true ones.

subspaces. From a statistical viewpoint, the polynomial that best fits the embedded data points is not necessarily the one that best clusters the data points into the subspaces. If the distributions of the data points are known (e.g., mixtures of Gaussians), many techniques from discriminant analysis in statistics can offer principled solutions to the optimal clustering of the data points. These techniques include Linear Discriminant Analysis (LDA), Quadratic Discriminant Analysis (QDA), and Regularized Discriminant Analysis (RDA) etc. [Hastie, 1984]. It is beyond the scope of this book to discuss all such techniques.² In this section, we show how we can adopt the concepts from Fisher Linear Discriminant Analysis (LDA)³ to improve the performance of the GPCA algorithm.

5.1.1 Fisher Linear Discriminant Analysis (LDA)

Given a set of labeled sample points $\{\mathbf{x}_i\}_{i=1}^N$ drawn from multiple, say n , clusters, the Fisher linear discriminant analysis aims to

Find a linear combination $\hat{\mathbf{x}} = \mathbf{c}^T \mathbf{x}$ such that the within-cluster variance is minimized relative to the between-cluster variance.

As discriminant analysis is primarily a supervised-learning method, it requires that the membership of samples is known. However, as we will see in the next section, some of the key notions of discriminant analysis can be adopted by GPCA to improve its performance without knowing the membership.

To minimize the within-cluster variance, we would like all data samples to be as close to their respective cluster means as possible. Thus for a given cluster j with the mean $\boldsymbol{\mu}_j \in \mathbb{R}^D$, we aims to minimize the following objective:

$$\min J_{W_j} \doteq \sum_{i=1}^{N_j} \|\hat{\mathbf{x}}_i - \hat{\boldsymbol{\mu}}_j\|^2 = \sum_{i=1}^{N_j} \|\mathbf{x}_i^T \mathbf{c} - \boldsymbol{\mu}_j^T \mathbf{c}\|^2 \quad (5.1)$$

$$= \mathbf{c}^T \left(\sum_{i=1}^{N_j} (\mathbf{x}_i - \boldsymbol{\mu}_j)(\mathbf{x}_i - \boldsymbol{\mu}_j)^T \right) \mathbf{c} \doteq \mathbf{c}^T W_j \mathbf{c} \quad (5.2)$$

with $\{\mathbf{x}_i\}_{i=1}^{N_j}$ belong to the j th cluster, $j = 1, 2, \dots, n$. We will call W_j the *within-cluster variance matrix* for the j th cluster.

To best separate the n clusters from each other, we would also like the cluster means themselves to be as far apart as possible. Let $\bar{\boldsymbol{\mu}}$ be the mean of the n cluster

²Most of the techniques of discriminant analysis are for supervised learning anyway. They are not directly usable in our unsupervised setting.

³Fisher LDA coincides with conventional LDA when all the clusters are Gaussian distributions with the same covariance. Although this assumption does not strictly apply to the statistical model of subspaces of different dimensions (given in Section 3.3.2), key ideas of Fisher LDA may still be adopted by GPCA.

means $\{\boldsymbol{\mu}_j\}_{j=1}^n$. Thus, we would like to maximize the variance of means:

$$\max J_B \doteq \frac{1}{n} \sum_{j=1}^n \|\hat{\boldsymbol{\mu}}_j - \hat{\boldsymbol{\mu}}\|^2 = \frac{1}{n} \sum_{j=1}^n \|\boldsymbol{\mu}_j^T \mathbf{c} - \bar{\boldsymbol{\mu}}^T \mathbf{c}\|^2 \quad (5.3)$$

$$= \mathbf{c}^T \left(\frac{1}{n} \sum_{j=1}^n (\boldsymbol{\mu}_j - \bar{\boldsymbol{\mu}})(\boldsymbol{\mu}_j - \bar{\boldsymbol{\mu}})^T \right) \mathbf{c} \doteq \mathbf{c}^T B \mathbf{c}. \quad (5.4)$$

The matrix B defined above is called the *between-cluster variance matrix* for n clusters with cluster means $\{\boldsymbol{\mu}_j\}_{j=1}^n$.

The objective of the Fisher LDA is to find the line $\mathbf{c} \in \mathbb{R}^D$ that minimizes the projected within-cluster variance $\mathbf{c}^T W \mathbf{c}$ relative to the projected between-cluster variance $\mathbf{c}^T B \mathbf{c}$. We can accomplish this by minimizing the ratio of these two variances, which takes the form of the *Rayleigh Quotient*:

Definition 5.1 (Rayleigh Quotient). *For two square symmetric matrices $W, B \in \mathbb{R}^{D \times D}$ and a vector $\mathbf{c} \in \mathbb{R}^D$, the Rayleigh Quotient is the ratio*

$$R(\mathbf{c}) \doteq \frac{\mathbf{c}^T W \mathbf{c}}{\mathbf{c}^T B \mathbf{c}}. \quad (5.5)$$

From $\nabla R(\mathbf{c}) = 0$, one can show that the unit vector \mathbf{c} that minimizes the Rayleigh Quotient is the minimal generalized eigenvector of the matrix pair (W, B) . That is, \mathbf{c} satisfies the equation

$$W\mathbf{c} = \lambda B\mathbf{c} \quad \text{for some } \lambda \in \mathbb{R}. \quad (5.6)$$

Furthermore, if B is invertible, then \mathbf{c} is just the eigenvector of the matrix $B^{-1}W$ associated with the smallest eigenvalue λ_{\min} .

5.1.2 Fisher Discriminant Analysis for Subspaces

The basic idea of GPCA, as described in the previous chapters, is to fit the entire data set sampled from an arrangement of subspaces with a set of polynomials so that the subspaces are the common zero set of the polynomials. If the data samples are drawn from an arrangement of hyperplanes, then the polynomials are all generated by a factorable polynomial:

$$p(\mathbf{x}) = \prod_{j=1}^n (\mathbf{b}_j^T \mathbf{x}) = \mathbf{c}^T \nu_n(\mathbf{x}) = 0 \quad (5.7)$$

with n the number of (different) hyperplanes and \mathbf{b}_j the normal vector to the j th plane. In this case, it is very easy to find the coefficient vector \mathbf{c} . The kernel of the data matrix $\mathbf{V}_n(D)$ is only one-dimensional, so the smallest singular vector will readily yield the vector \mathbf{c} up to a scale.

However, if the data samples are drawn from an arrangement of linear subspaces, not all of which are hyperplanes in \mathbb{R}^D , then in general the kernel of $\mathbf{V}_n(D)$ is multi-dimensional and the vectors \mathbf{c} are in general linear combinations of the coefficients of polynomials of the form (5.7).

In the presence of noise, it is likely that $p(\mathbf{x}) \neq 0$, but we would like to find the coefficient vector \mathbf{c} that minimizes the following least-square fitting error

$$\min J_W \doteq \sum_{i=1}^N |p(\mathbf{x}_i)|^2 = \mathbf{c}^T \left(\sum_{i=1}^N \nu_n(\mathbf{x}_i) \nu_n(\mathbf{x}_i)^T \right) \mathbf{c} \quad (5.8)$$

$$= \mathbf{c}^T [\mathbf{V}_n(D)^T \mathbf{V}_n(D)] \mathbf{c} \doteq \mathbf{c}^T W \mathbf{c}, \quad (5.9)$$

where the matrix W will be called the *within-subspace scatter matrix*. The eigenvectors of W associated with its smallest eigenvalues form a basis for the coefficients \mathbf{c} of all the polynomials that fit the data set with a given error threshold. Nevertheless, the polynomial that minimizes J_W is not necessarily the best for separating the noisy data into their respective subspaces.

Let us examine the derivative of the polynomial at each of the data samples. Let us assume that the data sample \mathbf{x}_1 lies exclusively in the subspace S_1 . Then we have:

$$\nabla p(\mathbf{x}_1) = \left(\prod_{j=2}^n \mathbf{b}_j^T \mathbf{x}_1 \right) \mathbf{b}_1 = \mathbf{c}^T \nabla \nu_n(\mathbf{x}_1). \quad (5.10)$$

The direction of $\nabla p(\mathbf{x}_1)$ in (5.10) is the same as the vector \mathbf{b}_1 , and its magnitude is given by

$$\|\nabla p(\mathbf{x}_1)\|^2 = \left| \left(\prod_{j=2}^n \mathbf{b}_j^T \mathbf{x}_1 \right) \right|^2. \quad (5.11)$$

The average of the quantity $\|\nabla p(\mathbf{x}_1)\|$ over all \mathbf{x}_1 in S_1 gives a good measure of “distance” from S_1 to $\bigcup_{j=2}^n S_j$, the union of the other subspaces in the model. We leave it as an exercise for the reader to verify for $n = 2$ lines in \mathbb{R}^2 , what the value $\int_{\mathbf{x} \in S_1} \|\nabla p(\mathbf{x})\|^2 d\mathbf{x}$ is when \mathbf{x} is a zero-mean Gaussian distribution in S_1 . Thus, for the segmentation purpose, so we would like to find the coefficient vector \mathbf{c} that maximizes $\|\nabla p(\mathbf{x})\|^2$.

$$\max J_B \doteq \sum_{i=1}^N \|\nabla p(\mathbf{x}_i)\|^2 = \mathbf{c}^T \left(\sum_{i=1}^N \nabla \nu_n(\mathbf{x}_i) \nabla \nu_n(\mathbf{x}_i)^T \right) \mathbf{c} \doteq \mathbf{c}^T B \mathbf{c}. \quad (5.12)$$

We will call B the *between-subspace scatter matrix*.

The coefficient vector \mathbf{c} that simultaneously minimizes the polynomial evaluated at each of the samples while maximizing the norm of the derivative at each point can be obtained by simply minimizing the ratio of these two metrics.

Definition 5.2 (Segmentation Polynomial). *The Segmentation Polynomial $p(\mathbf{x}) = \mathbf{c}^T \nu_n(\mathbf{x})$ is specified by the coefficient vector \mathbf{c}^* such that*

$$\mathbf{c}^* = \arg \min_{\mathbf{c}} \frac{\mathbf{c}^T W \mathbf{c}}{\mathbf{c}^T B \mathbf{c}}. \quad (5.13)$$

As before, the solution \mathbf{c}^* is given as the generalized eigenvector associated with the smallest generalized eigenvalue of the matrix pair (W, B) .

This ratio is just like the Rayleigh quotient described earlier in Fisher LDA. Let us compare the within-cluster variance matrix and the within-subspace scatter matrix. The former measures the squared Euclidean distance between samples and their cluster means; the latter measures the squares of the polynomial evaluated at the samples, which can be regarded as a squared “distance” between samples and the linear subspaces they lie on. Similarly we can compare the between-cluster variance matrix and the between-subspace scatter matrix. The former measures the squared Euclidean distance between cluster means; the latter measures the squared norms of the derivative of the polynomial evaluated at the samples, which can be regarded as a squared “distance” between one subspace to all the other linear subspaces in the arrangement.

The minimization of the Rayleigh quotient only requires that W and B are real, symmetric, positive semi-definite matrices. Thus the vector c^* that minimizes this ratio will be the minimal generalized eigenvector of W and B . In our context, the within-subspace scatter matrix B will always be full rank, because otherwise all of the data samples can be fitted with polynomials of degree lower than n . As a result, the vector c^* is simply the eigenvector of $B^{-1}W$ associated with the smallest eigenvalue.

5.1.3 Simulation Results

In the Basic GPCA Algorithm 4.4, the fitting polynomials $p(\mathbf{x})$ are estimated from eigenvectors of the matrix $W = \mathbf{V}_n(D)^T \mathbf{V}_n(D)$. We can replace the fitting polynomials with the Segmentation Polynomials estimated from the eigenvectors of the matrix pair $B^{-1}W$. For convenience, we call the resulting algorithm as *Fisher GPCA*.

To verify the improvement in performance of the GPCA algorithm, we present below a few simulations with synthetic data. Figure 5.1 shows an example data set that we will use in the first two experiments to evaluate the performance of Fisher GPCA in comparison with the basic GPCA algorithm. Notice in this case that the zero set of the Segmentation Polynomial (asymptotically) approximates a union of three planes, which results in a correct segmentation of the three subspaces – the two lines are contained in two of the planes, respectively.

Comparison of the Singular-Value Spectrums

The first experiment demonstrates how the normalization of W by B^{-1} may significantly improves the singular-value spectrum of W . That is, it makes the null space of W less sensitive to the corruption of noise, which makes the estimation of polynomials that fit the data a better-conditioned problem. To see this, let us consider a set of points drawn from two lines and one plane in \mathbb{R}^3 (see Figure 5.1) – 1000 points from the plane and 200 points from each line – with 5% Gaussian noise added. As Figure 5.2 illustrates, the generalized eigenvalues of the Rayleigh Quotient provide a much sharper “knee point” than the singular values of the (embedded) data matrix $\mathbf{V}_n(D)$ (or the eigenvalues of

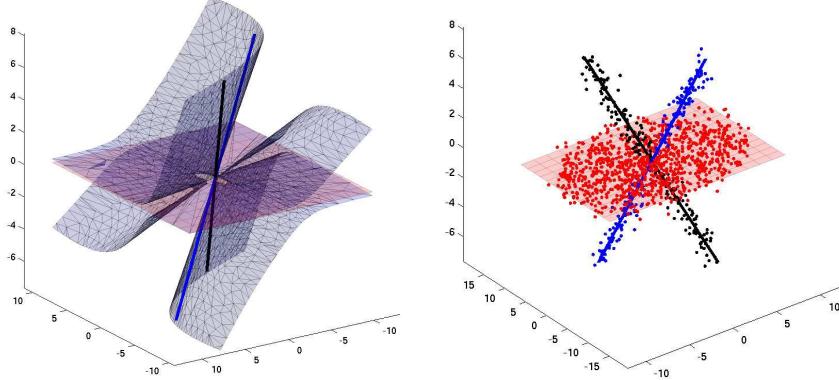


Figure 5.1. Left: The zero set of a Segmentation Polynomial for data samples drawn from two lines and a plane with 5% additive Gaussian noise. Right: The set of subspaces estimated by Fisher GPCA.

$W = V_n(D)^T V_n(D)$. With the new spectrum, one can more easily estimate the correct number of polynomials that fit the data (in this case four polynomials).

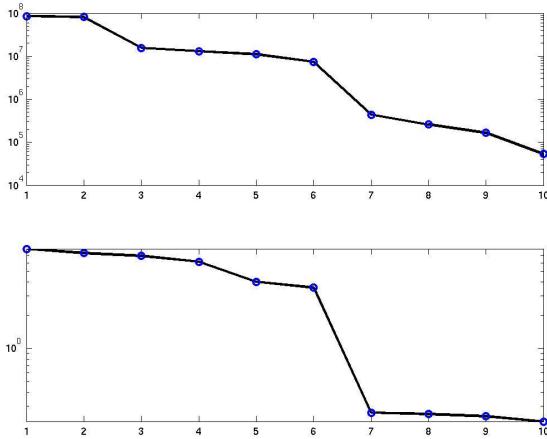


Figure 5.2. Top: Plot of the eigenvalues of the within-subspace scatter matrix W . Bottom: Plot of the eigenvalues of the matrix $B^{-1}W$ derived from the Rayleigh quotient.

Fisher GPCA versus GPCA

In this experiment, we randomly generate a number of different arrangements with one plane and two lines. The lines are chosen so as to have a random angle larger than 30° . The sample number of samples are drawn from the plane and the lines as before. We then add between 1% and 7% Gaussian noise and apply both

the basic GPCA algorithm and the Fisher GPCA to the data set, instructing them to fit the data with three linear subspaces. This test was performed 1000 times at each noise level, and for each test run the misclassification rate was computed using the known *a priori* sample labels.

Figure 5.3 shows the result of our experiment. The average misclassification rate is displayed as a function of the noise level. These results verify that while

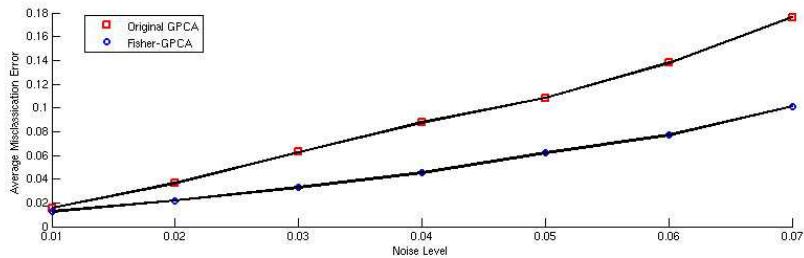


Figure 5.3. Plot of average misclassification error as a function of the noise for the GPCA algorithm and the Fisher GPCA algorithm.

the two algorithms have negligible difference in error for low noise level, as we increase the amount of noise, the difference in performance becomes much more dramatic. The ability of any set of subspaces to segment noisy data samples becomes limited as noise increases since samples near the intersections of the subspaces are more likely mis-classified. Our results show that Fisher GPCA approaches this limit.

To better understand the performance of Fisher GPCA, we can analyze the distribution of misclassification rates over the 1000 test runs for a given noise level. In Figure 5.4, the misclassification rates for 1000 test runs of the data set with 6% noise are sorted and displayed as a distribution. These distributions reveal that

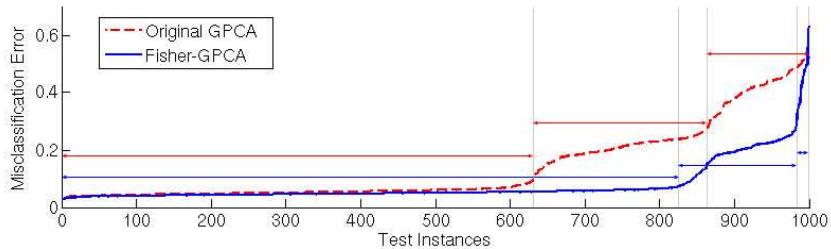


Figure 5.4. Left: Plot of misclassification error for 1000 test runs of GPCA and Fisher GPCA for 6% additive Gaussian noise.

both algorithms have performance types that can be grouped into one of three categories: (Class A), where the model and the segmentation are estimated correctly; (Class B), where the segmentation is reasonable, but the model estimation is incorrect (i.e., one or more of the subspaces has incorrect dimension); and (Class

C), where neither the model nor the segmentation is correct. As Figure 5.4 demonstrates, even in the presence of 6% noise, Fisher GPCA can produce a meaningful segmentation of the noisy data samples almost 98% of the time.

Piecewise Linear Fitting Nonlinear Data

Though Fisher GPCA is designed for subspace arrangements, it can be used to provide a piecewise linear approximation to any non-linear manifold as well. As most real world data is not necessarily strictly piecewise linear, small nonlinearity can be treated as noise (or deviation) from a (piecewise) linear model. Then the ability of Fisher GPCA in dealing with noise and in improving the separation of clusters provide strong justification for its potential success in modeling nonlinear data set.

Figure 5.5 shows an example with data samples drawn from a hemisphere in \mathbb{R}^3 with radius 5 in the presence of 2% Gaussian noise. We instruct Fisher GPCA to fit this non-linear data set with 3 or 6 subspaces, respectively. The results of the segmentation and subspaces fitted to the data set are displayed in Figure 5.5. Notice that the segmentation results for 6 groups resemble half of a dodecahedron, a regular polyhedron with 12 congruent planar faces. The results suggest that the algorithm places the 6 planes almost evenly around the sphere.

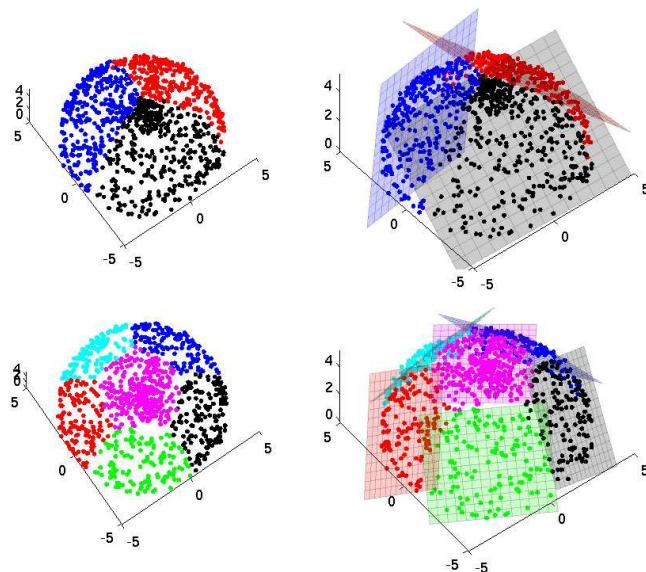


Figure 5.5. Left: Samples drawn from a hemisphere in \mathbb{R}^3 clustered into 3 and 6 groups, respectively. Right: Planes fitted to the data by the Fisher GPCA algorithm.

5.2 Voting Techniques

In the basic GPCA algorithm, the basis for each subspace is estimated from the derivatives of the fitting polynomials at a single representative point. However, if the chosen point is noisy, it may cause a large error in the estimated subspace and subsequently the segmentation. From a statistical viewpoint, more accurate estimates of the subspace can be obtained if we are able to use the derivatives at many points in the same subspace. However, a fundamental difficulty here is that we do not know which points belong to the same subspace in the first place.

There is yet another issue. In the basic GPCA algorithm, the rank of the derivatives at each point is the co-dimension of the subspace to which it belongs. In the presence of noise, it is suggested to use PCA to determine the rank. However, the estimated rank can be wrong if the point is noisy. Furthermore, it is difficult to find a common threshold for PCA that works for different subspaces.

For the rest of the section, we assume that we already know the correct number of subspaces and their individual dimensions.⁴ We show below how to improve the estimates of the subspaces from the derivatives of all the sample points.

5.2.1 Stacks of Bases and Counters

Suppose the subspace arrangement is a union of n subspaces: $Z_{\mathcal{A}} = S_1 \cup S_2 \cup \dots \cup S_n$. Let us assume that the dimensions of the subspaces are d_1, d_2, \dots, d_n and their co-dimensions are $c_i = D - d_i, i = 1, 2, \dots, n$. From the value of the Hilbert function $h_I(n)$ (see Appendix B), we know there should be $h_I(n)$ linearly independent polynomials of degree n that fit the arrangement. From a set of sample data $\mathbf{X} = \{\mathbf{x}_i\}$, we may find the set of fitting polynomials

$$P \doteq \{p_1(\mathbf{x}), p_2(\mathbf{x}), \dots, p_{h_I(n)}(\mathbf{x})\}$$

from the eigenvectors associated with the $h_I(n)$ smallest eigenvalues of the matrix $W = \mathbf{V}_n(D)^T \mathbf{V}_n(D)$.

Now suppose we pick a sample point \mathbf{x}_1 from \mathbf{X} . The derivatives of the fitting polynomials are

$$DP(\mathbf{x}_1) \doteq \{\nabla p_1(\mathbf{x}_1), \nabla p_2(\mathbf{x}_1), \dots, \nabla p_{h_I(n)}(\mathbf{x}_1)\}. \quad (5.14)$$

If there is no noise, $\text{rank}(DP(\mathbf{x}_1))$ will be exactly the co-dimension of the subspace to which \mathbf{x}_1 belongs. However, when the data are noisy, it can be very difficult to determine the co-dimension in this way. In principle, \mathbf{x}_1 can be in any of subspaces. Without loss of generality, we assume that c_1, c_2, \dots, c_n have m distinct values c'_1, c'_2, \dots, c'_m . As we do not know the exact dimension yet, we can compute a set of basis candidates

$$B_i(\mathbf{x}_1) \in \mathbb{R}^{D \times c'_i}, \quad i = 1, 2, \dots, m, \quad (5.15)$$

⁴We will study in the next section the case in which the number of subspaces and their dimensions are not known.

as $B_i(\mathbf{x}_1)$ collects the first c'_1, c'_2, \dots, c'_m principal components of $DP(\mathbf{x}_1)$. Thus, $B_i(\mathbf{x}_1)$ is a $D \times c'_i$ orthogonal matrix. The rationale here is, as we cannot yet decide the correct co-dimension at \mathbf{x}_1 , we keep all the possibilities open.

We also create a stack of bases of dimension c'_i :

$$U_i = \{U_i(1), U_i(2), \dots, U_i(J)\}, \quad (5.16)$$

where each $U_i(j)$ is a $D \times c'_i$ orthogonal matrix for all $j = 1, 2, \dots, J$. Correspondingly, we create another stack of numbers:

$$u_i = \{u_i(1), u_i(2), \dots, u_i(J)\}, \quad (5.17)$$

where each $u_i(j)$ is an integer that counts how many sample points $\mathbf{x}_k \in X$ with $B_i(\mathbf{x}_k) = U_i(j)$.

5.2.2 A Voting Scheme for Subspaces

With the above definitions, we now can outline a voting scheme that will select a set of n bases of the subspaces that in a sense achieve the highest consensus among all the sample points. For every sample point $\mathbf{x}_k \in X$,

1. we compute a set of basis candidates $B_i(\mathbf{x}_k), i = 1, \dots, m$ as defined in equation (5.15);
2. for each $B_i(\mathbf{x}_k)$, we compare it with each of the bases in the stack U_i :
 - (a) if $B_i(\mathbf{x}_k) = U_i(j)$ for some j , then increase the value of $u_i(j)$ by one;
 - (b) if $B_i(\mathbf{x}_k)$ is different from any of the bases in U_i , then add $U_i(J + 1) = B_i(\mathbf{x}_k)$ as a new basis to the stack U_i , and also add a new counter $u_i(J + 1)$ to the stack u_i with the initial value $u_i(J + 1) = 1$.

In the end, the bases of the n subspaces are chosen to be the n bases in the stacks $\{U_i\}_{i=1}^m$ that have the highest votes according to the corresponding counters in the stacks $\{u_i\}_{i=1}^m$. Once the subspaces of the highest consensus are chosen as above, each data point is assigned to the closest subspace.

In the above scheme, if the data are noisy, in order to compare $B_i(\mathbf{x}_k)$ with bases in U_i , we need to set an error tolerance. This tolerance, denoted as τ , can be a small subspace angle chosen by the user. Thus, if the subspace angle between $B_i(\mathbf{x}_k)$ and $U_i(j)$ is less than τ , we increase the value of the counter $u_i(j)$ by one and set the new value of $U_i(j)$ to be a weighted sum:

$$U_i(j) \leftarrow \frac{1}{u_i(j) + 1} (u_i(j)U_i(j) + B_i(\mathbf{x}_k)). \quad (5.18)$$

Notice that the weighted sum may no longer be an orthogonal matrix. If so, apply the Gram-Schmidt process to make $U_i(j)$ an orthogonal matrix again. We summarize the above process as Algorithm 5.1.

There are a few important features about the above voting scheme, in comparison with and different from other well-known statistical methods:

Algorithm 5.1 (Generalized Principal Component Analysis with Voting).

Given a set of samples $\{\mathbf{x}_k\}_{k=1}^N$ in R^D and a parameter for angle tolerance τ , fit n linear subspaces with co-dimensions c_1, c_2, \dots, c_n :

- 1: Let m be the number of distinct co-dimensions. Allocate u_1, u_2, \dots, u_m be m stacks of counters and U_1, U_2, \dots, U_m be m stacks of basis candidates.
- 2: Construct $V_n(D) = [\nu_n(\mathbf{x}_1), \dots, \nu_n(\mathbf{x}_N)]$.
- 3: Estimate the set of fitting polynomials $P(\mathbf{x})$, and compute $DP(\mathbf{x}_k)$ for all k .
- 4: **for all** sample \mathbf{x}_k **do**
- 5: **for all** $1 \leq i \leq m$ **do**
- 6: Assume \mathbf{x}_k is drawn from a subspace of co-dimension c_i . Find the first c_i principal vectors $B_i(\mathbf{x}_k) \in R^{D \times c_i}$ of $DP(\mathbf{x}_k)$.
- 7: Increase $u_i(j)$ by one if the subspace angle between $B_i(\mathbf{x}_k)$ and $U_i(j)$ is less than τ and reset $U_i(j)$ to be the weighted sum in (5.18). If the angle is always larger than τ for all j , create a new basis candidate in U_i and a new counter in u_i with initial value one.
- 8: **end for**
- 9: **end for**
- 10: Choose the n highest vote(s) in $\{u_i\}$ with their corresponding base(s) in $\{U_i\}$.
- 11: Assign the samples to their closest subspaces.
- 12: Re-estimate the basis of each subspace from the derivatives of all the points that belong to the same subspace.

1. *K-subspaces*: The K-subspaces algorithm keeps (iteratively updating) one basis for each subspace; while the voting scheme essentially keeps multiple basis candidates for the subspaces through the process. However, the above algorithm does not have the same problem with local minima as K-subspaces does. The weighted sum behaves as the estimation (averaging) step of K-subspaces. As we have seen from experiments, the voting algorithm has a performance very close to that of K-subspaces (or EM) initialized by the GPCA voting algorithm.
2. *RANSAC*: The RANSAC method computes multiple candidate models from multiple (down-sampled) subsets of the data and then chooses the one which achieves the most dominant consensus; while in GPCA, none of the n subspaces is likely to achieve a dominant consensus. This largely prevents us to directly apply RANSAC to GPCA (for more discussions on RANSAC, see Section 5.4). However, the voting scheme allows us to simultaneously identify the n subspaces that get the n highest votes relative to all other possible subspaces.

5.2.3 Simulation Results

We here give a preliminary comparison of the various algorithms for segmenting subspaces that we have studied so far. They include: The EM algorithm, the K-subspaces algorithm, the basic GPCA algorithm, and the GPCA algorithm with voting (as well as some combination of them).

We randomly generated some subspace arrangements of some pre-chosen dimensions. For instance, $(2, 2, 1)$ indicates an arrangement of three subspaces of dimensions 2, 2, 1, respectively. We then randomly draw a number of samples from them. The samples are corrupted with Gaussian noises. Here we choose the level of noise to be 4%.⁵ The error is measured in terms of the percentage of sample points that are wrongfully grouped.⁶ All cases are averaged over 200 trials. The performance of all the algorithms are compared in Table 5.1.

Table 5.1. The percentage of sample points mis-grouped by different algorithms. (Note: The number of subspaces and their dimensions are given to all algorithms. The EM and K-Subspaces algorithms are randomly initialized. But “GPCA-Voting+K-Subspaces” means the K-Subspaces method initialized with the GPCA-Voting algorithm.)

Methods	$(2, 2, 1) \in \mathbb{R}^3$	$(2, 2, 2) \in \mathbb{R}^3$	$(4, 2, 2, 1) \in \mathbb{R}^5$	$(4, 4, 4, 4) \in \mathbb{R}^5$
EM	29%	11%	53%	20%
K-Subspaces	27%	12%	57%	25%
GPCA-Basic	10.3%	10.6%	39.8%	25.3%
GPCA-Voting	6.4%	9.2%	5.7%	17%
GPCA-Voting + K-Subspaces	5.4%	8.6%	5.7%	11 %

5.3 Model-Selection Methods

Most subspace-segmentation algorithms (e.g., EM, K-subspaces, GPCA-voting) assume that the number of subspaces and their dimensions are known or given. If they are *not* given, the problem of fitting multiple subspaces to a data set becomes much more elusive. For instance, sample points drawn from two lines and one plane in \mathbb{R}^3 can also be fit by two planes, one of which is spanned by the two lines. In Chapter 4, we have suggested that in this case one can apply the basic GPCA algorithm in a *recursive* fashion to identify all the subspaces (and their dimensions).

However, when there is significant noise in the given data, the purely algebraic GPCA algorithm may fail to return a meaningful solution. In fact, up till now, we have been purposely avoiding a fundamental difficulty in our problem: it is inherently *ambiguous* in fitting multiple subspaces for any given data set, especially if the number of subspaces and their dimensions are not given *a priori*. When the data is noisy or nonlinear, any multi-subspace model unlikely will fit the data perfectly except for the pathological cases: 1. All points are viewed as in one

⁵The percentage is the variance of the Gaussian relative to the diameter of the data.

⁶Notice that even with perfect knowledge of the subspaces, with noise being added to a sample, the closest subspace to the sample may change as a consequence.

D -dimensional subspace – the ambient space; 2. Every point is viewed as in an individual one-dimensional subspace. Furthermore, from the example of a hemisphere (Figure 5.5), the more number of planes we use, the higher accuracy may we achieve in fitting the data. Thus, a fundamental question we like to address in this section is:

Among a class of subspace arrangements, what is the “optimal” model that fits a given data set?

From a practical viewpoint, we also need to know under what conditions the optimal model exists and is unique, and more importantly, how to compute it efficiently.

In Appendix C, we have seen that in general, any model selection criterion aims to strike a balance between the complexity of the resulting model and the fidelity of the model to the given data. However, its exact form often depends on the class of models of interest as well as how much information is given about the model in advance. If we were to apply any of the model-selection criteria (or their concepts) to subspace arrangements, at least two issues need to be addressed:

1. We need to know how to measure the model complexity of arrangements of subspaces (possibly of different dimensions).
2. As the choice of a subspace arrangement involves both continuous parameters (the subspace bases) and discrete parameters (the number of subspaces and their dimensions), we need to know how to properly balance the model complexity and the modeling error for subspace arrangements.

While model selection for subspace arrangements in its full generality is still an open problem at this point, in the next two subsections, we introduce a few specific approaches that address this problem from slightly different aspects. We hope the basic concepts introduced below may help the reader to better appreciate the subtlety and difficulty of the problem.

5.3.1 Minimum Effective Dimension

Definition 5.3 (Effective Dimension). *Given an arrangement of n subspaces $Z_{\mathcal{A}} \doteq \cup_{j=1}^n S_j$ in \mathbb{R}^D of dimension $d_j < D$, and N_j sample points \mathbf{X}_j drawn from each subspace S_j , the effective dimension of the entire set of $N = \sum_{j=1}^n N_j$ sample points, $\mathbf{X} = \cup_{j=1}^n \mathbf{X}_j$, is defined to be:*

$$\text{ED}(\mathbf{X}, Z_{\mathcal{A}}) \doteq \frac{1}{N} \left(\sum_{j=1}^n d_j(D - d_j) + \sum_{j=1}^n N_j d_j \right). \quad (5.19)$$

We contend that $\text{ED}(\mathbf{X}, Z_{\mathcal{A}})$ is the “average” number of (unquantized) real numbers that one needs to assign to \mathbf{X} per sample point in order to specify the configurations of the n subspaces and the relative locations of the sample points in

the subspaces.⁷ In the first term of equation (5.19), $d_j(D - d_j)$ is the total number of real numbers (known as the Grassmannian coordinates⁸) needed to specify a d_j -dimensional subspace S_j in \mathbb{R}^D ; in the second term of (5.19), $N_j d_j$ is the total number of real numbers needed to specify the d_j coordinates of the N_j sample points in the subspace S_j . In general, if there are more than one subspace in $Z_{\mathcal{A}}$, $\text{ED}(\mathbf{X}, Z_{\mathcal{A}})$ can be a rational number, instead of an integer for the conventional dimension.

Notice that in the above definition, the effective dimension of \mathbf{X} depends on the subspace arrangement $Z_{\mathcal{A}}$. This is because in general, there could be many subspace structures that can fit \mathbf{X} . For example, we could interpret the whole data set as lying in one D -dimensional subspace and we would obtain an effective dimension D . On the other hand, we could interpret every point in \mathbf{X} as lying in a one-dimensional subspace spanned by itself. Then there will be N such one-dimensional subspaces in total and the effective dimension, according to the above formula, will also be D . In general, such interpretations are obviously over-fitting. Therefore, we define the *effective dimension* of a given sample set \mathbf{X} to be the minimum one among all possible multiple-subspace models that can fit the data set:⁹

$$\text{MED}(\mathbf{X}) \doteq \min_{Z_{\mathcal{A}}: \mathbf{X} \subset Z_{\mathcal{A}}} \text{ED}(\mathbf{X}, Z_{\mathcal{A}}). \quad (5.20)$$

Example 5.4 (Effective Dimension of One Plane and Two Lines). Figure 1.2 shows data points drawn from one plane and two lines in \mathbb{R}^3 . Obviously, the points in the two lines can also be viewed as lying in the plane that is spanned by the two lines. However, that interpretation would result in an increase of the effective dimension since one would need two coordinates to specify a point in a plane, as opposed to one in a line. For instance, suppose there are fifteen points in each line; and thirty points in the plane. When we use two planes to represent the data, the effective dimension is: $\frac{1}{60}(2 \times 2 \times 3 - 2 \times 2^2 + 60 \times 2) = 2.07$; when we use one plane and two lines, the effective dimension is reduced to: $\frac{1}{60}(2 \times 2 \times 3 - 2^2 - 2 \times 1 + 30 \times 1 + 30 \times 2) = 1.6$. In general, if the number of points N is arbitrarily large (say approaching to infinity), depending on the distributions of points on the lines or the plane, the effective dimension may approach arbitrarily close to either 1 or 2, the true dimensions of the subspaces. ■

As suggested by this intuitive example, the multiple-subspace model that leads to the minimum effective dimension normally corresponds to a “natural” and hence “efficient” representation of the data in the sense that it achieves the

⁷We here choose real numbers as the basic “units” for measuring complexity in a similar fashion to binary numbers, “bits,” traditionally used in algorithmic complexity or coding theory.

⁸Notice that to represent a d -dimensional subspace in a D -dimensional space, we only need to specify a basis of d linearly independent vectors for the subspace. We may stack these vectors as rows of a $d \times D$ matrix. Any nonsingular linear transformation of these vectors span the same subspace. Thus, without loss of generality, we may assume that the matrix is of the normal form $[I_{d \times d}, G]$ where G is a $d \times (D - d)$ matrix consisting of the so-called Grassmannian coordinates.

⁹The space of multiple-subspace models is topologically compact and closed, hence the minimum effective dimension is always achievable and hence well-defined.

best compression (or dimension reduction) among all possible multiple-subspace models.

In practice, real data are corrupted with noise, hence we do not expect that the optimal model fits the data perfectly. The conventional wisdom is to strike a good balance between the complexity of the chosen model and the data fidelity (to the model). This is the same rationale that has been adopted in all the model-selection criteria. For instance, we may adopt the geometric-AIC (GAIC) criterion¹⁰ and use the following objective to select the optimal multiple-subspace model:

$$Z_{\mathcal{A}}^* = \arg \min_{Z_{\mathcal{A}}: \hat{\mathbf{X}} \subset Z_{\mathcal{A}}} (\|\mathbf{X} - \hat{\mathbf{X}}\|^2 + 2\epsilon^2 \text{ED}(\mathbf{X}, Z_{\mathcal{A}})), \quad (5.21)$$

where ϵ^2 is the noise variance of the data. However, there is one practical problem in achieving this objective: it is computationally costly and the variance ϵ^2 might not be known *a priori*. We need to conduct a global search in the configuration space of all subspace arrangements, which has very complicated topological and geometric structures. This often results in a very difficult optimization problem.

To alleviate some of the difficulty, in practice, we may instead minimize the effective dimension subject to a maximum allowable error residue. That is, among all the multiple-subspace models that fit the data within a given error bound, we choose the one with the smallest effective dimension. To this end, we define the minimum effective dimension *subject to an error tolerance* τ as:

$$\text{MED}(\mathbf{X}, \tau) \doteq \min_{Z_{\mathcal{A}}: \|\mathbf{X} - \hat{\mathbf{X}}\|_{\infty} \leq \tau} \text{ED}(\hat{\mathbf{X}}, Z_{\mathcal{A}}), \quad (5.22)$$

where $\hat{\mathbf{X}}$ is the projection of \mathbf{X} onto the subspaces in $Z_{\mathcal{A}}$ and the error norm $\|\cdot\|_{\infty}$ indicates the maximum norm: $\|\mathbf{X} - \hat{\mathbf{X}}\|_{\infty} = \max_{\mathbf{x} \in \mathbf{X}} \|\mathbf{x} - \hat{\mathbf{x}}\|$. Based on the above definition, the effective dimension of a data set is then a notion that depends on the error tolerance. In the extreme, if the error tolerance is arbitrarily large, the “optimal” subspace-model for any data set can simply be the (zero-dimensional) origin; if the error tolerance is zero instead, for data with random noise, most sample points need to be treated as one-dimensional subspaces in \mathbb{R}^D and that brings the effective dimension up close to D .

In many applications, the notion of maximum allowable error tolerance is particularly relevant. For instance, in image representation and compression, the task is often to find a linear or hybrid linear model to fit the imagery data subject to a given peak signal to noise ratio (PSNR).¹¹ The resulting effective dimension directly corresponds to the number of coefficients needed to store the resulting representation. The smaller the effective dimension is, the more compact or compressed is the final representation. In Chapter 6, we will see exactly how the minimum effective dimension principle is applied to image representation. The

¹⁰We here adopt the GAIC criterion only to convey the basic ideas. In practice, depending on the problem and application, it is possible that other model selection criteria may give better performance.

¹¹In this context, the noise is the difference between the original image and the approximate image (the signal).

same principle can be applied to any situation in which one tries to fit a piecewise linear model to a data set whose structure is nonlinear or unknown.

Unlike the geometric AIC (5.21), the MED objective is relatively easy to achieve. For instance, the recursive GPCA algorithm in Section 4.3.3 can be modified to minimize the effective dimension subject to an error tolerance. For instance, we allow the recursion to proceed only if the effective dimension would decrease while the resulting subspaces still fit the data with the given error bound.

Figure 5.6 demonstrates such a recursive GPCA algorithm on segmenting synthetic data drawn from two lines (100 points each) and one plane (400 points) in \mathbb{R}^3 corrupted with 5% uniform noise (Figure 5.6 top-left). Given a reasonable error tolerance, the algorithm stops after two levels of recursion (Figure 5.6 top-right). Note that the pink line (top-right) or group 4 (bottom-left) is a “ghost” line at the virtual intersection of the original plane and the plane spanned by the two lines.¹² Figure 5.6 bottom-right is the plot of MED versus different error tolerances for the same data set. As we see, the effective dimension decreases monotonically with the increase of error tolerance.

5.3.2 Hilbert Function for Model Selection

Besides the effective dimension, there is another way to measure the “complexity” of a subspace arrangement. For a single subspace, its dimension is a natural measure of the complexity of the model. Suppose we have a collection of noisy points $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^N$ drawn from some subspace in \mathbb{R}^D . Let us collect the data into a $D \times N$ matrix \mathbf{V} . One popular method to determine the dimension d of the subspace is by minimizing the so-called geometric-AIC (GAIC) criterion [?]:

$$\text{GMDL}(d) = \sum_{l=d+1}^D \sigma_l^2 + 2\epsilon^2 d(N + D - d), \quad (5.23)$$

where in our context σ_l is the l th singular value of the data matrix \mathbf{V} and ϵ^2 is the noise variance of the column vectors of the data matrix.¹³ Notice that in the GAIC expression, the term $d(N + D - d) = d(D - d) + dN$ is exact the effective dimension of the data set fit with a d -dimensional subspace in \mathbb{R}^D .

Then given an arrangement of n subspaces in \mathbb{R}^D , we may embed them into a high-dimensional space via a Veronese map $\nu_{n'}$ with $n' \geq n$. Then the image of the arrangement is a single subspace in $\mathbb{R}^{M_{n'}(D)}$ whose co-dimension is given by the value of the Hilbert function $h_I(n')$. In other words, the dimension of the subspace is $d = M_{n'}(D) - h_I(n')$. We may view the dimension of this embedded subspace as a measure of complexity for the subspace arrangement. If so, a

¹²This is exactly what we would have expected since robust GPCA first segments the data into two planes. If needed, the points on the ghost line can be merged with the plane by some simple post-processing.

¹³In practice, ϵ^2 is not necessarily known and can be hard to estimate.

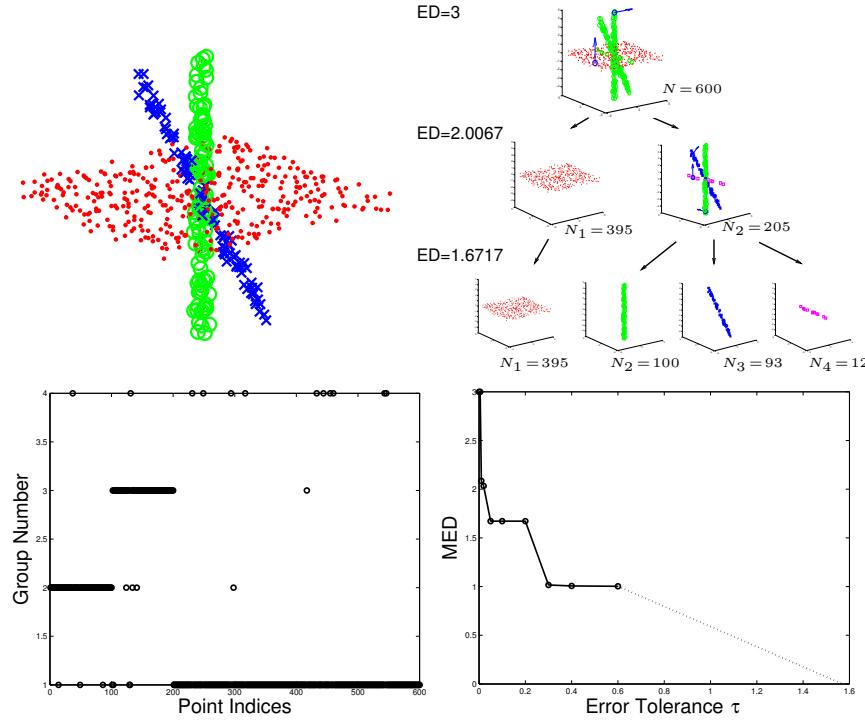


Figure 5.6. Simulation results. Top-left: sample points drawn from two lines and a plane in \mathbb{R}^3 with 5% uniform noise; Top-right: the process of recursive segmentation by the recursive GPCA algorithm with the error tolerance $\tau = 0.05$; Bottom-left: group assignment for the points; Bottom-right: plot of MED versus error tolerance.

straightforward generalization of the GAIC criterion to subspace arrangements is

$$GMDL = \sum_{l=h_I(n')+1}^{M_{n'}(D)} \sigma_l^2 + \kappa(M_{n'}(D) - h_I(n'))(N + h_I(n')), \quad (5.24)$$

where $\kappa \in \mathbb{R}$ is a weighting parameter that plays a similar role as $2\epsilon^2$ in the original GAIC.

Estimating the Number of Hyperplanes

To see why in the above GMDL criterion we do not consider other values of subspace dimension in $\mathbb{R}^{M_{n'}(D)}$ than the ones given by the Hilbert function, let us consider the example of determining a number of hyperplanes that fit a data in \mathbb{R}^5 .¹⁴

¹⁴The importance of this example is that one can show the problem of segmenting multiple affine motions in computer vision can be converted to the problem of segmenting multiple hyperplanes in \mathbb{R}^5 [?, ?].

Suppose our data are drawn from an arrangement of n hyperplanes but we only know an upper bound for the number, say $n' \geq n$. We may embed the data by a Veronese map of degree n' . Table B.2 gives the possible ranks of the data matrix $\mathbf{V}_4(5)$ for $n \leq n' = 4$. Notice that $\text{rank}(\mathbf{V}_4(5)) = M_4(5) - h_I(4) = 70 - h_I(4)$.

c_1	c_2	c_3	c_4	$h_I(4)$	$\text{rank}(\mathbf{V}_4(5))$
1	1	1	1	1	69
1	1	1	5	5	65
1	1	5	5	15	55
1	5	5	5	35	35

Table 5.2. Values of the Hilbert function of arrangements of up to four hyperplanes in \mathbb{R}^5 .

Figure B.2 shows a super-imposed plot of the singular values of $\mathbf{V}_4(5)$ of data from the four arrangements without noise.

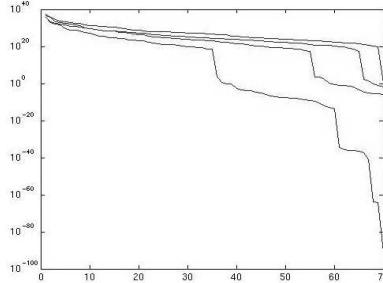


Figure 5.7. A super-imposed semi-log plot of the singular values when $n = 1, 2, 3, 4$. The ranks drop at position 35, 55, 65, 69, which match the theoretical values of the Hilbert function.

The above table suggests that if we embed the data by the Veronese map of the same degree $4 \geq N$. By doing so, we over-fit the hyperplane arrangement. But the image of the arrangement in the high-dimensional space is a subspace whose dimension can only be one of the four possible values 35, 55, 65, 69, instead of any value between 1 to 69. Therefore, if we only consider those values for the subspace dimension d in the GAIC criterion, then the optimal number of hyperplanes n^* can be determined as

$$n^* = \arg \min_{n \leq 4} \left\{ \sum_{l=h_I(4)+1}^{M_4(5)} \sigma_l^2 + \kappa (M_4(5) - h_I(4))(N + h_I(4)) \right\}, \quad (5.25)$$

where $h_I(4)$ is the value of the Hilbert function for n hyperplanes and $n' - n = 4 - n$ trivial subspaces (the origin), N is the number of samples, and κ is a small positive weighting constant. One can verify with simulations that the new GAIC

criterion has better performance than other traditional model-selection criteria that do not harness the information of the Hilbert function.

Estimating the Dimensions of Subspaces

In the above, we have shown through an example how one can incorporate model selection criteria with Hilbert functions to help determine the rank of the data matrix and subsequently determine the number of hyperplanes. However, if the subspaces in the arrangement have different dimensions (instead of all being hyperplanes), we need to determine the dimension of each subspace too. Here we assume we only know the dimensions of the subspaces up to some possibilities but do not know the exact dimensions.¹⁵ Below, we show how one can harness the information in the Hilbert function to help determine the correct dimensions

To better illustrate the basic ideas, we again use a simple example. Suppose that we know our data are drawn from an arrangement of $n = 4$ subspaces in \mathbb{R}^5 . The dimensions of the subspaces are either 4 or 3 (i.e., , of co-dimension 1 or 2.)

Table 5.3 shows the values of the Hilbert functions of the 5 possible classes of subspace arrangements for the 4 subspaces.

c_1	c_2	c_3	c_4	$h_I(4)$	rank($V_4(5)$)
1	1	1	1	1	69
1	1	1	2	2	68
1	1	2	2	4	66
1	2	2	2	8	62
2	2	2	2	16	54

Table 5.3. Values of Hilbert function for arrangements of four subspaces in \mathbb{R}^5 with dimension 3 or 4 (codimension 1 or 2, respectively).

To determine the best subspace arrangement that fits and segments the noisy data, we decompose the overall estimation process into three steps:

1. **Segmentation via Over-Fitting Hyperplanes.** We first obtain a segmentation of the data by using a single polynomial that fit the best the (noisy) data (e.g., the segmentation polynomial from Fisher GPCA)
2. **Combinatorial Dimension Selection.** Assume we know the class (i.e., row of Table 5.3) of the subspace arrangements. There remains a combinatorial problem of determining which segment needs to be fit with a subspace of which dimension. For instance, if the class is the second row in Table 5.3, we have to determine which segment has codimension 2, and there are obviously four possible configurations: (2, 1, 1, 1), (1, 2, 1, 1), (1, 1, 2, 1), and (1, 1, 1, 2).

¹⁵If the dimensions were known, one could simply apply the voting GPCA method introduced earlier.

For each configuration, we can apply PCA to each segment and identify a basis for each subspace. Then a K-subspaces method can be applied to further improve the estimates of the bases. Finally, we project the original data points \mathbf{x} to their respective subspaces as $\hat{\mathbf{x}}$. Select the configuration that gives the smallest residual $\sum \|\mathbf{x} - \hat{\mathbf{x}}\|^2$ as the optimal solution for the given model class.

3. **Optimal Model Class Selection.** It remains to determine which model class gives the (globally) optimal subspace arrangement for the data. The value of the Hilbert function h is used to measure the complexity of the subspace arrangements. We adopt the GAIC criterion that we proposed earlier, but replace modeling error $\sum \sigma^2$ with the residual of the optimal configuration (found in Step 2) for each class:

$$i^* = \arg \min_i \left\{ \sum \|\mathbf{x} - \hat{\mathbf{x}}\|^2 + \kappa(M_4(5) - h_I^i(4))(N + h_I^i(4)) \right\}, \quad (5.26)$$

where $h_I^i(4) = 69, 68, 66, 62, 54$ for $i = 1, 2, 3, 4, 5$, respectively, n is the number of samples, and κ is a small positive weighting parameter.

5.4 Robust Statistical Techniques

Given a set of sample points $\{\mathbf{x}_i\}_{i=1}^N$ drawn from an arrangement of subspaces $\{S_i\}_{i=1}^N$ in \mathbb{R}^D , GPCA seeks to simultaneously infer the subspaces and segment the data points to their closest subspaces. The key idea is to identify the set of polynomials $P(\mathbf{x}) = \{p_n(\mathbf{x})\}$ of degree n that vanish on (or fit) all the sample data. The way to estimate the polynomials is to use the Veronese embedding ν_n of degree n as a kernel map and embed the data points \mathbf{x} into a higher-dimensional space. The coefficients of each vanishing polynomial $p_n(\mathbf{x})$ is then in the left null space of the following embedded data matrix

$$\mathbf{V}_n(D) \doteq [\nu_n(\mathbf{x}_1), \nu_n(\mathbf{x}_2), \dots, \nu_n(\mathbf{x}_N)] \in \mathbb{R}^{M_n(D) \times N}, \quad (5.27)$$

where $M_n(D)$ is the number of monomials of degree n in D variables, and $\dim(\text{Null}(\mathbf{V}_n(D)))$ is given by the Hilbert function of the subspace arrangement (see Appendix B). Once the vanishing polynomials are found, the derivatives of the polynomials $DP(\mathbf{x}_i) = \{\nabla p(\mathbf{x}_i)\}$ at each sample point \mathbf{x}_i give the normal vectors to the subspace to which it belongs, which further allows us to segment points that belong to the same subspace. In the presence of noise, the voting scheme introduced earlier can be used to significantly improve the stability of the estimated subspace.

However, all these algorithms would not give a good estimation of the subspaces if the sample data are corrupted by just a small amount of outliers. Figure 5.8 shows the performance of the GPCA algorithm with various percentages of outliers. As we see, with only 6% outliers in the sample data, the segmentation error can be as high as 50% for four subspaces of dimensions 4, 2, 2, 1 in \mathbb{R}^5 ,

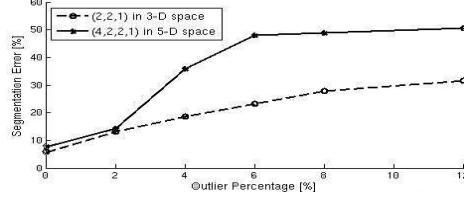


Figure 5.8. Performance of GPCA with 6% Gaussian noise and different percents of outliers.

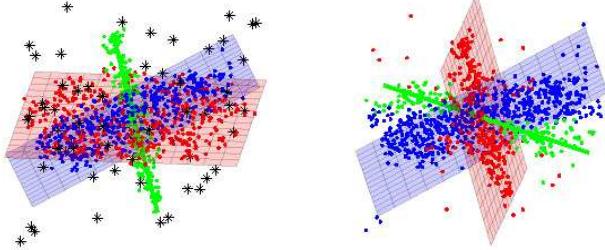


Figure 5.9. A GPCA estimation result with 6% noise and 6% outliers (black stars) in 3-D space. Left: apriori structure. Right: segmentation result.

respectively. More seriously, the estimated subspaces can be very far from the ground truth, as illustrated in Figure 5.9.

The outliers affects the segmentation result mainly by influencing $\text{Null}(\mathbf{V}_n)$, and results in erroneous estimates of the coefficients of the fitting polynomials in P . Therefore, to eliminate the effect of outliers, we are essentially seeking a *robust* PCA method to estimate $\text{Null}(\mathbf{V}_n)$ such that it is insensitive to the outliers, or to reject the outliers before estimating $\text{Null}(\mathbf{V}_n)$. Such a robust modification applies to all GPCA algorithms mentioned before.

5.4.1 The Sample Influence Function

The most direct way to detect outliers is the “leave-one-out” method from the influence function theory in statistics, which measures the change of the estimated parameter by omitting one sample at a time from the sample set. Notice that for the GPCA problem, we are not concerned with the changes of individual base vectors in $\text{Null}(\mathbf{V}_n)$, but rather the null space as a whole. The base vectors may freely rotate in the same subspace, but their corresponding polynomials will fit the subspaces equally well. Therefore, we define the influence function for the m th sample to be:

$$I_-(\mathbf{x}_m; \mathbf{V}_n) = \langle \text{Null}(\mathbf{V}_n), \text{Null}(\mathbf{V}_n(-m)) \rangle, \quad (5.28)$$

where $\langle \cdot, \cdot \rangle$ is the subspace angle between two subspaces, and $\text{Null}(\mathbf{V}_n(-m))$ is the null space of \mathbf{V}_n with the m th column omitted. All samples then can be sorted by their influence values, and the ones with the highest values will be rejected as

“outliers” and will not be used for the estimation of the null space. Equation (5.28) is also called the *sample influence function* in the literature of robust statistics.

5.4.2 First Order Approximation of the Sample Influence

Equation (5.28) is a precise expression in describing the influence of samples on the estimation of the vanishing polynomials $P(\mathbf{x})$. However, the complexity of the resulting algorithm is rather high. Suppose we have N samples, then we need conduct PCA $N + 1$ times in order to evaluate the influence values for the N samples. In light of this setback, some first order approximations of the influence values were developed at roughly the same period as the sample influence function was proposed [?, ?], when the computational resource was scarcer than it is today. In robust statistics, formulae that approximate an influence function are referred to as *theoretical influence functions*. We here derive one for our problem.

Let $\{\mathbf{v}_1, \dots, \mathbf{v}_c\}$ be a basis for $\text{Null}(\mathbf{V}_n)$, which is also the last c eigenvectors of the covariance matrix $\mathbf{V}_n \mathbf{V}_n^T$ and c is the value of the Hilbert function of the subspace arrangement of interest. Denote the embedded data vector as $\mathbf{u}_i = \nu_n(\mathbf{x}_i)$. The vector $\mathbf{u} = \nu_n(\mathbf{x})$ can be treated as a random vector with a cumulative distribution function (c.d.f.) F . Therefore, its mean and covariance matrix can be represented as:

$$\bar{\mathbf{u}}(F) = \int \mathbf{u} dF(\mathbf{u}), \quad \Sigma(F) = \int (\mathbf{u} - \bar{\mathbf{u}}(F))(\mathbf{u} - \bar{\mathbf{u}}(F))^T dF(\mathbf{u}). \quad (5.29)$$

Now F can be perturbed by a change of the weighting $\epsilon \in [0, 1]$ of the m th sample:

$$\tilde{F}_m(\epsilon) = (1 - \epsilon)F + \epsilon\delta_m, \quad (5.30)$$

where δ_m is the c.d.f. of a random variable that takes the value \mathbf{u}_m with probability one.

If we stack $\{\mathbf{v}_1, \dots, \mathbf{v}_c\}$ into a single vector $T \in \mathbb{R}^{c \cdot M_n(D)}$, T is also a random vector of F . When F becomes $\tilde{F}(\epsilon)$, let $\tilde{T}(\epsilon)$ be the value of T and $\mathbf{v}_i(m)$ be the value of the i th eigenvector \mathbf{v}_i after the change. Then we can define an influence function $I(\mathbf{u}_m; T)$ of the m th sample as the first order approximation of the leave-one-out influence value $\tilde{T}_m(\epsilon) - T$, that is,

$$I(\mathbf{u}_m; T) = \lim_{\epsilon \rightarrow 0} \frac{\tilde{T}_m(\epsilon) - T}{\epsilon} = \lim_{\epsilon \rightarrow 0} \begin{bmatrix} \frac{\mathbf{v}_1(m) - \mathbf{v}_1}{\epsilon} \\ \dots \\ \frac{\mathbf{v}_c(m) - \mathbf{v}_c}{\epsilon} \end{bmatrix} \in \mathbb{R}^{c \cdot M_n(D)}. \quad (5.31)$$

Essentially, $I(\mathbf{u}_m; T)$ is the first order approximation of $\tilde{T}_m(\epsilon) - T = I(\mathbf{u}_m; T) + \text{h.o.t.}(\epsilon)$.

As derived in [?], the influence function $I(\mathbf{u}_m; \mathbf{v}_i) \doteq \lim_{\epsilon \rightarrow 0} \frac{\mathbf{v}_i(m) - \mathbf{v}_i}{\epsilon}$ in equation (5.31) has the form

$$I(\mathbf{u}_m; \mathbf{v}_i) = -z_i \sum_{h \neq i}^c z_h \mathbf{v}_h (\lambda_h - \lambda_i)^{-1} \in \mathbb{R}^{M_n(D)}, \quad (5.32)$$

where z_h is the h th principal component of the sample \mathbf{u}_m , i.e., the coordinate value with respect to the h th eigenvector \mathbf{v}_h of the covariance matrix $\Sigma(F)$, and λ_h is the h th eigenvalue of $\Sigma(F)$. A further discussion of this solution can be found in [?].

The \mathbf{v} 's and λ 's in the above equation are the ones with respect to $\Sigma(F)$ of the whole data set, therefore only one PCA operation is required to compute $I(\mathbf{u}_m; T)$ for all samples, which dramatically reduces the algorithm complexity. However, $I(\mathbf{u}_m; T)$ is only a first-order approximation of the ideal influence $I_-(x_m; \mathbf{V}_n)$. Since it stacks all eigenvectors into a long vector, it can only measure the absolute changes of each individual vector. As we have mentioned earlier, an arbitrary rotation of $\{\mathbf{v}_1, \dots, \mathbf{v}_c\}$ in the same space has *zero* influence on the value of $I_-(x_m; \mathbf{V}_n)$ in (5.28), but it has significant influence on the value of $I(\mathbf{u}_m; T)$.

5.4.3 Multivariate Trimming

The influence function method essentially tries to reject part of the samples (as outliers) from the data set using the standard PCA algorithm. However, since the principal vectors found by PCA are eigenvectors of the corresponding covariance matrix, we can also seek a robust estimator of the covariance matrix that can eliminate the outlier effect. One of the popular robust estimators is the multivariate trimming (MVT) method [?] because of its computational efficiency and tolerance of large percentage of outliers. The basic idea is that the outliers typically have larger Mahalanobis distance with respect to the correct covariance matrix Σ .¹⁶ In the context of PCA, the Mahalanobis distance of a sample \mathbf{x} is approximated with the weighted sum of all PCs

$$\mathbf{x}^T \hat{\Sigma}^{-1} \mathbf{x} = \sum_{i=1}^{M_n(D)} \frac{z_i^2}{l_i},$$

where $\hat{\Sigma}$ is the empirical covariance of the data set $\{\mathbf{x}_k\}$, z_i is the i th PC of the sample \mathbf{x} , and l_i is the variance of the i th PC [?].

The first step of MVT is a robust estimator of the data mean $\bar{\mathbf{u}}$ of the samples $\{\mathbf{u}_i = \nu_n(\mathbf{x}_i)\}$ [?]. The user needs to specify a trimming parameter α , which indicates the percentage of samples to be trimmed as outliers. To initialize the covariance matrix Σ_0 , all samples are sorted by their Euclidean distance $\|\mathbf{u}_i - \bar{\mathbf{u}}\|$, and Σ_0 is calculated with the set of the first $N(1 - \alpha)\%$ samples with the smallest distance, denoted as U :

$$\Sigma_0 = \sum_{h \in U} (\mathbf{u}_h - \bar{\mathbf{u}})(\mathbf{u}_h - \bar{\mathbf{u}})^T. \quad (5.33)$$

¹⁶Given a positive definite matrix Σ (such as the covariance matrix), the Mahalanobis distance is defined to be $\mathbf{x}^T \Sigma^{-1} \mathbf{x}$.

In the k th iteration, the Mahalanobis distance of each sample, $(\mathbf{u}_i - \bar{\mathbf{u}})^T \Sigma_{k-1}^{-1} (\mathbf{u}_i - \bar{\mathbf{u}})$, is calculated, and Σ_k is again calculated with the set of first $N(1 - \alpha)\%$ samples with the smallest Mahalanobis distance. The iteration terminates when the difference between Σ_{k-1} and Σ_k is small enough.

To proceed with the rest of the subspace segmentation algorithm, we treat the trimmed samples in the final iteration as outliers, and estimate $P(\mathbf{x})$ from the last c eigenvectors of the final covariance matrix.

5.4.4 Simulation Comparison

A. Simulation with Outlier Percentage Known.

We test and compare the performance of the three robust methods discussed in the previous section. The GPCA-voting algorithm introduced earlier in this chapter is compared to its robust version which we call Robust GPCA (RGPCA). The only difference in the RGPCA algorithm is in Step 3 of the GPCA-voting Algorithm ??: the fitting polynomials $P(\mathbf{x})$ are estimated with the robust PCA techniques discussed earlier in this section.

Two synthetic data sets are tested for the comparison. Data set one: three subspaces in \mathbb{R}^3 of dimensions 2, 2, 1, with 400, 400, 200 sample points drawn from the subspaces, respectively. Data set two: four subspaces in \mathbb{R}^5 of dimensions 4, 2, 2, 1, with 600, 400, 400, 300 sample points. For each data set, we generate an additional set of outliers with percentages ranging from 0% to 48% of the total sample number, and the rejection rate in the algorithm is set to be the same as the true outlier percentage. At each percentage level, the simulation is repeated for 200 times.

Figure 5.10 shows the segmentation error and subspace angle difference (in degree). Notice that because of randomness, some outliers may be very close to the subspaces and are not rejected as such; some samples might be rejected as outliers because large noise. Thus, the segmentation error does not count the original outliers and the rejected samples by the algorithms, as they are not assigned to any subspace. Figure 5.11 shows two representative experiments. Table 5.4 summarizes the average running time of the Matlab codes for one trial on a dual 2.7GHz G5 Macintosh workstation. One may wonder why the segmentation error of MTV decreases in the plot. We believe it is due to the fact that since some random outliers may lie on the subspaces, the algorithm might trim out more noisy “good” samples when the rejection rate increases.

To summarize, given a sample rejection rate that is close to the true outlier percentage, the robust covariance estimator turns out to be the most accurate and fastest method for solving the subspace estimation and segmentation problem. The corresponding RGPCA algorithm can tolerate as much as 50% outliers. The sample influence method also gives reasonable results when the outliers are less than 30%, but it is the slowest among the three. The theoretical influence method gives slightly worse segmentation than the sample influence method at high outlier levels, but it is much faster.

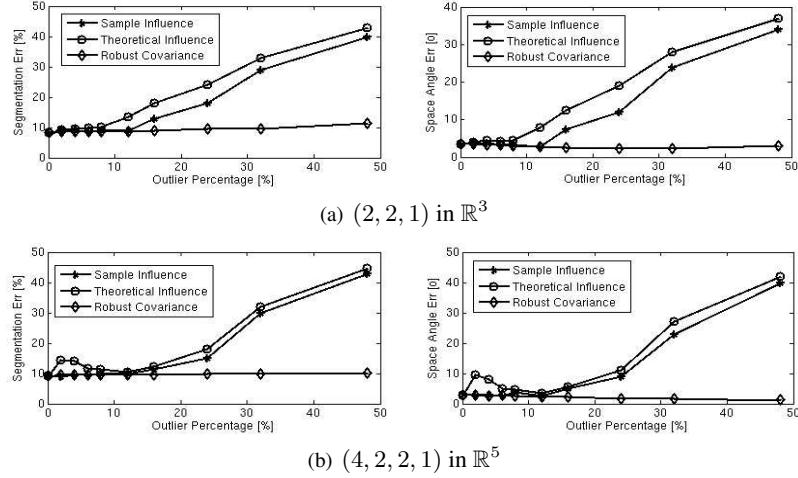


Figure 5.10. Performance of RGPCA. For each outlier percentage, 6% Gaussian noise is added in for each sample, and the experiment repeats 200 times.

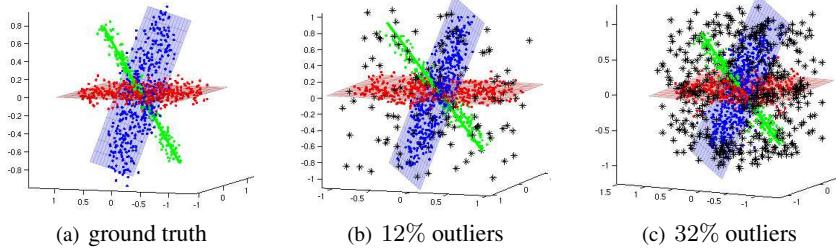


Figure 5.11. Two segmentation results (on the right) of RGPCA with the sample influence method. The black stars are rejected samples by the algorithm.

Table 5.4. Average computing time for the arrangement $(2, 2, 1)$ in \mathbb{R}^3 with sample sizes (400, 400, 200).

Outlier Percentage	0%	4%	8%	16%	24%	32%	48%
Data Size	1000	1042	1087	1190	1316	1470	1923
Sample Influence	5.4s	2.5m	2.8m	3.7m	5m	7.8m	18m
Theoretical Influence	5.4s	9s	9.2s	9.3s	9.3s	9.6s	10.8s
MVT	5.4s	5.4s	5.5s	5.6s	5.7s	5.7s	5.8s

B. Simulation with Outlier Percentage Over-Estimated.

In the above experiments, the correct outlier percentage is given to the algorithm. However, in practice, that information is not always available and it is very difficult to estimate. Thus, we need to study the performance of the algorithms when the percentage is under-estimated or over-estimated.

The situation with under-estimation is relatively simple. Since in general outliers have larger influence, or larger Mahalanobis distance, only part of the outliers will be rejected. After trimming out a smaller portion of the outliers, the remaining outliers will randomly perturb the estimates, as illustrated at the beginning of Section ??.

Thus, in order to obtain satisfactory estimates, we should always try to avoid under-estimate the percentage of outliers. Nevertheless, in the over-estimation situation the performance differs for the three robust methods. Figure 5.12 shows the segmentation results of the three methods with a fixed percentage of outliers in the data set but with varying rejection rate assigned to the algorithms. Figure 5.13 illustrates the difference of the three methods in over-estimation.

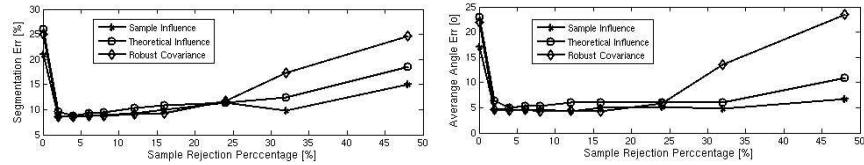


Figure 5.12. The segmentation and angle errors of the three methods on $(2, 2, 1)$ in \mathbb{R}^3 with different rejection rates. The subspaces are fixed with sizes $(300, 200, 100)$, 6% noise and 4% outliers.

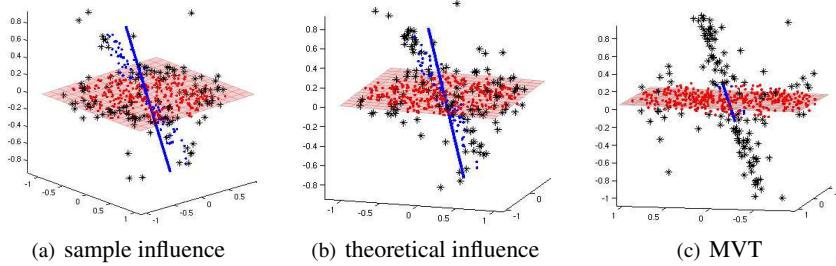


Figure 5.13. A plane (400 samples) and a line (100 samples) in \mathbb{R}^3 with 6% noise and 4% outliers. The rejection rate is 24%. The black stars are the samples rejected by the algorithms.

For the sample influence method, when the rejection rate is greater than the true outlier percentage, samples of larger magnitude are rejected first from the sample set. The result of the theoretical influence method is similar to the sample influence method, as it is the first order approximation to it. As pointed out in [?], samples of larger magnitude are more informative of the subspaces. Hence, the estimation performance decreases when the rejection rate increases. Nevertheless, this is not so critical as long as there are enough good samples left on all the subspaces. Figure 5.12 shows that the average subspace angle error for the sample influence method is about only 5 degree for the rejection rate as high as 50%.

The covariance estimator is a greedy algorithm minimizing the weighted sum of all the PCs. It turns out the robust covariance estimator is dominated by the subspaces of more samples, and the subspaces that have the less samples will be trimmed out first. In Figure 5.12, the robust covariance method starts diverging at 24%, which is roughly the percentage of outliers plus the samples on the 1-D subspace.

In summary, the three robust methods work well only when the sample rejection rate is larger than the true outlier percentage but still with sufficient samples left on the subspaces. If the data points are evenly sampled from all subspaces, or an accurate estimate of the outlier percentage can be obtained, the MVT method is the most efficient and accurate among the three. However, when this condition is not met, one has to resort to either the sample influence method or its approximation, the theoretical influence method, depending on the requirement to balance the accuracy and speed.

C. Comparison with RANSAC.

We now examine the difficulties in applying RANSAC to the estimation of multiple subspaces. Let us consider the example of four subspaces in \mathbb{R}^5 of dimensions 4, 2, 2, 1, respectively. The minimal numbers of points needed to determine each subspace are 4, 2, 2, 1, respectively. Thus, a total of 9 subsamples is needed for a candidate estimate of the subspaces. By repeating the subsampling process for a sufficient number of times, RANSAC selects the model that achieves the highest consensus among all samples. Figure 5.14 shows the segmentation error and the subspace angle error in the estimate given by RANSAC for three subspaces of dimension (2, 2, 1) in \mathbb{R}^3 and four subspaces of dimension (4, 2, 2, 1) in \mathbb{R}^5 with the same setup as in the previous simulations.

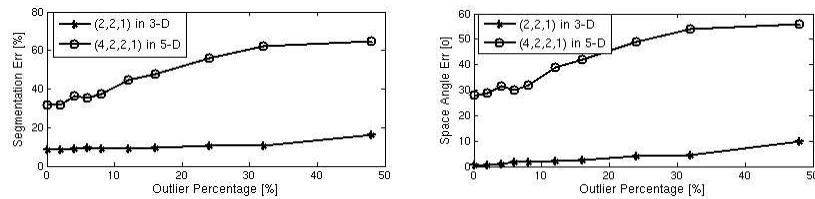


Figure 5.14. Performance of RANSAC for two subspace models. 6% Gaussian noise is added in for each sample, and the experiment repeats 200 times at each outlier level.

From the results, we see that RANSAC gave very accurate estimation for the (2, 2, 1) subspace model, but failed badly for the (4, 2, 2, 1) case (even with 0% outlier). The speed of the algorithm is also slower than RGPCA with the theoretical influence method and the MVT method. From a statistical viewpoint, RANSAC is not well-conditioned in dealing with multiple subspaces of *different dimensions*, because a collection of lower-dimensional subspaces can be modeled as a higher-dimensional subspace. In our example, on one hand, the union of the two 2-D subspaces can be confused with a 4-D subspace; and on the other hand,

subsets of samples from the 4-D subspace can be mistakenly fit with various 1-D or 2-D models. This type of erroneous model may also achieve high consensus. A more thorough study reveals that in the context of multiple subspaces, such a straightforward implementation of RANSAC scales very poorly with the number subspaces and their dimensions.

There are also alternative ways of implementing RANSAC. For instance, one can apply RANSAC to estimate one subspace at a time, or to reduce the complexity one can apply GPCA to a subsampled set. In the first case, the maximal consensus for each subspace can be extremely low, as its samples consist of only a small portion of the data set. In the second case, the size of the minimal sample set is difficult to determine, and it is not clear how many subsampled sets are needed in order for GPCA to find a good model with a high consensus. Our analysis (not shown in the paper due to limit of space) shows that these two approaches often result in prohibitive complexity when the number of subspace is large and the dimensions of the subspaces are high.

5.5 Bibliographic Notes

+ This is page 114
Printer: Opaque this

Part II

Applications in Image Processing & Computer Vision

+ This is page 116
Printer: Opaque this

Chapter 6

Image Representation, Segmentation & Classification

In this chapter, we demonstrate why subspace arrangements can be a very useful class of models for image processing and how the subspace-segmentation techniques may facilitate many important image processing tasks, such as image representation (compression), segmentation, and classification.

6.1 Lossy Image Representation

Researchers in image processing and computer vision have long sought for efficient and sparse representations of images. Except for a few image representations such as fractal-based approaches [?], most existing sparse image representations use an effective linear transformation so that the energy of the (transformed) image will be concentrated in the coefficients of a small set of bases of the transformation. Computing such a representation is typically the first step of subsequent (lossy) compression of the image.¹ The result can also be used for other purposes such as image segmentation,² classification, and object recognition.

Most of the popular methods for obtaining a sparse representation of images can be roughly classified into two categories.

¹Which involves further quantization and entropy-coding of the so-obtained representation.

²As we will study in the next section.

1. Fixed-Basis Linear Transformations.

Methods of the first category seek to transform all images using a *pre-fixed* linear transformation. Each image is then represented as a superposition of a set of basis functions (specified by the transformation). These methods essentially all evolved from the classical Fourier Transform. One variation of the (discrete) Fourier Transform, the Discrete Cosine Transform (DCT), serves as the core of the JPEG standard [?]. Due to the Gibbs' phenomenon, DCT is poor at approximating discontinuities in the imagery signal. Wavelets [?, ?, ?, ?] have been developed to remedy this problem and have been shown to be optimal for representing 1-D signals with discontinuities. JPEG-2000 adopted wavelets as its standard. However, because wavelet transforms only deal with 1-D discontinuities, they are not well-suited to represent 2-D singularities along edges or contours. Anisotropic bases such as wedgelets [?], curvelets [?], countourlets [?] and bandlets [?] have been proposed explicitly to capture different 2-D discontinuities. These x-lets have been shown to be (approximately) optimal for representing objects with singularities along C^2 -smooth edges.³

However, natural images, especially images that have complex textures and patterns, do not consist solely of discontinuities along C^2 -smooth edges. This is probably the reason why these edge-based methods do not seem to outperform (separable) wavelets on complex images. More generally, one should not expect that a (fixed) “gold-standard” transformation would work optimally for all images (and signals) in the world. Furthermore, conventional image (or signal) processing methods are developed primarily for gray-scale images. For color images or other multiple-valued images, one has to apply them to each value separately (e.g., one color channel at a time). The strong correlation that is normally present among the multiple values or colors is unfortunately ignored.

2. Adaptive Transformations & Hybrid Models

Methods of the second category aim to identify the optimal (or approximately optimal) representation that is *adaptive* to specific statistics or structures of each image.⁴ The Karhunen-Loëve transform (KLT) or principal component analysis (PCA) [?] identifies the optimal principal subspace from the statistical correlation of the imagery data and represents the image as a superposition of the basis of the subspace. In theory, PCA provides the optimal linear sparse representation assuming that the imagery data satisfy a uni-modal distribution. However in reality, this assumption is rarely true. Natural images typically exhibit multi-modal statistics as they usually contain many heterogeneous regions with significantly different geometric structures or statistical characteristics (e.g. Figure 6.2). Heterogeneous

³Here, “optimality” means that the transformation achieves the optimal asymptotic for approximating the class of functions considered [?].

⁴Here, unlike in the case of prefixed transformations, “optimality” means the representation obtained is the optimal one within the class of models considered, in the sense that it minimizes certain discrepancy between the model and the data.

data can be better-represented using a mixture of parametric models, one for each homogeneous subset. Such a mixture of models is often referred to as a *hybrid model*. Vector quantization (VQ) [?] is a special hybrid model that assumes the imagery data are clustered around many different centers. From the dimension reduction point of view, VQ represents the imagery data with many 0-dimensional (affine) subspaces. This model typically leads to an excessive number of clusters or subspaces.⁵ The primal sketch model [?] is another hybrid model which represents the high entropy parts of images with Markov random fields [?, ?] and the low entropy parts with sketches. The result is also some kind of a “sparse” representation of the image as superposition of the random fields and sketches. However, the primary goal of primal sketch is not to authentically represent and approximate the original image. It is meant to capture the (stochastic) generative model that produces the image (as random samples). Therefore, this type of models are more suited for image parsing, recognition, and synthesis than approximation and compression. In addition, finding the sketches and estimating the parameters of the random fields are computationally expensive and therefore less appealing for developing efficient image representation and compression schemes.

In this chapter, we would like to show how to combine the benefits of PCA and VQ by representing an image with multiple (affine) subspaces – one subspace for one image segment. The dimension and basis of each subspace are pertinent to the characteristics of the image segment it represents. We call this a *hybrid linear model* and will show that it strikes a good balance between simplicity and expressiveness for representing natural images.

A Multi-Scale Hybrid Linear Model for Lossy Image Representation.

One other important characteristic of natural images is that they are comprised of structures at many different (spatial) scales. Many existing frequency-domain techniques harness this characteristic [?]. For instance, wavelets, curvelets, and fractals have all demonstrated effectiveness in decomposing the original imagery signal into multiple scales (or subbands). As the result of such a *multi-scale* decomposition, the structures of the image at different scales (e.g., low v.s. high frequency/entropy) become better exposed and hence can be more compactly represented. The availability of multi-scale structures also significantly reduces the size and dimension of the problem and hence reduces the overall computational complexity.

Therefore, in this chapter we introduce a new approach to image representation by combining the hybrid paradigm and the multi-scale paradigm. The result is a *multi-scale hybrid linear model* which is based on an extremely simple concept: Given an image, at each scale level of its down-sample pyramid, fit the (residual)

⁵Be aware that compared to methods in the first category, representations in the second category typically need additional memory to store the information about the resulting model itself, e.g., the basis of the subspace in PCA, the cluster means in VQ.

image by a (multiple-subspace) hybrid linear model. Compared to the single-scale hybrid linear model, the multi-scale scheme can reduce not only the size of the resulting representation but also the overall computational cost. Surprisingly, as we will demonstrate, such a simple scheme is able to generate representations for natural images that are more compact, even with the overhead needed to store the model, than most state-of-the-art representations, including DCT, PCA, and wavelets.

6.1.1 A Hybrid Linear Model

In this section we introduce and examine the hybrid linear model for image representation. The relationship between hybrid linear models across different spatial scales will be discussed in Section 6.1.2.

An image \mathbf{I} with width W , height H , and c color channels resides in a very high-dimensional space $\mathbb{R}^{W \times H \times c}$. We may first reduce the dimension by dividing the image into a set of non-overlapping b by b blocks.⁶ Each b by b block is then stacked into a vector $\mathbf{x} \in \mathbb{R}^D$, where $D = b^2c$ is the dimension of the ambient space. For example, if $c = 3$ and $b = 2$, then $D = 12$. In this way, the image \mathbf{I} is converted to a set of vectors $\{\mathbf{x}_i \in \mathbb{R}^D\}_{i=1}^N$, where $N = WH/b^2$ is the total number of vectors.

Borrowing ideas from existing unsupervised learning paradigms, it is tempting to assume the imagery data $\{\mathbf{x}_i\}$ are random samples from a (non-singular) probability distribution or noisy samples from a smooth manifold. As the distribution or manifold can be very complicated, a common approach is to infer a best approximation within a simpler class of models for the distributions or manifolds. The “optimal” model is then the one that minimizes certain distance to the true model. Different choices of model classes and distance measures have led to many different learning algorithms developed in machine learning, pattern recognition, computer vision, and image processing. The most commonly adopted distance measure, for image compression, is the Mean Square Error (MSE) between the original image \mathbf{I} and approximated image $\hat{\mathbf{I}}$,

$$\epsilon_I^2 = \frac{1}{WHc} \|\hat{\mathbf{I}} - \mathbf{I}\|^2. \quad (6.1)$$

Since we will be approximating the (block) vectors $\{\mathbf{x}_i\}$ rather than the image pixels, in the following derivation, it is more convenient for us to define the Mean Square Error (MSE) *per vector* which is different from ϵ_I^2 by a scale,

$$\epsilon^2 = \frac{1}{N} \sum_{i=1}^N \|\hat{\mathbf{x}}_i - \mathbf{x}_i\|^2 = \frac{b^2}{WH} \sum_{i=1}^N \|\hat{\mathbf{x}}_i - \mathbf{x}_i\|^2 = \frac{b^2}{WH} \|\hat{\mathbf{I}} - \mathbf{I}\|^2 = (b^2c)\epsilon_I^2. \quad (6.2)$$

⁶Therefore, b needs to be a common divisor of W and H .

The Peak Signal to Noise Ratio (PSNR) of the approximated image is defined as,⁷

$$\text{PSNR} \doteq -10 \log \epsilon_I^2 = -10 \log \frac{\epsilon^2}{b^2 c}. \quad (6.3)$$

Linear Models.

If we assume that the vectors \mathbf{x} are drawn from an anisotropic Gaussian distribution or a linear subspace, the optimal model subject to a given PSNR can be inferred by Principal Component Analysis (PCA) [Pearson, 1901, Hotelling, 1933, Jolliffe, 2002] or equivalently the Karhunen-Loëve Transform (KLT) [?]. The effectiveness of such a linear model relies on the assumption that, although D can be large, all the vectors \mathbf{x} may lie in a subspace of a much lower dimension in the ambient space \mathbb{R}^D . Figure 6.1 illustrates this assumption.

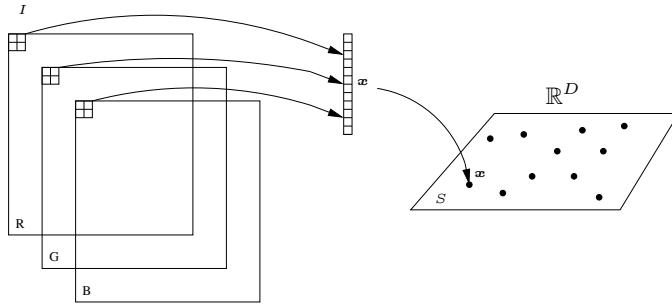


Figure 6.1. In a linear model, the imagery data vectors $\{\mathbf{x}_i \in \mathbb{R}^D\}$ reside in an (affine) subspace S of dimension $d \ll D$.

Let $\bar{\mathbf{x}} = \frac{1}{N} \sum_{i=1}^N \mathbf{x}_i$ be the mean of the imagery data vectors, and $\mathbf{X} = [\mathbf{x}_1 - \bar{\mathbf{x}}, \mathbf{x}_2 - \bar{\mathbf{x}}, \dots, \mathbf{x}_N - \bar{\mathbf{x}}] = U\Sigma V^T$ be the SVD of the mean-subtracted data matrix \mathbf{X} . Then all the vectors \mathbf{x}_i can be represented as a linear superposition: $\mathbf{x}_i = \bar{\mathbf{x}} + \sum_{j=1}^D \alpha_i^j \phi_j$, $i = 1, \dots, N$, where $\{\phi_j\}_{j=1}^D$ are just the columns of the matrix U .

The matrix $\Sigma = \text{diag}(\sigma_1, \sigma_2, \dots, \sigma_D)$ contains the ordered singular values $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_D$. It is well known that the optimal linear representation of \mathbf{x}_i subject to the MSE ϵ^2 is obtained by keeping the first d (principal) components

$$\hat{\mathbf{x}}_i \doteq \bar{\mathbf{x}} + \sum_{k=1}^d \alpha_i^k \phi_k, \quad i = 1, \dots, N, \quad (6.4)$$

where d is chosen to be

$$d = \min(k), \quad \text{s.t.} \quad \frac{1}{N} \sum_{i=k+1}^D \sigma_i^2 \leq \epsilon^2. \quad (6.5)$$

⁷The peak value of the imagery data is normalized into 1.

The model complexity of the linear model, denoted as Ω , is the total number of coefficients needed for representing the model $\{\alpha_i^k, \phi_k, \bar{x}\}$ and subsequently a lossy approximation \hat{I} of the image I . It is given by

$$\Omega(N, d) \doteq Nd + d(D - d + 1), \quad (6.6)$$

where the first term is the number of coefficients $\{\alpha_i^k\}$ to represent $\{\hat{x}_i - \bar{x}\}_{i=1}^N$ with respect to the basis $\Phi = \{\phi_k\}_{k=1}^d$ and the second term is the number of Grassmannian coordinates⁸ needed for representing the basis Φ and the mean vector \bar{x} . The second term is often called *overhead*.⁹ Notice that the original set of vectors $\{\mathbf{x}_i\}$ contain ND coordinate entries. If $\Omega \ll ND$, the new representation, although lossy, is more compact. The search for such a compact representation is at the heart of any (lossy) image compression method. When the image I is large and the block size b is small, N will be much larger than D so that the overhead will be much smaller than the first term. However, in order to compare fairly with other methods, in the subsequent discussions and experiments, we always count the total number of coefficients needed for the representation, including the overhead.

Hybrid Linear Models.

The linear model is very efficient when the target manifold or distribution function is indeed unimodal. However, if the image I contains several heterogeneous regions $\{I_j\}_{j=1}^n$, the data vectors \mathbf{x}_i can be samples from a collection of subspaces of possibly different dimensions or from a mixture of multiple (Gaussian) distributions. Figure 6.2 shows the first three principal components of the data

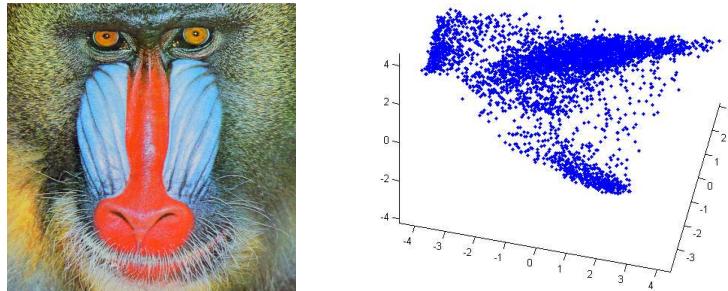


Figure 6.2. Left: The baboon image. Right: The coordinates of each dot are the first three principal components of the vectors \mathbf{x}_i . There is a clear multi-modal structure in the data.

⁸Notice that to represent a d -dimensional subspace in a D -dimensional space, we only need to specify a basis of d linearly independent vectors for the subspace. We may stack these vectors as rows of a $d \times D$ matrix. Any nonsingular linear transformation of these vectors span the same subspace. Thus, without loss of generality, we may assume that the matrix is of the normal form $[I_{d \times d}, G]$ where G is a $d \times (D - d)$ matrix consisting of the so-called Grassmannian coordinates.

⁹Notice that if one uses a pre-chosen basis such as discrete Fourier transform, discrete cosine transform (JPEG), and wavelets (JPEG-2000), there is no such overhead.

vector \mathbf{x}_i (as dots in \mathbb{R}^3) of an image. Note the clear multi-modal characteristic in the data.

Suppose that a natural image \mathbf{I} can be segmented into n disjoint regions $\mathbf{I} = \cup_{j=1}^n \mathbf{I}_j$ with $\mathbf{I}_j \cap \mathbf{I}_{j'} = \emptyset$ for $j \neq j'$. In each region \mathbf{I}_j , we may assume the linear model (6.4) is valid for the subset of vectors $\{\mathbf{x}_{j,i}\}_{i=1}^{N_j}$ in \mathbf{I}_j :

$$\hat{\mathbf{x}}_{j,i} = \bar{\mathbf{x}}_j + \sum_{k=1}^{d_j} \alpha_i^k \phi_{j,k}, \quad i = 1, \dots, N_j. \quad (6.7)$$

Intuitively, the hybrid linear model can be illustrated by Figure 6.3.

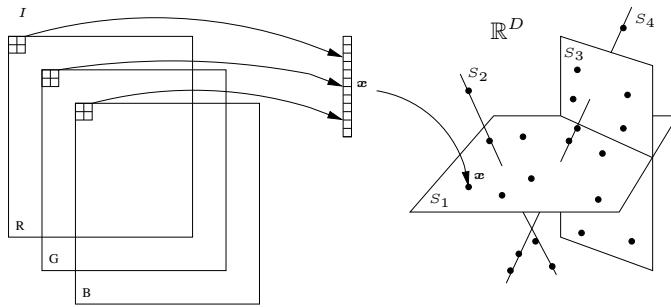


Figure 6.3. In hybrid linear models, the imagery data vectors $\{\mathbf{x}_i\}$ reside in multiple (affine) subspaces which may have different dimensions.

As in the linear model, the dimension d_j of each subspace is determined by a common desired MSE ϵ^2 using equation (6.5). The model complexity, i.e., the total number of coefficients needed for representing the hybrid linear model $\{\phi_{j,k}, \hat{\mathbf{x}}_{j,i}\}$ is¹⁰

$$\Omega = \Omega(N_1, d_1) + \dots + \Omega(N_n, d_n) = \sum_{j=1}^n (N_j d_j + d_j(D - d_j + 1)). \quad (6.8)$$

Notice that Ω is similar to the effective dimension (ED) of the hybrid linear representation defined in [?]. Thus, finding a representation that minimizes Ω is the same as minimizing the effective dimension of the imagery data set.¹¹

Instead, if we model the union of all the vectors $\cup_{j=1}^n \{\mathbf{x}_{j,i}\}_{i=1}^{N_j}$ with a single subspace (subject to the same MSE), the dimension of the subspace in general needs to be $d = \min\{d_1 + \dots + d_n, D\}$. It is easy to verify from the definition (6.6) that under reasonable conditions (e.g., n is bounded from being too large),

¹⁰We also need a very small number of binary bits to store the membership of the vectors. But those extra bits are insignificant comparing to Ω and often can be ignored.

¹¹In fact, the minimal Ω can also be associated to the Kolmogorov entropy or to the minimum description length (MDL) of the imagery data.

we have

$$\Omega(N, d) > \Omega(N_1, d_1) + \cdots + \Omega(N_n, d_n). \quad (6.9)$$

Thus, if a hybrid linear model can be identified for an image, the resulting representation will in general be much more compressed than that with a single linear or affine subspace. This will also be verified by experiments on real images in Section 6.1.3.

However, such a hybrid linear model alone is not able to generate a representation that is as compact as that by other competitive methods such as wavelets. There are at least two aspects in which the above model can be further improved. Firstly, we need to further reduce the negative effect of overhead by incorporating a pre-projection of the data onto a lower dimensional space. Secondly, we need to implement the hybrid linear model in a multi-scale fashion. We will discuss the former aspect in the remainder of this section and leave the issues with multi-scale implementation to the next section.

Dimension Reduction via Projection.

In the complexity of the hybrid linear model (6.8), the first term is always smaller than that of the linear model (6.6) because $d_j \leq d$ for all j and $\sum_{j=1}^n N_j = N$. The second overhead term however can be larger than in that of the linear model (6.6) because the bases of multiple subspaces now must be stored. We here propose a method to further reduce the overhead by separating the estimation of the hybrid model into two steps.

In the first step, we may project the data vectors $\{\mathbf{x}_i\}$ onto a lower-dimensional subspace (e.g., via PCA) so as to reduce the dimension of the ambient space from D to D' . The justification for such a subspace projection has been discussed earlier in Section 3.2.2. Here, the dimension D' is chosen to achieve an MSE $\frac{1}{2}\epsilon^2$. The data vectors in the lower ambient space $\mathbb{R}^{D'}$ are denoted as $\{\mathbf{x}'_i\}$. In the second step, we identify a hybrid linear model for $\{\mathbf{x}'_i\}$ within the lower-dimension ambient space $\mathbb{R}^{D'}$. In each subspace, we determine the dimension d_j subject to the MSE $\frac{1}{2}\epsilon^2$. The two steps combined achieve an overall MSE ϵ^2 , but they can actually reduce the total model complexity to

$$\Omega = \sum_{j=1}^n (N_j d_j + d_j(D' - d_j + 1)) + D(D' + 1). \quad (6.10)$$

This Ω will be smaller than the Ω in equation (6.8) because D' is smaller than D . The reduction of the ambient space will also make the identification of the hybrid linear model (say by GPCA) much faster.

If the number of subspaces, n , is given, algorithms like GPCA or EM can always find a segmentation. The basis $\{\phi_{j,k}\}$ and dimension d_j of each subspace are determined by the desired MSE ϵ^2 . As n increases, the dimension of the subspaces may decrease, but the overhead required to store the bases may increase. The optimal n^* therefore can be found recursively by minimizing Ω for different n 's, as shown in Figure 6.4. From our experience, we found that n is typically in the

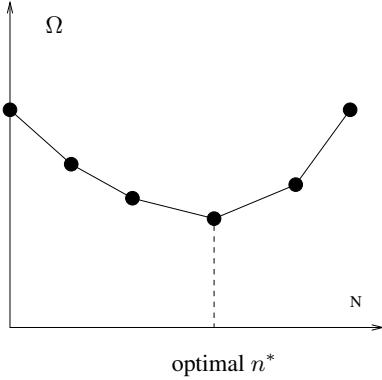


Figure 6.4. The optimal n^* can be found by minimizing Ω with respect to n .

range from 2 to 6 for natural images, especially in a multi-scale implementation that we will introduce next.

Algorithm 6.1 describes the pseudocode for estimating the hybrid linear model of an image \mathbf{I} , in which the *SubspaceSegmentation(·)* function is implemented (for the experiments in this chapter) using the GPCA algorithm given in earlier chapters. But it can also be implemented using EM or other subspace segmentation methods.

Algorithm 6.1 (Hybrid Linear Model Estimation).

```

1: function  $\hat{\mathbf{I}} = \text{HybridLinearModel}(\mathbf{I}, \epsilon^2)$ 
2:  $\{\mathbf{x}_i\} = \text{StackImageIntoVectors}(\mathbf{I});$ 
3:  $\{\mathbf{x}'_i\}, \{\phi_k\}, \{\alpha_i^k\} = \text{PCA}(\{\mathbf{x}_i - \bar{\mathbf{x}}\}, \frac{1}{2}\epsilon^2);$ 
4: for each possible  $n$  do
5:    $\{\mathbf{x}'_{j,i}\} = \text{SubspaceSegmentation}(\{\mathbf{x}'_i\}, n);$ 
6:    $\{\hat{\mathbf{x}}'_{j,i}\}, \{\phi_{j,k}\}, \{\alpha_{j,i}^k\} = \text{PCA}(\{\mathbf{x}'_{j,i} - \bar{\mathbf{x}}'_j\}, \frac{1}{2}\epsilon^2);$ 
7:   compute  $\Omega_n$ ;
8: end for
9:  $\Omega_{opt} = \min(\Omega_n);$ 
10:  $\hat{\mathbf{I}} = \text{UnstackVectorsIntoImage}(\{\hat{\mathbf{x}}'_{j,i}\} \text{ with } \Omega_{opt});$ 
11: output  $\{\alpha_i^k\}, \{\phi_k\}, \bar{\mathbf{x}}, \{\alpha_{j,i}^k\}, \{\phi_{j,k}\}, \{\bar{\mathbf{x}}'_j\}$  with  $\Omega_{opt};$ 
12: return  $\hat{\mathbf{I}}.$ 

```

Example 6.1 (A Hybrid Linear Model for the Gray-Scale Barbara Image). Figure 6.5 and Figure 6.6 show intuitively a hybrid linear model identified for the 8×8 blocks of the standard 512×512 gray-scale Barbara image. The total number of blocks is $N = 4,096$. The GPCA algorithm identifies three subspaces for these blocks (for a given error tolerance), as shown in Figure 6.5. Figure 6.6 displays the three sets of bases for the three



Figure 6.5. The segmentation of the 4,096 image blocks from the Barbara image. The image (left) is segmented into three groups (right three). Roughly speaking, the first subspace contains mostly image blocks with homogeneous textures; the second and third subspaces contain blocks with textures of different spatial orientations and frequencies.

subspaces identified, respectively. It is worth noting that these bases are very consistent with the textures of the image blocks in the respective groups. ■

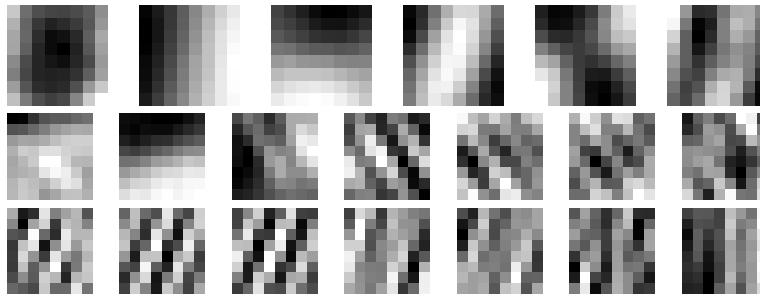


Figure 6.6. The three sets of bases for the three subspaces (of blocks) shown in Figure 6.5, respectively. One row for one subspace and the number of base vectors (blocks) is the dimension of the subspace.

6.1.2 Multi-Scale Hybrid Linear Models

There are at least several reasons why the above hybrid linear model needs further improvement. Firstly, the hybrid linear model treats low frequency/entropy regions of the image in the same way as the high frequency/entropy regions, which is inefficient. Secondly, by treating all blocks the same, the hybrid linear model fails to exploit stronger correlations that typically exist among adjacent image blocks.¹² Finally, estimating the hybrid linear model is computationally expensive when the image is large. For example, we use 2 by 2 blocks, a 512 by 512 color image will have $M = 65,536$ data vectors in \mathbb{R}^{12} . Estimating a hybrid linear model for such a huge number of vectors is difficult (if not impossible) on a regular PC. In this section, we introduce a multi-scale hybrid linear representation which is able to resolve the above issues.

¹²For instance, if we take all the b by b blocks and scramble them arbitrarily, the scrambled image would be fit equally well by the same hybrid linear model for the original image.

The basic ideas of multi-scale representations such as the Laplacian pyramid [?] have been exploited for image compression for decades (e.g., wavelets, sub-band coding). A multi-scale method will give a more compact representation because it encodes low frequency/entropy parts and high frequency/entropy parts separately. The low frequency/entropy parts are invariant after low-pass filtering and down-sampling, and can therefore be extracted from the much smaller down-sampled image. Only the high frequency/entropy parts need to be represented at a level of higher resolution. Furthermore, the stronger correlations among adjacent image blocks will be captured in the down-sampled images because every four images blocks are merged into one block in the down-sampled image. At each level, the number of imagery data vectors is one fourth of that at one level above. Thus, the computational cost can also be reduced.

We now introduce a multi-scale implementation of the hybrid linear model. We use the subscript l to indicate the level in the pyramid of down-sampled images.¹³ The finest level (the original image) is indicated by $l = 0$. The larger is l , the coarser is the down-sampled image. We denote the highest level to be $l = L$.

Pyramid of Down-Sampled Images.

First, the level- l image \mathbf{I}_l passes a low-pass filter F_1 (averaging or Gaussian filter, etc) and is down-sampled by 2 to get a coarser version image \mathbf{I}_{l+1} :

$$\mathbf{I}_{l+1} \doteq F_1(\mathbf{I}_l) \downarrow 2, \quad l = 0, \dots, L - 1. \quad (6.11)$$

The coarsest level- L image \mathbf{I}_L is approximated by $\hat{\mathbf{I}}_L$ using a hybrid linear model with the MSE ϵ_L^2 . The number of coefficients needed for the approximation is Ω_L .

Pyramid of Residual Images.

At all other level- l , $l = 0, \dots, L - 1$, we do *not* need to approximate the down-sampled image \mathbf{I}_l because it has been roughly approximated by the image at level- $(l + 1)$ upsampled by 2. We only need to approximate the residual of this level, denoted as \mathbf{I}'_l :

$$\mathbf{I}'_l \doteq \mathbf{I}_l - F_2(\hat{\mathbf{I}}_{l+1}) \uparrow 2, \quad l = 0, \dots, L - 1, \quad (6.12)$$

where the F_2 is an interpolation filter. Each of these residual images \mathbf{I}'_l , $l = 0, \dots, L - 1$ is approximated by $\hat{\mathbf{I}}'_l$ using a hybrid linear model with the MSE ϵ_l^2 . The number of coefficients needed for the approximation is Ω_l , for each $l = 0, \dots, L - 1$.

Pyramid of Approximated Images.

The approximated image at the level- l is denoted as $\hat{\mathbf{I}}_l$:

$$\hat{\mathbf{I}}_l \doteq \hat{\mathbf{I}}'_l + F_2(\hat{\mathbf{I}}_{l+1}) \uparrow 2, \quad l = 0, \dots, L - 1. \quad (6.13)$$

The Figure 6.7 shows the structure of a three-level ($L = 2$) approximation of the

¹³This is not to be confused with the subscript j used to indicate different segments of an image.

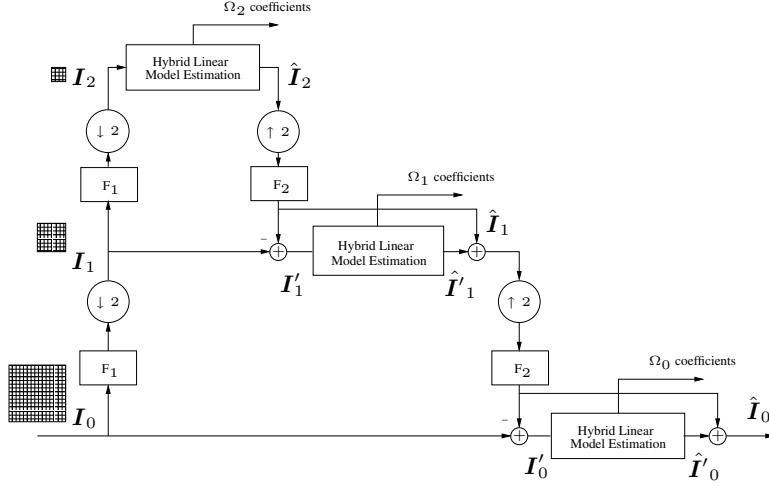


Figure 6.7. Laplacian pyramid of the multi-scale hybrid linear model.

image \mathbf{I} . Only the hybrid linear models for $\hat{\mathbf{I}}_2$, $\hat{\mathbf{I}}'_1$, and $\hat{\mathbf{I}}'_0$, which are approximation for \mathbf{I}_2 , \mathbf{I}'_1 , and \mathbf{I}'_0 respectively, are needed for the final representation of the image. Figure 6.8 shows the \mathbf{I}_2 , \mathbf{I}'_1 , and \mathbf{I}'_0 for the baboon image.



Figure 6.8. Multi-scale representation of the Baboon image. Left: The coarsest level image \mathbf{I}_2 . Middle: The residual image \mathbf{I}'_1 . Right: The residual image \mathbf{I}'_0 . The data at each level are modeled as the hybrid linear models. The contrast of the middle and right images has been adjusted so that they are visible.

The total number of coefficients needed for the representation will be

$$\Omega = \sum_{l=0}^L \Omega_l. \quad (6.14)$$

MSE Threshold at Different Scale Levels.

The MSE thresholds at different levels should be different but related because the up-sampling by 2 will enlarge 1 pixel at level- $(l+1)$ into 4 pixels at level- l . If the MSE of the level- $(l+1)$ is ϵ_{l+1}^2 , the MSE of the level- l after the up-sampling will become $4\epsilon_{l+1}^2$. So the MSE thresholds of level- $(l+1)$ and level- l are related as

$$\epsilon_{l+1}^2 = \frac{1}{4}\epsilon_l^2, \quad l = 0, \dots, L-1. \quad (6.15)$$

Usually, the user will only give the desired MSE for the approximation of original image which is ϵ^2 . So we have

$$\epsilon_l^2 = \frac{1}{4^l}\epsilon^2, \quad l = 0, \dots, L. \quad (6.16)$$

Vector Energy Constraint at Each Level.

At each level- l , $l = 0, \dots, L-1$, not all the vectors of the residual need to be approximated. We only need to approximate the (block) vectors $\{\mathbf{x}_i\}$ of the residual image \mathbf{I}'_l that satisfy the following constraint:

$$\|\mathbf{x}'_i\|^2 > \epsilon_l^2. \quad (6.17)$$

In practice, the energy of most of the residual vectors is close to zero. Only a small portion of the vectors at each level- l need to be modeled (e.g. Figure 6.9). This property of the multi-scale scheme not only significantly reduces the overall

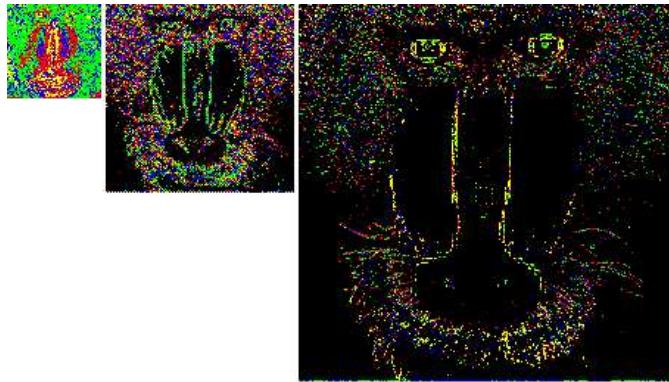


Figure 6.9. The segmentation of (residual) vectors at the three levels—different subspaces are denoted by different colors. The black regions correspond to data vectors whose energy is below the MSE threshold ϵ_l^2 in equation (6.17).

representation complexity Ω but also reduces the overall computational cost as the number of data vectors processed at each level is much less than those of the original image. In addition, for a single hybrid linear model, when the image size increases, the computational cost will increase in proportion to the square of the image size. In the multi-scale model, if the image size increases, we can

correspondingly increase the number of levels and the complexity increases only linearly in proportion to the image size.

The overall process of estimating the multi-scale hybrid linear model can be written as the recursive pseudocode in Algorithm 6.2.

Algorithm 6.2 (Multi-Scale Hybrid Linear Model Estimation).

```

1: function  $\hat{\mathbf{I}} = \text{MultiscaleModel}(\mathbf{I}, level, \epsilon^2)$ 
2: if  $level < \text{MAXLEVEL}$  then
3:    $\mathbf{I}_{down} = \text{Downsample}(\mathbf{F}_1(\mathbf{I}))$ ;
4:    $\hat{\mathbf{I}}_{nextlevel} = \text{MultiscaleModel}(\mathbf{I}_{down}, level + 1, \frac{1}{4}\epsilon^2)$ ;
5: end if
6: if  $level = \text{MAXLEVEL}$  then
7:    $\mathbf{I}' = \mathbf{I}$ ;
8: else
9:    $\mathbf{I}_{up} = \mathbf{F}_2(\text{Upsample}(\hat{\mathbf{I}}_{nextlevel}))$ ;
10:   $\mathbf{I}' = \mathbf{I} - \mathbf{I}_{up}$ ;
11: end if
12:  $\hat{\mathbf{I}}' = \text{HybridLinearModel}(\mathbf{I}', \epsilon^2)$ ;
13: return  $\mathbf{I}_{up} + \mathbf{I}'$ .

```

6.1.3 Experiments and Comparisons

Comparison of Different Lossy Representations.

The first experiment is conducted on two standard images commonly used to compare image compression schemes: the 480×320 hill image and the 512×512 baboon image shown in Figure 6.10. We choose these two images because they are representative of two different types of images. The hill image contains large low frequency/entropy regions and the baboon image contains mostly high frequency/entropy regions. The size of the blocks b is chosen to be 2 and the level of the pyramid is 3 – we will test the effect of changing these parameters in subsequent experiments. In Figure 6.11, the results of the multi-scale hybrid linear model are compared with several other commonly used image representations including DCT, PCA/KLT, single-scale hybrid linear model and Level-3 (Daubechies) biorthogonal 4.4 wavelets (adopted by JPEG-2000). The x -axis of the figures is the ratio of coefficients (including the overhead) kept for the representation, which is defined as,

$$\eta = \frac{\Omega}{WHc}. \quad (6.18)$$

The y -axis is the PSNR of the approximated image defined in equation (6.3). The



Figure 6.10. Testing images: the hill image (480×320) and the baboon image (512×512).

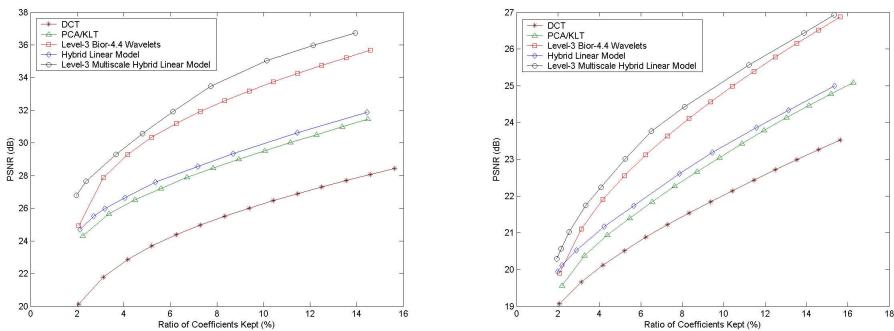


Figure 6.11. Left: Comparison of several image representations for the hill image. Right: Comparison for the baboon image. The multi-scale hybrid linear model achieves the best PSNR among all the methods for both images.

multi-scale hybrid linear model achieves the best PSNR among all the methods for both images. Figure 6.12 shows the two recovered images using the same amount of coefficients for the hybrid linear model and the wavelets. Notice that in the area around the whiskers of the baboon, the hybrid linear model preserves the detail of the textures better than the wavelets. But the multiscale hybrid linear model produces a slight block effect in the smooth regions.

We have tested the algorithms on a wide range of images. We will summarize the observations in Section 6.1.4.

Effect of the Number of Scale Levels.

The second experiment shown in Figure 6.13 compares the multi-scale hybrid linear representation with wavelets for different number of levels. It is conducted on the hill and baboon image with 2 by 2 blocks. The performance increases while the number of levels is increased from 3 to 4. But if we keep increasing the number

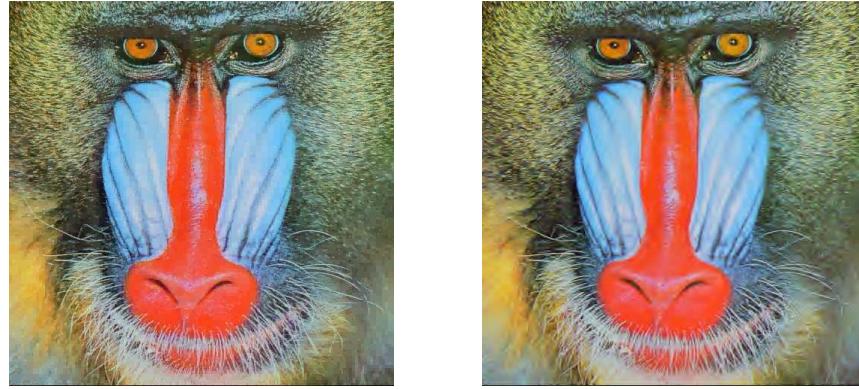


Figure 6.12. Left: The baboon image recovered from the multi-scale hybrid linear model using 7.5% coefficients of the original image. (PSNR=24.64). Right: The baboon image recovered from wavelets using the same amount of coefficients. (PSNR=23.94).

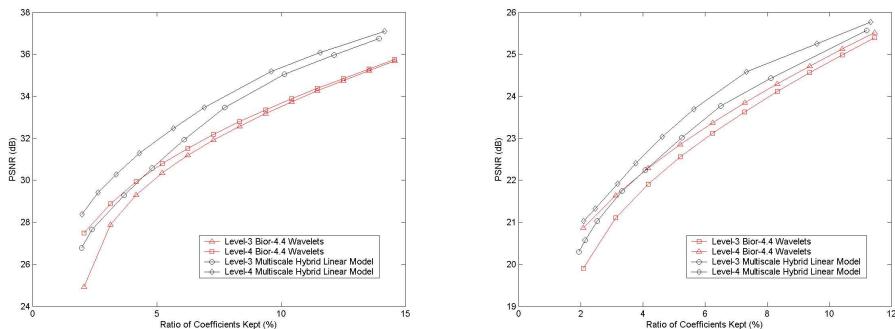


Figure 6.13. Top: Comparison of the multi-scale hybrid linear model with wavelets for level-3 and level-4 for the hill image. Bottom: The same comparison for the baboon image. The performance increases while the number of levels increases from 3 to 4.

of levels to 5, the level-5 curves of both wavelets and our method (which are not shown in the figures) coincide with the level-4 curves. The performance cannot improve any more because the down-sampled images in the fifth level are so small that it is hard to be further compressed. Only when the image is large, can we use more levels of down-sampling to achieve a more compressed representation.

Effect of the Block Size.

The third experiment shown in Figure 6.14 compares the multi-scale hybrid linear models with different block sizes from 2×2 to 16×16 . The dimension of the ambient space of the data vectors \mathbf{x} ranges from 12 to 192 accordingly. The testing image is the baboon image and the number of down-sampling levels is 3. For large blocks, the number of data vectors is small but the dimension of the

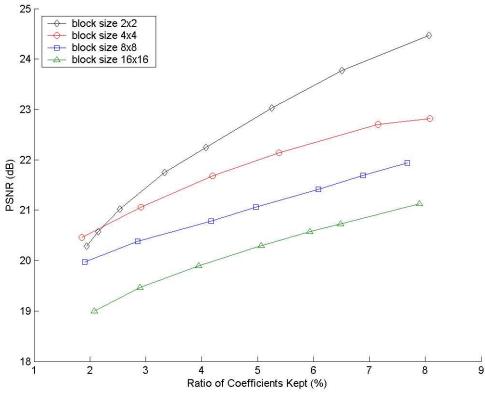


Figure 6.14. Comparison of the multi-scale hybrid linear model with different block sizes: 16, 8, 4, 2. The performance increases while the size of blocks decreases.

subspaces is large. So the overhead would be large and seriously degrade the performance. Also the block effect will be more obvious when the block size is large. This experiment shows that 2 is the optimal block size, which also happens to be compatible with the simplest down-sampling scheme.

6.1.4 Limitations

We have tested the multi-scale hybrid linear model on a wide range of images, with some representative ones shown in Figure 6.15. From our experiments and experience, we observe that the multi-scale hybrid linear model is more suitable than wavelets for representing images with multiple high frequency/entropy regions, such as those with sharp 2-D edges and rich of textures. Wavelets are prone to blur sharp 2-D edges but better at representing low frequency/entropy regions. This probably explains why the hybrid linear model performs slightly worse than wavelets for the Lena and the monarch – the backgrounds of those two images are out of focus so that they do not contain much high frequency/entropy content.

Another limitation of the hybrid-linear model is that it does not perform well on gray-scale images (e.g., the Barbara image, Figure 6.5). For a gray-scale image, the dimension D of a 2 by 2 block is only 4. Such a low dimension is not adequate for any further dimension reduction. If we use a larger block size, say 8 by 8, the block effect will also degrade the performance.

Unlike pre-fixed transformations such as wavelets, our method involves identifying the subspaces and their bases. Computationally, it is more costly. With unoptimized MATLAB codes, the overall model estimation takes 30 seconds to 3 minutes on a Pentium 4 1.8GHz PC depending on the image size and the desired PSNR. The smaller the PSNR, the shorter the running time because the number of blocks needed to be coded in higher levels will be less.

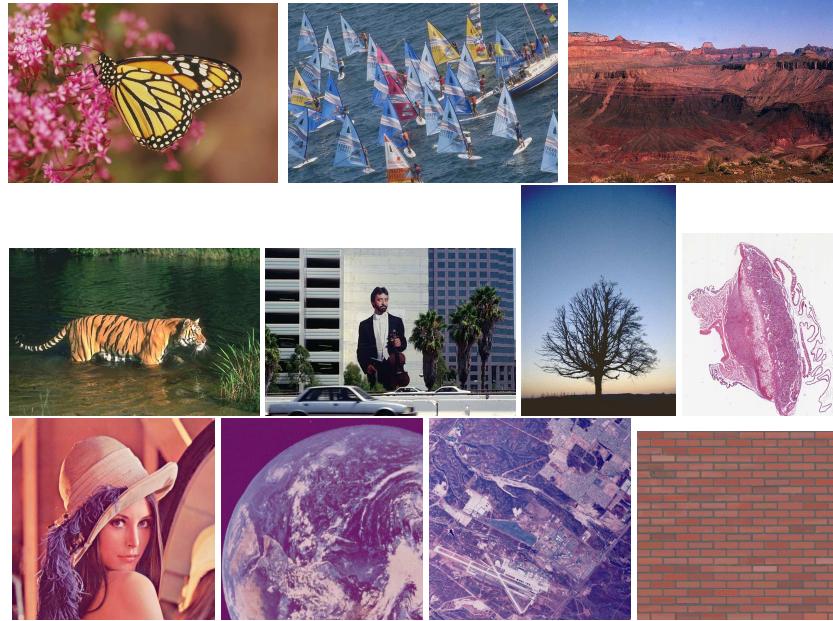


Figure 6.15. A few standard testing images. From the top-left to the bottom-right: monarch (768×512), sail (768×512), canyon (752×512), tiger (480×320), tree (512×768), tissue (microscopic) (1408×1664), Lena (512×512), earth (satellite) (512×512), urban (aerial) (512×512), bricks (696×648). The multi-scale hybrid linear model out-performs wavelets except for the Lena and the monarch.

6.2 Multi-Scale Hybrid Linear Models in Wavelet Domain

From the discussion in the previous section, we have noticed that wavelets can achieve a better representation for smooth regions and avoid the block artifacts. Therefore, in this section, we will combine the hybrid linear model with the wavelet approach to build multi-scale hybrid linear models in the wavelet domain. For readers who are not familiar with wavelets, we recommend the books of [?].

6.2.1 Imagery Data Vectors in Wavelet Domain

In the wavelet domain, an image is typically transformed into an octave tree of subbands by certain separable wavelet. At each level, the LH, HL, HH subbands contain the information about high frequency edges and the LL subband is further decomposed into subbands at the next level. Figure 6.16 shows the octave tree structure of a level-2 wavelet decomposition. As shown in the Figure 6.17, the vectors $\{x_i \in \mathbb{R}^D\}_{i=1}^M$ are constructed by stacking the corresponding wavelet

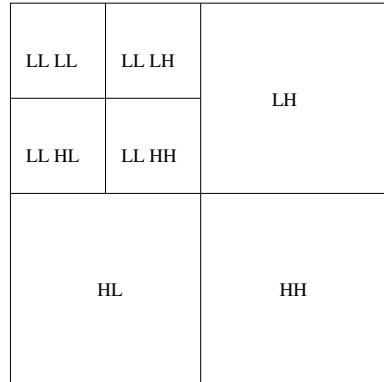


Figure 6.16. The subbands of a level-2 wavelet decomposition.

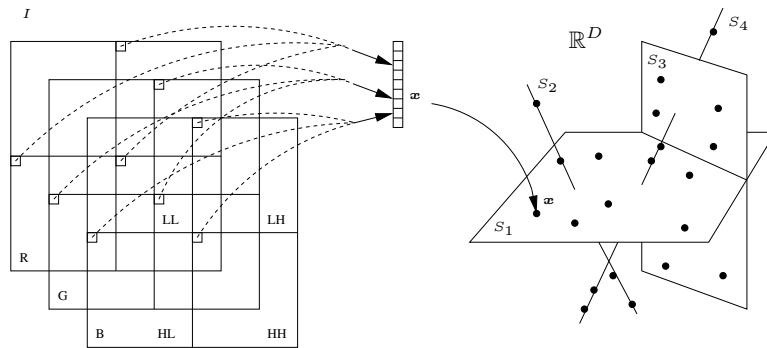


Figure 6.17. The construction of imagery data vectors in the wavelet domain. These data vectors are assumed to reside in multiple (affine) subspaces which may have different dimensions.

coefficients in the LH, HL, HH subbands. The dimension of the vectors is $D = 3c$ because there are c color channels. One of the reasons for this choice of vectors is because for edges along the same direction, these coefficients are linearly related and reside in a lower dimensional subspace. To see this, let us first assume that the color along an edge is constant. If the edge is along the horizontal, vertical or diagonal direction, there will be an edge in the coefficients in the LH, HL, or HH subband, respectively. The other two subbands will be zero. So the dimension of the imagery data vectors associated with such an edge will be 1. If the edge is not exactly in one of these three directions, there will be an edge in the coefficients of all the three subbands. For example, if the direction of the edge is between the horizontal and diagonal, the amplitude of the coefficients in the LH and HH subbands will be large. The coefficients in the HL subband will be insignificant relative to the coefficients in the other two subbands. So the dimension of the data vectors associated with this edge is approximately 2 (subject to a small error ϵ^2). If the color along an edge is changing, the dimension the subspace will be

higher but generally lower than the ordinal dimension $D = 3c$. Notice that the above scheme is only one of many possible ways in which one may construct the imagery data vector in the wavelet domain. For instance, one may construct the vector using coefficients across different scales. It remains an open question whether such new constructions may lead to even more efficient representations than the one presented here.

6.2.2 Estimation of Hybrid Linear Models in Wavelet Domain

In the wavelet domain, there is no need to build a down-sampling pyramid. The multi-level wavelet decomposition already gives a multi-scale structure in the wavelet domain. For example, Figure 6.18 shows the octave three structure of

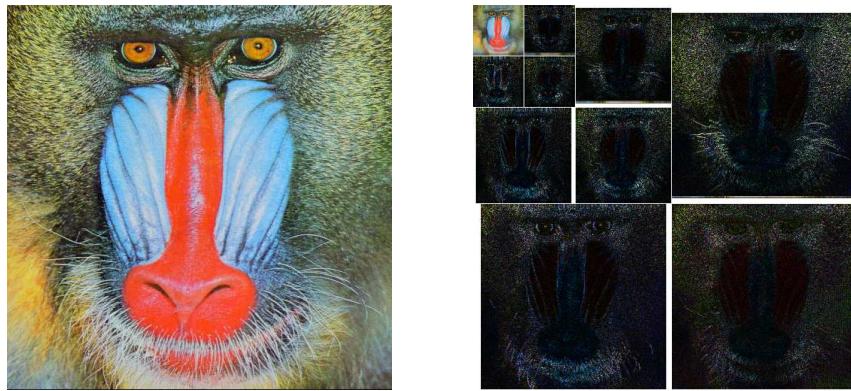


Figure 6.18. The subbands of level-3 bior-4.4 wavelet decomposition of the baboon image.

a level-3 bior-4.4 wavelet transformation of the baboon image. At each level, we may construct the imagery data vectors in the wavelet domain according to the previous section. A hybrid linear model will be identified for the so-obtained vectors at each level. Figure 6.19 shows the segmentation results using the hybrid linear model at three scale levels for the baboon image.

Vector Energy Constraint at Each Level. In the nonlinear wavelet approximation, the coefficients which are below an error threshold will be ignored. Similarly in our model, not all the vectors of the imagery data vectors need to be modeled and approximated. We only need to approximate the (coefficient) vectors $\{x_i\}$ that satisfy the following constraint:

$$\|x_i\|^2 > \epsilon^2. \quad (6.19)$$

Notice that here we do not need to scale the error tolerance at different levels because the wavelet basis is orthonormal by construction. In practice, the energy of most of the vectors is close to zero. Only a small portion of the vectors at each level need to be modeled (e.g. Figure 6.19).

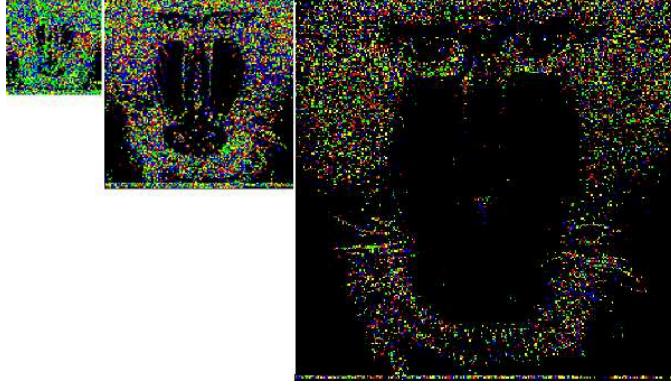


Figure 6.19. The segmentation of data vectors constructed from the three subbands at each level—different subspaces are denoted by different colors. The black regions correspond to data vectors whose energy is below the MSE threshold ϵ^2 in equation (6.19).

The overall process of estimating the multi-scale hybrid linear model in the wavelet domain can be summarized as the pseudocode in Algorithm 6.3.

Algorithm 6.3 (Multi-Scale Hybrid Linear Model: Wavelet Domain).

```

1: function  $\hat{\mathbf{I}} = \text{MultiscaleModel}(\mathbf{I}, \text{level}, \epsilon^2)$ 
2:    $\tilde{\mathbf{I}} = \text{WaveletTransform}(\mathbf{I}, \text{level});$ 
3:   for each  $\text{level}$  do
4:      $\hat{\tilde{\mathbf{I}}}_{\text{level}} = \text{HybridLinearModel}(\tilde{\mathbf{I}}_{\text{level}}, \epsilon^2);$ 
5:   end for
6:    $\hat{\mathbf{I}} = \text{InverseWaveletTransform}(\hat{\tilde{\mathbf{I}}}, \text{level});$ 
7:   return  $\hat{\mathbf{I}}.$ 

```

6.2.3 Comparison with Other Lossy Representations

In this section, in order to obtain a fair comparison, the experimental setting is the same as that of the spatial domain in the previous section. The experiment is conducted on the same two standard images – the 480×320 hill image and the 512×512 baboon image shown in Figure 6.10.

The number of levels of the model is also chosen to be 3. In Figure 6.20, the results are compared with several other commonly used image representations including DCT, PCA/KLT, single-scale hybrid linear model and Level-3 biorthogonal 4.4 wavelets (JPEG 2000) as well as the multi-scale hybrid linear model in the spatial domain. The multi-scale hybrid linear model in the wavelet domain achieves better PSNR than that in the spatial domain. Figure 6.21 shows the three recovered images using the same amount of coefficients for wavelets, the hybrid

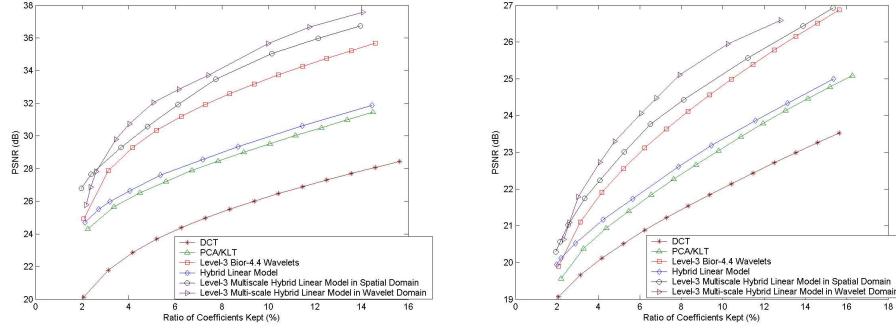


Figure 6.20. Top: Comparison of several image representations for the hill image. Bottom: Comparison for the baboon image. The multi-scale hybrid linear model in the wavelet domain achieves better PSNR than that in the spatial domain.

linear model in the spatial domain, and that in the wavelet domain, respectively. Figure 6.22 shows the visual comparison with the enlarged bottom-right corners of the images in Figure 6.21.

Notice that in the area around the baboon's whiskers, the wavelets blur both the whiskers and the subtle details in the background. The multi-scale hybrid linear model (in the spatial domain) preserves the sharp edges around the whiskers but generates slight block artifacts in the relatively smooth background area. The multi-scale hybrid linear model in the wavelet domain successfully eliminates the block artifacts, keeps the sharp edges around the whiskers, and preserves more details than the wavelets in the background. Among the three methods, the multi-scale hybrid linear model in the wavelet domain achieves not only the highest PSNR, but also produces the best visual effect.

As we know from the previous section, the multi-scale hybrid linear model in the spatial domain performs slightly worse than the wavelets for the Lena and monarch images (Figure 6.15). Nevertheless, in the wavelet domain, the multi-scale hybrid linear model can generate very competitive results, as shown in Figure 6.23. The multi-scale hybrid linear model in the wavelet domain achieves better PSNR than the wavelets for the monarch image. For the Lena image, the comparison is mixed and merits further investigation.

6.2.4 Limitations

The above hybrid linear model (in the wavelet domain) does not produce so competitive results for gray-scale images as the dimension of the vector is merely 3 and there is little room for further reduction. For gray-scale images, one may have to choose a slightly larger window in the wavelet domain or to construct the vector using wavelet coefficients across different scales. A thorough investigation of all the possible cases is beyond the scope of this book. The purpose here is to demonstrate (using arguably the simplest cases) the vast potential of a

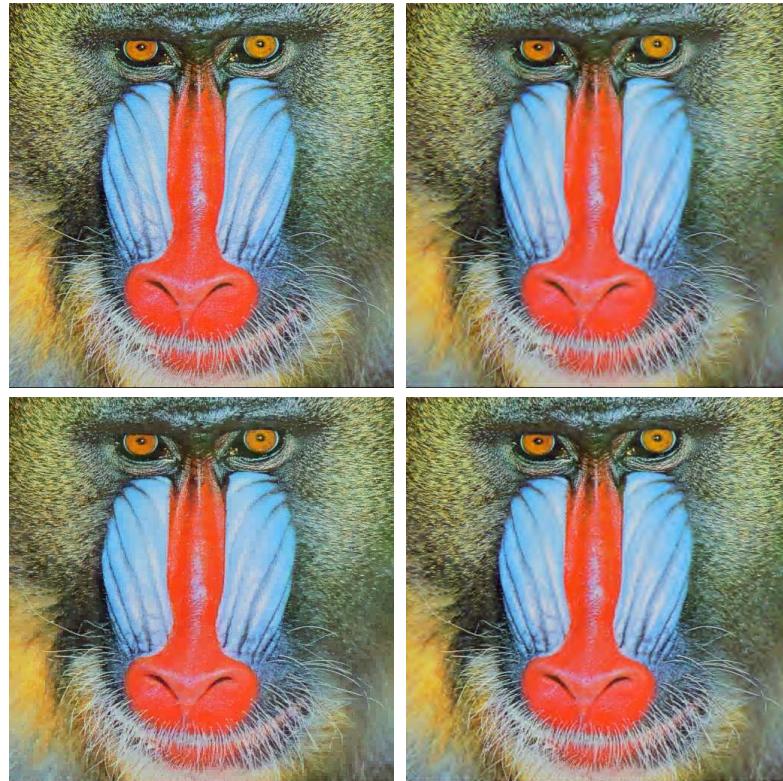


Figure 6.21. Visual comparison of three representations for the baboon image approximated with 7.5% coefficients. Top-left: The original image. Top-right: The level-3 biorthogonal 4.4 wavelets (PSNR=23.94). Bottom-left: The level-3 multi-scale hybrid linear model in the spatial domain (PSNR=24.64). Bottom-right: The level-3 multi-scale hybrid linear model in the wavelet domain (PSNR=24.88).

new spectrum of image representations suggested by combining subspace methods with conventional image representation/approximation schemes. The quest for the more efficient and more compact representations for natural images without doubt will continue as long as the nature of natural images remains a mystery and the mathematical models that we use to represent and approximate images improve.

6.3 Image Segmentation

6.3.1 Hybrid Linear Models for Image Segmentation

Notice that for the purpose of image representation, we normally divide the image I into *non-overlapping* blocks (see the beginning of Section 6.1.1). The hybrid

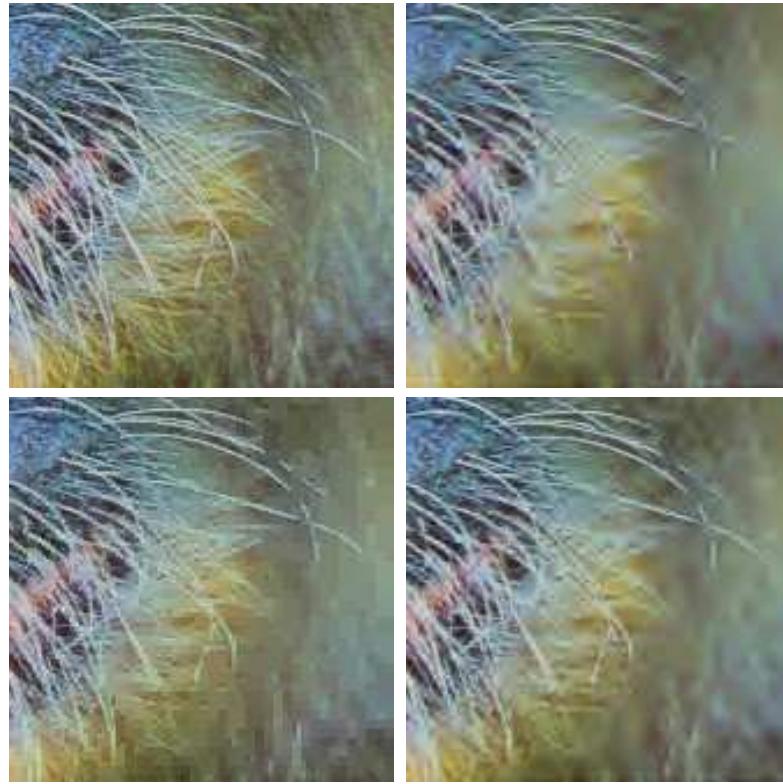


Figure 6.22. Enlarged bottom-right corner of the images in Figure 6.21. Top-left: The original image. Top-right: The level-3 biorthogonal 4.4 wavelets. Bottom-left: The level-3 multi-scale hybrid linear model in the spatial domain. Bottom-right: the level-3 multi-scale hybrid linear model in the wavelet domain.

linear model fit to the block vectors $\{\mathbf{x}_i\}$ essentially gives some kind of a segmentation of the image – pixels that belong to blocks in the same subspace are grouped into one segment. However, such a segmentation of the image has some undesirable features. If we choose a very large block size, then there will be severe “block effect” in the resulting segmentation, as all b^2 pixels of each block are always assigned into the same segment (see Figure 6.5). If we choose a small block size to reduce the block effect, then the block might not contain sufficient neighboring pixels that allow us to reliably extract the local texture.¹⁴ Thus, the resulting segmentation will very much be determined primarily by the color of the pixels (in each small block) but not the texture.

One way to resolve the above problems is to choose a block of a reasonable size around each pixel and view the block as a (vector-valued) “label” or “feature”

¹⁴Notice that a smaller block size is ok for compression as long as it can reduce the overhead and subsequently improve the overall compression ratio.

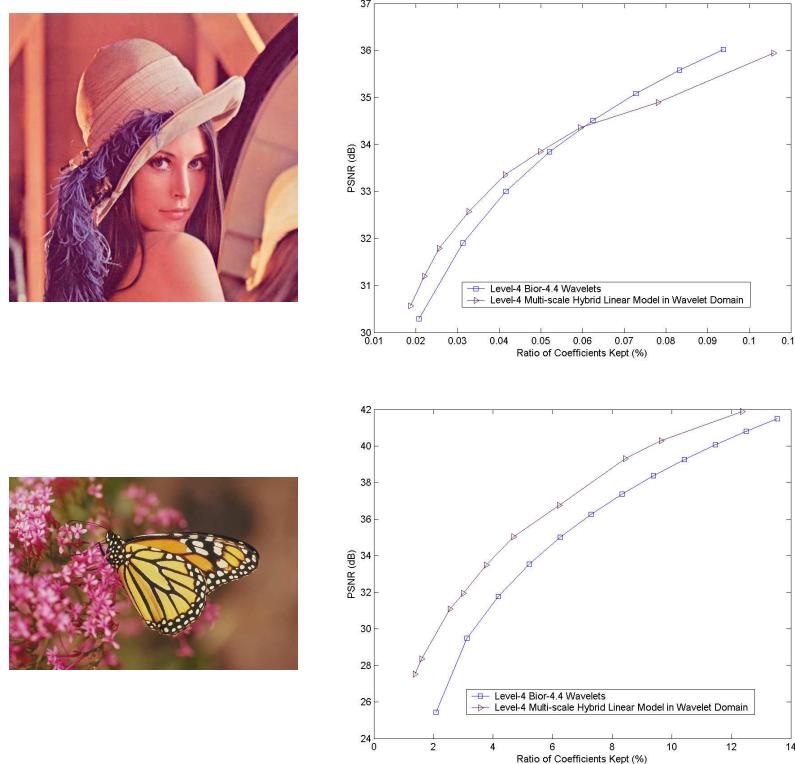


Figure 6.23. Top: Comparison of multi-scale hybrid linear model in the wavelet domain with wavelets for the Lena image. Bottom: Comparison of multi-scale hybrid linear model in the wavelet domain with wavelets for the Monarch image. The multi-scale hybrid linear model in the wavelet domain achieves better PSNR than wavelets for a wide range of PSNR for these two images.

attached to the pixel. In many existing image segmentation methods, the feature (vector) is chosen instead to be the outputs of the block passing through a (pre-fixed) bank of filters (e.g., the Garbor filters). That is, the feature is the block transformed by a set of pre-fixed linear transformations. Subsequently, the image is segmented by grouping pixels that have “similar” features.

From the lessons that we have learned from image representation in the previous section, we observe that the hybrid linear model may be adopted to facilitate this approach of image segmentation in several aspects. Firstly, we can fit directly a hybrid linear model to the un-transformed and un-processed block vectors, without the need of choosing beforehand which filter bank to use. Secondly, the hybrid linear model essentially chooses the linear transformations (or filters) adaptively for different images and different image segments. Thirdly, once the hybrid linear model is identified, there is no further need of introducing a similarity measure for

the features. Feature vectors (and hence pixels) that belong to the same subspace are naturally grouped into the same image segment.

6.3.2 Dimension and Size Reduction

Mathematically, identifying such a hybrid linear model for image segmentation is equivalently to that for image representation. However, two things have changed from image representation and make image segmentation a computationally much more challenging problem. First, the number of feature vectors (or blocks) is now always the same as the number of pixels: $N = WH$, which is larger than that ($N = WH/b^2$) in the case of image representation. For a typical 512×512 image, we have $N = 31,744$. Second, the block size b now can be much larger than in the case of image representation. Thus, the dimension of the block vector $D = b^2c$ is much higher. For instance, if we choose $b = 10$ and $c = 3$, then $D = 300$. It is impossible to implement the GPCA algorithm on a regular PC for 31,744 vectors in \mathbb{R}^{300} , even if we are looking for up to only four or five subspaces.¹⁵

Dimension Reduction via Projection.

To reduce dimension of the data, we rely on the assumption (or belief) that “the feature vectors lie on very low-dimensional subspaces in the high-dimensional ambient space \mathbb{R}^D .” Then based on our discussion in Section 3.2.2, we can project the data into a lower-dimensional space while still being able to preserve the separation of the subspaces. Principal component analysis (PCA) can be recruited for this purpose as the energy of the feature vectors is mostly preserved by their first few principal components. From our experience, in practice it typically suffice to keep up to ten principal components. Symbolically, the process is represented by the following steps:

$$\{\mathbf{x}_i\} \subset \mathbb{R}^D \xrightarrow{\mathbf{x}'_i=\pi(\mathbf{x}_i)} \{\mathbf{x}'_i\} \subset \mathbb{R}^{D'} \xrightarrow{\text{GPCA}} \{\mathbf{x}'_i\} \subset \bigcup_{j=1}^n S'_j \subset \mathbb{R}^{D'},$$

where $D' \ll D$ and $i = 1, \dots, N = WH$.

Data Size Reduction via Down-Sampling.

Notice that the number of feature vectors $N = WH$ might be too large for all the data to be processed together since a regular PC has trouble in performing singular value decomposition (SVD) for tens of thousands of vectors.¹⁶ Thus, we have to down sample the data set and identify a hybrid linear model for only a subset of the data. The exact down-sampling scheme can be determined by the user. One can use periodic down-sampling (e.g., every other pixel) or random down-sampling. From our experience, we found periodic down-sampling often

¹⁵The dimension of the Veronese embedding of degree 5 will be in the order of 10^{10} .

¹⁶With the increase of memory and speed of modern computers, we hope this step will soon become unnecessary.

gives visually better segmentation results. The size of the down-sampled subset can be determined by the memory and speed of the computer the user has. Once the hybrid linear model is obtained, we may assign the remaining vectors to their closest subspaces. Of course, in practice, one may run the process on multiple subsets of the data and choose the one which gives the smallest fitting error for all the data. This is very much in the same spirit as the random sample consensus (RANSAC) method. Symbolically, the process is represented by the following steps:

$$\{\mathbf{x}_i\} \xrightarrow{\text{sample}} \{\mathbf{x}_{i'}\} \xrightarrow{\text{GPCA}} \{\mathbf{x}_{i'}\} \subset \bigcup_{j=1}^n S_j \xrightarrow{\min d(\mathbf{x}_i, S_j)} \{\mathbf{x}_i\} \subset \bigcup_{j=1}^n S_j,$$

where $\{\mathbf{x}_{i'}\}$ is a (down-sampled) subset of $\{\mathbf{x}_i\}$.

6.3.3 Experiments

Figure 6.24 shows the results of applying the above schemes to the segmentation of some images from the Berkeley image database. A $20 \times 20 \times 3$ “feature” vector is associated with each pixel that corresponds to the color values in a 20×20 block. We first apply PCA to project all the feature vectors onto a 6-dimensional subspace. We then apply the GPCA algorithm to further identify subspace-structures of the features in this 6-dimensional space and to segment the pixels to each subspace. The algorithm happens to find three segments for all the images shown below. Different choices in the error tolerance, window size, and color space (HSV or RGB) may affect the segmentation results. Empirically, we find that HSV gives visually better segments for most images.

6.4 Image Classification

Notice that images of similar scenes (e.g., landscape, or urban areas, or wild life, etc.) often contain similar textures and characteristics. It is then interesting to see what would be the underlying hybrid linear model for images of each category.

For the example below, we select two groups of eight gray-scale images from the Berkeley segmentation data set [Martin et al., 2001] shown in Figure 6.25. One group contains common natural scenes, and the other contains more structured urban scenes. We randomly sample 100 blocks of 8 by 8 windows from each image,¹⁷ stack them into 64-dimensional vectors, and apply our algorithm to obtain a hybrid linear model.

For this example, we preset the desired subspace number as 3, and the algorithm identifies the bases for the subspaces for each group. Figure 6.26 shows all base vectors as 8 by 8 windows. Visually, one can see that these vectors capture the essential difference between natural scenes and urban scenes.

¹⁷The window size and the number of blocks is limited by the computer memory.

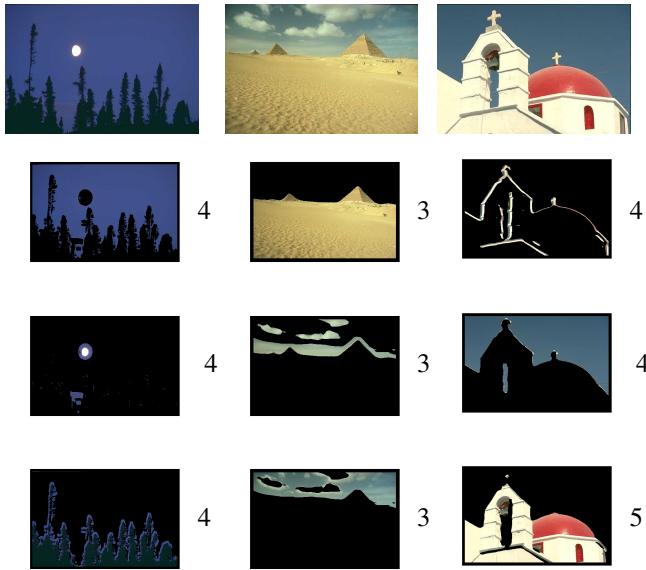


Figure 6.24. Image segmentation results obtained from the hybrid linear model. The dimension of the subspace (in homogeneous coordinates) associated with each segment is marked by the number to its right.



Figure 6.25. Top: natural scenes. Down: Urban scenes.

Unlike most of our other examples, such hybrid linear models are “learned” in a *supervised* fashion as the training images are classified by humans. The so-obtained hybrid linear models can potentially be used to classify new images of natural or urban scenes into one of the two categories.

In principle, we can also treat each image as a single point (in a very high-dimensional space) and fit a hybrid linear model to an ensemble of images. Then different subspaces of the model lead to a segmentation of the images into different categories. Images in the same category form a single linear subspace. We have seen such an example for face images in Chapter 1 (Figure 1.6).

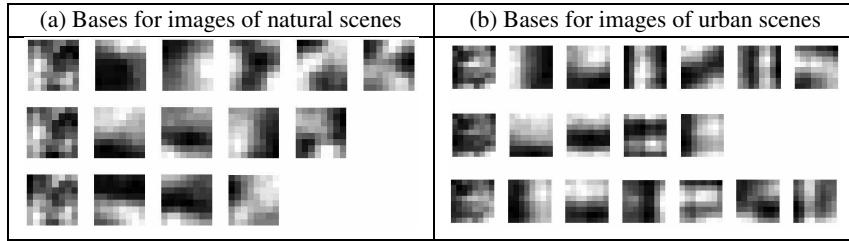


Figure 6.26. Base vectors of the identified subspaces for each category of images.

6.5 Bibliographic Notes

Image Representation and Compression.

There is a vast amount of literature on finding adaptive bases (or transforms) for signals. Adaptive wavelet transform and adapted wavelet packets have been extensively studied [?, ?, ?, ?, ?, ?, ?]. The idea is to search for an optimal transform from a limited (although large) set of possible transforms. Another approach is to find some universal optimal transform based on the signals [?, ?, ?, ?]. Spatially adapted bases have also been developed such as [?, ?, ?].

The notion of hybrid linear model for image representation is also closely related to the *sparse component analysis*. In [?], the authors have identified a set of non-orthogonal base vectors for natural images such that the representation of the image is sparse (i.e., only a few components are needed to represent each image block). In the work of [?, ?, ?, ?, ?, ?, ?, ?, ?], the main goal is to find a mixture of models such that the signals can be decomposed into multiple models and the overall representation of the signals is sparse.

Image Segmentation.

Image segmentation based on local color and texture information extracted from various filter banks has been studied extensively in the computer vision literature (see e.g., [?, ?, ?]). In this chapter, we directly used the unfiltered pixel values of the image. Our segmentation is a byproduct of the global fitting of a hybrid linear model for the entire image. Since the image compression standard JPEG-2000 and the video compression standard MPEG-4 have started to incorporate texture segmentation [?], we expect that the method introduced in this chapter will be useful for developing new image processing techniques that can be beneficial to these new standards.

Chapter 7

2-D Motion Segmentation from Image Partial Derivatives

In the previous chapter, we studied how to use hybrid (linear) models to represent (static) images. The multiple (linear) components of a hybrid model were used to effectively account for the multi-modal characteristics of natural images, e.g., different textures. GPCA offers an efficient solution to the estimation of such a model from the image. In the next chapters, we will show how to apply the same techniques to the study of videos and in particular, how to model and extract multi-modal characteristics of the dynamics of a video sequence.

To understand the dynamics of a video sequence, we need to model how every pixel moves from one image to the next. One of the fundamental difficulties in modeling the motion of all the pixels is that pixels in different regions of the image may move differently from one region to another (which we will articulate soon). Therefore, quantitatively, we may need multiple parametric models to describe the motions of different regions. Sometimes the models needed for different regions may even belong to different classes of models. The problem of determining which models to use for different motions is further compounded with the fact that we normally do not know which pixels correspond to which motion model.

This is a challenging problem in the study of any dynamical visual data, whose mathematical nature depends largely on the type of image measurements (image derivatives, optical flows, point correspondences), camera models (orthographic, spherical, perspective), and motion models (translational, rigid, affine) that one adopts to describe such measurements.

Depending on whether one is interested in understanding the motion in the 2-D image, or the motion in 3-D space, 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 (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 motions 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.

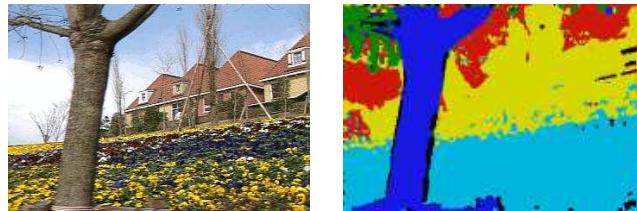


Figure 7.1. 2-D motion segmentation of the *flower-garden sequence* from <http://www-bcs.mit.edu/people/jyawang/demos>.

7.1 An Algebraic Approach to Motion Segmentation

In the next two chapters, we present an algebraic framework for segmenting 2-D motion models from spatial-temporal image derivatives (Chapter 7) and 3-D motion models from point correspondences (Chapter 8). The key to this algebraic approach is to view the estimation of multiple motion models as the estimation of a *single*, though more complex, *multibody motion model* that is then factored into the original models. This is achieved by (1) algebraically eliminating the data segmentation problem, (2) fitting a single multibody motion model to all the image measurements, and (3) segmenting the multibody motion model into its individual components. More specifically, the approach proceeds as follows:

1. *Eliminate Data Segmentation:* Find algebraic equations that are satisfied by all the image measurements, regardless of the motion model associated with each measurement. For instance, if the i th motion model is defined by an algebraic equation of the form $f_j(\mathbf{x}, \mathcal{M}_j) = 0$, where \mathbf{x} is a measurement,

\mathcal{M}_j represents the parameters of the j th motion model and $j = 1, \dots, n$, then an algebraic equation that is satisfied by all the data is

$$g(\mathbf{x}, \mathcal{M}) = f_1(\mathbf{x}, \mathcal{M}_1) \cdot f_2(\mathbf{x}, \mathcal{M}_2) \cdots f_n(\mathbf{x}, \mathcal{M}_n) = 0. \quad (7.1)$$

This equation represents a single *multibody motion model* whose parameters \mathcal{M} encode those of the original motion models $\{\mathcal{M}_j\}_{j=1}^n$.

2. *Multibody Motion Estimation:* Estimate the parameters \mathcal{M} of the multibody motion model from the given image measurements. For the motion models we will consider, \mathcal{M} will correspond to the coefficients of a polynomial p_n of degree n . We will show that the number of motions n and the coefficients \mathcal{M} can be obtained *linearly* after properly embedding the image measurements into a higher-dimensional space.
3. *Motion Segmentation:* Recover the parameters of the original motion models from the parameters of the multibody motion model \mathcal{M} , that is,

$$\mathcal{M} \rightarrow \{\mathcal{M}_j\}_{j=1}^n. \quad (7.2)$$

We will show that the type of motion model can be determined from the rank of a matrix formed from the partial *derivatives* of p_n , while the individual motion parameters \mathcal{M}_j can be computed from the first and second *derivatives* of p_n evaluated at a collection of n image measurements which can be obtained automatically from the data.

In this chapter, we concentrate on the problem of segmenting 2-D translational and 2-D affine motion models directly from the spatial-temporal image derivatives. In the case of 2-D translational motions, the resulting motion models are linear, hence the above framework reduces to GPCA. In the case of 2-D affine motions the resulting motion models are bilinear, hence the above framework is a particular case of QSA. In the case in which both 2-D translational and 2-D affine motion models are present, we develop new techniques that combine GPCA and QSA to simultaneously recover the type of motion, the motion parameters, and the segmentation of the image measurements.

7.2 The Multibody Brightness Constancy Constraint

Consider a motion sequence taken by a moving camera observing an *unknown* number m of independently and rigidly moving objects. The 3-D motion of each object relative to the camera induces a 2-D motion field in the image plane called *optical flow*. Because of perspective effects, depth discontinuities, occlusions, transparent motions, etc., each 3-D motion induces one or more 2-D motions. Therefore, we assume that the optical flow of the scene is generated from an

unknown number $n \geq m$ of 2-D motion models $\{\mathcal{M}_j\}_{j=1}^n$ of the form

$$\mathbf{u}(\mathbf{x}) = \begin{cases} \mathbf{u}_1(\mathbf{x}) & \text{if } \mathbf{x} \in \mathcal{R}_1, \\ \mathbf{u}_2(\mathbf{x}) & \text{if } \mathbf{x} \in \mathcal{R}_2, \\ & \vdots \\ \mathbf{u}_n(\mathbf{x}) & \text{if } \mathbf{x} \in \mathcal{R}_n, \end{cases} \quad (7.3)$$

where $\mathbf{u}(\mathbf{x}) = [\mathbf{u}, \mathbf{v}, 1]^\top \in \mathbb{P}^2$ is the optical flow of pixel $\mathbf{x} = [x_1, x_2, 1]^\top \in \mathbb{P}^2$, and $\mathcal{R}_j \subset \mathbb{P}^2$ is the region of the image where the j th motion model holds.

Assume now that each one of the surfaces in the scene is Lambertian. That is, if $I(\mathbf{x}(t), t)$ is the brightness of pixel \mathbf{x} at time t , then $I(\mathbf{x}(t), t) = I(\mathbf{x}(t_0), t_0)$. In other words, the brightness of any 3-D point in any of the surfaces does not change as the surfaces and the camera move. Then, the optical flow $\mathbf{u}_j(\mathbf{x})$ at $\mathbf{x} \in \mathcal{R}_j$ can be related to the spatial-temporal image derivatives $\mathbf{y}(\mathbf{x}) = [I_{x_1}, I_{x_2}, I_t]^\top \in \mathbb{R}^3$ at \mathbf{x} by the well-known *brightness constancy constraint* (BCC) [Ma et al., 2003]

$$\text{BCC}(\mathbf{x}, \mathbf{y}) = \mathbf{y}^\top \mathbf{u}_j(\mathbf{x}) = I_{x_1} \mathbf{u} + I_{x_2} \mathbf{v} + I_t = 0. \quad (7.4)$$

Thus, if (\mathbf{x}, \mathbf{y}) is an image measurement associated with any of the n motion models, then there exists a motion model \mathcal{M}_j whose optical flow $\mathbf{u}_j(\mathbf{x})$ is such that $\mathbf{y}^\top \mathbf{u}_j(\mathbf{x}) = 0$. Therefore, the following *multibody brightness constancy constraint* (MBCC) must be satisfied by every pixel in the image

$$\text{MBCC}(\mathbf{x}, \mathbf{y}) = \prod_{j=1}^n (\mathbf{y}^\top \mathbf{u}_j(\mathbf{x})) = 0. \quad (7.5)$$

Notice that the MBCC clearly resembles the mathematical nature of the product polynomial (4.16) introduced in Section 4.1.3 for segmenting hyperplanes. In fact, if the optical flow $\mathbf{u}(\mathbf{x})$ is a piecewise constant function of \mathbf{x} , meaning that the optical flow of the i th region does not depend the coordinates of \mathbf{x} , i.e., $\mathbf{u}_j(\mathbf{x}) = \mathbf{u}_j$, then according to (7.5) the image partial derivatives $\mathbf{y} \in \mathbb{R}^3$ can be seen as data points lying on a union of planes in \mathbb{R}^3 , $\mathcal{H}_j = \{\mathbf{y} \in \mathbb{R}^3 : \mathbf{y}^\top \mathbf{u}_j = 0\}$, with normal vectors $\{\mathbf{u}_j\}_{j=1}^n$. Furthermore, if the optical flow $\mathbf{u}(\mathbf{x})$ is a piecewise affine function of \mathbf{x} , meaning that $\mathbf{u}_j(\mathbf{x}) = A_j \mathbf{x}$, where $A_j \in \mathbb{R}^{3 \times 3}$, then it follows from (7.5) that the data (\mathbf{x}, \mathbf{y}) lives in the union of n quadratic surfaces. These two observations establish a clear connection between the segmentation of 2-D translational models and hyperplane segmentation, and between the segmentation of 2-D affine motion models and the segmentation of quadratic surfaces.

In the following sections, we explore these connections further and show that the MBBC allows us to solve the 2-D motion segmentation problem in closed form for the class of 2-D translational and 2-D affine motion models. More specifically, we will consider the following problem:

For the sake of simplicity, we divide our analysis into three parts. In Section 7.3 we consider the particular case of purely translational motions, i.e., $n = n_t$ and $n_a = 0$, and show that Problem 7.1 can be solved as a direct application of GPCA. In Section 7.4 we consider the particular case of purely affine motion models, i.e.,

Problem 7.1 (2-D Motion Segmentation from Image Partial Derivatives)

Given the spatial-temporal partial derivatives $\{(I_{x_1}^j, I_{x_2}^j, I_t^j)\}_{j=1}^N$ of a motion sequence generated from n_t 2-D translational and n_a 2-D affine motion models $\{\mathcal{M}_j\}_{j=1}^{n_a+n_t}$, estimate the number of motion models (n_a, n_t) , the optical flow $\mathbf{u}(\mathbf{x})$ at each pixel \mathbf{x} , the type of motion model at each pixel, and the motion parameters of the $n = n_a + n_t$ models, without knowing which image measurements correspond to which motion model.

$n = n_a$ and $n_t = 0$, and show that Problem 7.1 can be solved by considering the first order partial derivatives of the MBCC. In Section 7.5 we consider the most general case of both 2-D translational and 2-D affine motion models, , and show that Problem 7.1 can be solved by considering both first and second order partial derivatives of the MBCC.

Before proceeding further, we state a general property of the MBCC that will be used extensively in the remainder of the chapter.

Theorem 7.1 (Computing Optical Flow from the MBCC). *Let (\mathbf{x}, \mathbf{y}) be an image measurement associated to only one of the $n = n_a + n_t$ motion models, i.e., there is a unique $k = 1, \dots, n$ such that $\mathbf{y}^\top \mathbf{u}_k(\mathbf{x}) = 0$. Then the optical flow of a pixel \mathbf{x} is given by*

$$\mathbf{u}(\mathbf{x}) = \frac{\partial \text{MBCC}(\mathbf{x}, \mathbf{y})}{\partial \mathbf{y}} / \left(e_3^\top \frac{\partial \text{MBCC}(\mathbf{x}, \mathbf{y})}{\partial \mathbf{y}} \right), \quad (7.6)$$

where $e_3 = [0, 0, 1]^\top$.

Proof. The partial derivative of the MBCC is given by

$$\frac{\partial \text{MBCC}(\mathbf{x}, \mathbf{y})}{\partial \mathbf{y}} = \sum_{j=1}^n \mathbf{u}_j(\mathbf{x}) \prod_{\ell \neq i} (\mathbf{y}^\top \mathbf{u}_\ell(\mathbf{x})). \quad (7.7)$$

Since there is a unique $k = 1, \dots, n$ such that $\mathbf{y}^\top \mathbf{u}_k(\mathbf{x}) = 0$, we have that $\prod_{\ell \neq i} (\mathbf{y}^\top \mathbf{u}_\ell(\mathbf{x})) = 0$ for all $i \neq k$. After replacing this in (7.7), we obtain that the partial derivative of the MBCC with respect to \mathbf{y} is proportional to $\mathbf{u}_k(\mathbf{x})$. Since the last entry of $\mathbf{u}(\mathbf{x})$ is one, the result follows. \square

The importance of Theorem 7.1 is that it allows us to estimate the optical flow at a pixel \mathbf{x} *without* knowing either the type or the parameters of the motion model associated with \mathbf{x} . This offers an important advantage over existing methods for computing optical flow (see Section 7.8 for a brief review) which use the single body BCC on a local neighborhood of each pixel.

Remark 7.2. Notice that there are some pixels for which the optical flow cannot be computed as stated in Theorem 7.1. If a pixel \mathbf{x} is such that its image measurement \mathbf{y} happens to satisfy $\mathbf{y}^\top \mathbf{u}_j(\mathbf{x}) = \mathbf{y}^\top \mathbf{u}_k(\mathbf{x}) = 0$ for $j \neq k$, then the MBCC has a repeated factor, hence its derivative is zero, and we cannot compute $\mathbf{u}(\mathbf{x})$ as before. These cases occur when the image measurements satisfy certain polynomial equations, thus they violate the non-degeneracy assumption.

7.3 Segmentation of 2-D Translational Motion Models

7.3.1 The Multibody Optical Flow

For the sake of simplicity, let us first assume that the 2-D motion of the scene is generated only from 2-D translational motions, i.e., $n = n_t$ and $n_a = 0$. In this case, the optical flow $\mathbf{u}(\mathbf{x}) = [\mathbf{u}, \mathbf{v}, 1]^\top \in \mathbb{P}^2$ at pixel $\mathbf{x} = [x_1, x_2, 1]^\top \in \mathbb{P}^2$ can be described by the equations

$$\mathbf{u}(x_1, x_2) = u_1, \quad (7.8)$$

$$\mathbf{v}(x_1, x_2) = u_2, \quad (7.9)$$

where u_1 and u_2 are the so-called *translational motion parameters*. Since the optical flow of the i th region, $\mathbf{u}(\mathbf{x}) = \mathbf{u}_j$, does not depend on the pixel coordinates \mathbf{x} , the MBCC takes the form of a homogeneous polynomial of degree n in \mathbf{y}

$$\text{MBCC}(\mathbf{x}, \mathbf{y}) = \prod_{j=1}^n (\mathbf{y}^\top \mathbf{u}_j) = \nu_n(\mathbf{y})^\top \mathcal{U} = 0, \quad (7.10)$$

where $\nu_n : \mathbb{R}^3 \rightarrow \mathbb{R}^{M_n(3)}$ is the Veronese map of degree n defined in Section 4.2.2 and the *multibody optical flow* $\mathcal{U} \in \mathbb{R}^{M_n(3)}$ is the vector of coefficients of the MBCC.

7.3.2 Computing the 2-D Translational Model Parameters

Thanks to the MBCC, the problem of segmenting 2-D translational motions is equivalent to the hyperplane segmentation problem described in Section 4.1.3. The points on the hyperplanes are the image partial derivatives $\mathbf{y} \in \mathbb{R}^3$ and the normal vectors to the hyperplanes are the optical flows $\{\mathbf{u}_j\}_{j=1}^n$. Therefore, we can estimate the number of 2-D translational models and the parameters of the 2-D translational models by using the GPCA algorithm for hyperplanes (Algorithm 4.3), with minor modifications to account for the fact that the third entry of each \mathbf{u}_j is one. This leads to the following solution to Problem 7.1 in the case of 2-D translational motions.

- Given $N \geq M_n(3) - 1$ image measurements $\{\mathbf{y}_j\}_{j=1}^N$ in general configuration on the n hyperplanes, one can write the MBCC for all the measurements as the following relationship between the number of motions n and the multibody optical flow \mathcal{U}

$$\mathbf{V}_n^{\mathcal{U}} \mathcal{U} \doteq [\nu_n(\mathbf{y}_1) \ \nu_n(\mathbf{y}_2) \ \cdots \ \nu_n(\mathbf{y}_N)]^\top \mathcal{U} = 0. \quad (7.11)$$

From this relationship, one can compute the number of motions as

$$n = \min\{j : \text{rank}(\mathbf{V}_j^{\mathcal{U}}) = M_j(3) - 1\} \quad (7.12)$$

where $\mathbf{V}_j^{\mathcal{U}} \in \mathbb{R}^{N \times M_j(3)}$ is the matrix in (7.11), but computed with the Veronese map ν_j of degree $i \geq 1$. Given n , one can solve for \mathcal{U} uniquely

from the linear system (7.11) by enforcing the last entry of \mathcal{U} to be one. This additional equation results from the fact that the last entry of each \mathbf{u}_j is one.

2. Given \mathcal{U} and n , as per Theorem 7.1 one can compute the optical flow $\mathbf{u}(\mathbf{x})$ at each pixel \mathbf{x} without knowing which motion model is associated with each pixel from the partial derivative of the MBCC with respect to \mathbf{y} as

$$\mathbf{u}(\mathbf{x}) = \frac{\partial \text{MBCC}(\mathbf{x}, \mathbf{y})}{\partial \mathbf{y}} / \left(e_3^\top \frac{\partial \text{MBCC}(\mathbf{x}, \mathbf{y})}{\partial \mathbf{y}} \right). \quad (7.13)$$

3. Solve for the n 2-D translational motion models $\{\mathbf{u}_j\}_{j=1}^n$ by choosing one pixel per motion model $\{\mathbf{x}_j\}_{j=1}^n$ and then setting $\mathbf{u}_j = \mathbf{u}(\mathbf{x}_j)$, where $\mathbf{u}(\mathbf{x})$ is computed as in (7.6). Pixel \mathbf{x}_j for $j = n, n-1, \dots, 1$ is chosen so that $\mathbf{y}(\mathbf{x}_j)$ has a minimizes the distance d_j to the i th motion model, where

$$d_n^2(\mathbf{x}, \mathbf{y}) = \frac{|\text{MBCC}(\mathbf{x}, \mathbf{y})|^2}{\left\| \frac{\partial \text{MBCC}(\mathbf{x}, \mathbf{y})}{\partial \mathbf{y}} \right\|^2} \quad \text{and} \quad d_{j-1}^2(\mathbf{x}, \mathbf{y}) = \frac{d_j^2(\mathbf{x}, \mathbf{y})}{\frac{|\mathbf{y}^\top \mathbf{u}(\mathbf{x}_j)|^2}{\|\mathbf{u}(\mathbf{x}_j)\|^2}}. \quad (7.14)$$

Recall that for any surface $f(\mathbf{y}) = 0$, a first order approximation to the geometric distance to the surface is given by $|f(\mathbf{y})|/\|\nabla f(\mathbf{y})\|$. Notice that in choosing the points there is no optimization involved. We just need to evaluate the distance functions at each pixel and choose the one giving the minimum distance. Notice also that the distance functions are slightly different from the ones in Chapter 4, because we must enforce the additional constraint that the last entry of each \mathbf{u}_j is one.

7.4 Segmentation of 2-D Affine Motion Models

Assume now that the 2-D motion of the scene is generated only from 2-D affine motions, i.e., $n = n_a$ and $n_t = 0$. In this case, the optical flow $\mathbf{u}(\mathbf{x}) = [u, v, 1]^\top \in \mathbb{P}^2$ at pixel $\mathbf{x} = [x_1, x_2, 1]^\top \in \mathbb{P}^2$ can be described by the equations

$$u(x_1, x_2) = a_{11}x_1 + a_{12}x_2 + a_{13}, \quad (7.15)$$

$$v(x_1, x_2) = a_{21}x_1 + a_{22}x_2 + a_{23}, \quad (7.16)$$

where a_{11}, \dots, a_{23} are the so-called *affine motion parameters*. By combining equations (7.15) and (7.16) we see that the BCC for the affine model case takes the following form

$$\mathbf{y}^\top A\mathbf{x} = \begin{bmatrix} I_{x_1} & I_{x_2} & I_t \end{bmatrix} \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ 1 \end{bmatrix} = 0, \quad (7.17)$$

We shall refer to the above constraint as the *affine constraint*.

Remark 7.3. Notice that when $a_{11} = a_{21} = a_{12} = a_{22} = 0$, the affine motion model reduces to the translational motion model $\mathbf{u} = [a_{13}, a_{23}, 1]^\top$ discussed

in the previous subsection, and the affine constraint $\mathbf{y}^\top A\mathbf{x} = 0$ reduces to the brightness constancy constraint $\mathbf{y}^\top \mathbf{u} = 0$.

Unlike the 2-D translational model, in the case of 2-D affine motions the optical flow associated with the i th region, $\mathbf{u}_i(\mathbf{x}) = A_i\mathbf{x}$, does depend on the pixel coordinates. Therefore, the MBCC (7.5) takes the form of a bi-homogeneous polynomial of degree n in (\mathbf{x}, \mathbf{y})

$$\text{MBCC}(\mathbf{x}, \mathbf{y}) \doteq \prod_{j=1}^n (\mathbf{y}^\top A_j \mathbf{x}) = 0, \quad (7.18)$$

i.e., a homogeneous polynomial of degree n in either \mathbf{x} or \mathbf{y} . We call this constraint the *multibody affine constraint*, since it is a natural generalization of the affine constraint valid for $n = 1$.

7.4.1 The Multibody Affine Matrix

The multibody affine constraint converts Problem 7.1 into one of solving for the number of affine motions n and the affine motion parameters $\{A_j\}_{j=1}^n$ from the polynomial equation (7.18). In principle, this problem is equivalent to segmenting data (\mathbf{x}, \mathbf{y}) living on a collection of n quadratic surfaces, thus we could solve the 2-D affine motion segmentation problem using the QSA algorithm described in Chapter 12. However, note that $\mathbf{y}^\top A_j \mathbf{x} = 0$ is not a general quadratic surface, but rather a bilinear surface. Therefore, we propose to exploit this additional structure in order to obtain a motion segmentation algorithm which is simpler and more efficient than QSA.

To this end, notice that if we expand the polynomial in (7.18) as a sum of all the possible monomials of degree n in \mathbf{x} and \mathbf{y} , then we can write the MBCC as a bilinear expression in $(\nu_n(\mathbf{x}), \nu_n(\mathbf{y}))$ as stated by the following result.

Theorem 7.4 (The Bilinear Multibody Affine Constraint). *The multibody affine constraint (7.18) can be written in bilinear form as*

$$\nu_n(\mathbf{y})^\top \mathcal{A} \nu_n(\mathbf{x}) = 0, \quad (7.19)$$

where each entry of $\mathcal{A} \in \mathbb{R}^{M_n(3) \times M_n(3)}$ is multilinear on the matrices $\{A_j\}_{j=1}^n$.

Proof. Let $\mathbf{u}_j = A_j \mathbf{x} \in \mathbb{R}^3$, for $j = 1, \dots, n$. Then, the multibody affine constraint is a homogeneous polynomial of degree n in $\mathbf{y} = [y_1, y_2, y_3]^\top$, i.e.,

$$\text{MBCC}(\mathbf{x}, \mathbf{y}) = \prod_{j=1}^n (\mathbf{y}^\top \mathbf{u}_j) = \sum c_{n_1, n_2, n_3} y_1^{n_1} y_2^{n_2} y_3^{n_3} = \nu_n(\mathbf{y})^\top \mathbf{c}_n,$$

where $\mathbf{c}_n \in \mathbb{R}^{M_n(3)}$ is the vector of coefficients. From the properties of polynomial multiplication, each entry of \mathbf{c}_n , c_{n_1, n_2, n_3} , must be a symmetric multilinear function of $(\mathbf{u}_1, \dots, \mathbf{u}_n)$, i.e., it is linear in each \mathbf{u}_j and $c_{n_1, n_2, n_3}(\mathbf{u}_1, \dots, \mathbf{u}_n) = c_{n_1, n_2, n_3}(\mathbf{u}_{\sigma(1)}, \dots, \mathbf{u}_{\sigma(n)})$ for all $\sigma \in \mathfrak{S}_n$, where \mathfrak{S}_n is the permutation group

of n elements. Since each \mathbf{u}_j is linear in \mathbf{x} , then each c_{n_1, n_2, n_3} must be a homogeneous polynomial of degree n in \mathbf{x} , i.e., $c_{n_1, n_2, n_3} = \mathbf{a}_{n_1, n_2, n_3}^\top \nu_n(\mathbf{x})$, where each entry of $\mathbf{a}_{n_1, n_2, n_3} \in \mathbb{R}^{M_n(3)}$ is a symmetric multilinear function of the entries of the A_j 's. Letting

$$\mathcal{A} \doteq [\mathbf{a}_{n,0,0}, \mathbf{a}_{n-1,1,0}, \dots, \mathbf{a}_{0,0,n}]^\top \in \mathbb{R}^{M_n(3) \times M_n(3)},$$

we obtain $\text{MBCC}(\mathbf{x}, \mathbf{y}) = \nu_n(\mathbf{y})^\top \mathcal{A} \nu_n(\mathbf{x}) = 0$. \square

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 (7.19) clearly resembles the bilinear form of the affine constraint for a single affine motion, we will refer to both equations (7.18) and (7.19) as the *multibody affine constraint* from now on.

7.4.2 Computing the Number of 2-D Affine Motion Models

Since the multibody affine constraint (7.19) is valid for each $(\mathbf{x}_j, \mathbf{y}_j)$, one can write the MBCC for all the measurements as the following relationship between the number of motions and the stack of the columns of \mathcal{A} , $\text{vec}(\mathcal{A})$

$$\mathbf{V}_n^{\mathcal{A}} \mathbf{a} = [\nu_n(\mathbf{x}_1) \otimes \nu_n(\mathbf{y}_1) \cdots \nu_n(\mathbf{x}_N) \otimes \nu_n(\mathbf{y}_N)]^\top \text{vec}(\mathcal{A}) = 0. \quad (7.20)$$

In order to determine n and \mathcal{A} , we assume that a large enough number of image measurements are given and that such measurements are non-degenerate, i.e., they do not satisfy any homogeneous polynomial of degree less than or equal to n other than the MBCC. This assumption is analogous to the standard assumption in structure from motion that image measurements should not be images of 3-D points lying in a critical surface. Under this non-degeneracy assumption we have that

1. There is no polynomial of degree i less than n that is satisfied by every data point and so the data matrix of degree i , $\mathbf{V}_j^{\mathcal{A}}$, is of full column rank;
2. There is only one polynomial of degree n that is satisfied by all the data, and so $\mathbf{V}_n^{\mathcal{A}}$ is of rank $M_n(3)^2 - 1$ respectively;
3. There are two or more polynomials of degree $i > n$, namely any multiple of the MBCC, that are satisfied by all the data points, hence the null space of $\mathbf{V}_j^{\mathcal{A}}$ is at least two-dimensional.

This analysis imposes a rank constraints on $\mathbf{V}_n^{\mathcal{A}}$ which allows us to determine the number of affine motion models from the given image intensities as stated by the following theorem.

Theorem 7.5 (Number of 2-D Affine Motion Models). *Let $\mathbf{V}_j^{\mathcal{A}} \in \mathbb{R}^{N \times (M_j(3)^2)}$ be the matrix in (7.20), but computed with the Veronese map ν_j of degree $i \geq 1$. If $\text{rank}(A_j) \geq 2$ for all $j = 1, \dots, n$, and a large enough set of N image*

measurements in general configuration is given, then

$$\text{rank}(\mathbf{V}_j^{\mathcal{A}}) \begin{cases} > M_j(3)^2 - 1, & \text{if } i < n, \\ = M_j(3)^2 - 1, & \text{if } i = n, \\ < M_j(3)^2 - 1, & \text{if } i > n. \end{cases} \quad (7.21)$$

Therefore, the number of affine motions n is given by

$$n \doteq \min\{j : \text{rank}(\mathbf{V}_j^{\mathcal{A}}) = M_j(3)^2 - 1\}. \quad (7.22)$$

Proof. Since each affine matrix A_j satisfies $\text{rank}(A_j) \geq 2$, the polynomial $p_j = \mathbf{y}^\top A_j \mathbf{x}$ is irreducible over the real field \mathbb{R} . Let Z_j be the set of (\mathbf{x}, \mathbf{y}) that satisfy $\mathbf{y}^\top A_j \mathbf{x} = 0$. Then due to the irreducibility of p_j , any polynomial p in \mathbf{x} and \mathbf{y} that vanishes on the entire set Z_j must be of the form $p = p_j h$, where h is some polynomial. Therefore, if A_1, \dots, A_n are different, a polynomial that vanishes on the set $\cup_{j=1}^n Z_j$ 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

$$p = p_1 p_2 \cdots p_n = (\mathbf{x}_2^\top F_1 \mathbf{x}_1) (\mathbf{x}_2^\top F_2 \mathbf{x}_1) \cdots (\mathbf{x}_2^\top F_n \mathbf{x}_1). \quad (7.23)$$

Since the rows of $\mathbf{V}_n^{\mathcal{A}}$ are of the form $(\nu_n(\mathbf{x}) \otimes \nu_n(\mathbf{y}))^\top$ and the entries of $\nu_n(\mathbf{x}) \otimes \nu_n(\mathbf{y})$ are exactly the independent monomials of p (as we will show shortly), this implies that if a large enough number of image pairs in general configuration is given, then:

1. There is no polynomial of degree $2i < 2n$ whose coefficients are in the null space of $\mathbf{V}_j^{\mathcal{A}}$, i.e., $\text{rank}(\mathbf{V}_j^{\mathcal{A}}) = M_j^2 > M_j^2 - 1$ for $i < n$.
2. There is a unique polynomial of degree $2n$, namely p , whose coefficients are in the null space of $\mathbf{V}_n^{\mathcal{A}}$, i.e., $\text{rank}(\mathbf{V}_n^{\mathcal{A}}) = M_n^2 - 1$.
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 $\mathbf{V}_j^{\mathcal{A}}$, i.e., $\text{rank}(\mathbf{V}_j^{\mathcal{A}}) < M_j^2 - 1$ for $j > n$.

The rest of the proof is to show that the entries of $\nu_n(\mathbf{x}) \otimes \nu_n(\mathbf{y})$ 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 $\mathbf{x} = [x_1, x_2, x_3]^\top$ and $\mathbf{y} = [y_1, y_2, y_3]^\top$. Then the entries of $\nu_n(\mathbf{x}) \otimes \nu_n(\mathbf{y})$ are of the form $(y_1^{m_1} y_2^{m_2} y_3^{m_3})(x_1^{n_1} x_2^{n_2} x_3^{n_3})$ with $m_1 + m_2 + m_3 = n_1 + n_2 + n_3 = n$, while the entries of $\mathbf{x} \otimes \mathbf{y}$ are of the form $(y_1^{i_1} y_2^{i_2} y_3^{i_3})(x_1^{j_1} x_2^{j_2} x_3^{j_3})$ 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 $\nu_{n+1}(\mathbf{x}) \otimes \nu_{n+1}(\mathbf{y})$. \square

Once n is known, we can solve for \mathcal{A} uniquely from the linear system (7.20) by enforcing the $(M_n(3), M_n(3))$ entry of \mathcal{A} to be one. This additional equation results from the fact that the $(3, 3)$ entry of each A_j is one.

7.4.3 Computing the 2-D Affine Motion Model Parameters

In this section, we exploit the geometric properties of \mathcal{A} to obtain the following purely geometric solution for computing $\{A_j\}_{j=1}^n$.

1. Compute derivatives of the MBCC with respect to \mathbf{x} to obtain linear combinations of the rows of each A_j .
2. Obtain the rows of each A_j up to a scale factor from the cross products of these linear combinations.
3. Solve linearly for the unknown scales the rows of A_j from the optical flow.

For step 1, note that if the image measurement (\mathbf{x}, \mathbf{y}) comes from the i th motion model, i.e., if $\mathbf{y}^\top A_j \mathbf{x} = 0$, then

$$\frac{\partial \text{MBCC}(\mathbf{x}, \mathbf{y})}{\partial \mathbf{x}} \sim \mathbf{y}^\top A_j. \quad (7.24)$$

That is, the derivatives of the MBCC with respect to \mathbf{x} give linear combinations of the rows of the affine model at \mathbf{x} . Now, since the optical flow $\mathbf{u} = [u, v, 1]^\top$ at pixel \mathbf{x} can be computed as in 7.6, we can define the vectors $\tilde{\mathbf{y}}_1 = [1, 0, -u]^\top$ and $\tilde{\mathbf{y}}_2 = [0, 1, -v]^\top$. Although these vectors are not actual image measurements, they do satisfy $\tilde{\mathbf{y}}_1^\top \mathbf{u} = \tilde{\mathbf{y}}_2^\top \mathbf{u} = 0$. Hence we can use them to obtain the following linear combination of the rows of the affine model A_j at (\mathbf{x}, \mathbf{y})

$$\mathbf{g}_{j1} \sim \mathbf{a}_{j1} - ue_3 \quad \text{and} \quad \mathbf{g}_{j2} \sim \mathbf{a}_{j2} - ve_3, \quad (7.25)$$

from the derivatives of the MBCC at $(\mathbf{x}, \tilde{\mathbf{y}}_1)$ and $(\mathbf{x}, \tilde{\mathbf{y}}_2)$, respectively.

For step 2, notice that since the 3-rd row of A_j is e_3^\top , we can obtain two vectors \mathbf{b}_{j1} and \mathbf{b}_{j2} that are orthogonal to \mathbf{a}_{j1} and \mathbf{a}_{j2} , respectively, as $\mathbf{b}_{j1} = \mathbf{g}_{j1} \times e_3 \sim \mathbf{a}_{j1} \times e_3$ and $\mathbf{b}_{j2} = \mathbf{g}_{j2} \times e_3 \sim \mathbf{a}_{j2} \times e_3$. Although the pairs (\mathbf{b}_{j1}, e_1) and (\mathbf{b}_{j2}, e_2) are not actual image measurements, they do satisfy $e_1^\top A_j \mathbf{b}_{j1} = \mathbf{a}_{j1}^\top \mathbf{b}_{j1} = 0$ and $e_2^\top A_j \mathbf{b}_{j2} = \mathbf{a}_{j2}^\top \mathbf{b}_{j2} = 0$. Therefore we can immediately compute the rows of A_j up to a scale factor as

$$\mathbf{a}_{j1}^\top \sim \tilde{\mathbf{a}}_{j1}^\top = \left. \frac{\partial \text{MBCC}(\mathbf{x}, \mathbf{y})}{\partial \mathbf{x}} \right|_{(\mathbf{x}, \mathbf{y})=(\mathbf{b}_{j1}, e_1)}, \quad (7.26)$$

$$\mathbf{a}_{j2}^\top \sim \tilde{\mathbf{a}}_{j2}^\top = \left. \frac{\partial \text{MBCC}(\mathbf{x}, \mathbf{y})}{\partial \mathbf{x}} \right|_{(\mathbf{x}, \mathbf{y})=(\mathbf{b}_{j2}, e_2)}. \quad (7.27)$$

For step 3, we know the rows of A_j up to scale, i.e., $\mathbf{a}_{j1} = \lambda_{j1} \tilde{\mathbf{a}}_{j1}$ and $\mathbf{a}_{j2} = \lambda_{j2} \tilde{\mathbf{a}}_{j2}$, and the optical \mathbf{u} flow at pixel \mathbf{x} , i.e., $\mathbf{u} = A_j \mathbf{x}$. Therefore, $u = \lambda_{j1} \tilde{\mathbf{a}}_{j1}^\top \mathbf{x}$ and $v = \lambda_{j2} \tilde{\mathbf{a}}_{j2}^\top \mathbf{x}$, and so the unknown scales are automatically given by

$$\lambda_{j1} = u / (\tilde{\mathbf{a}}_{j1}^\top \mathbf{x}) \quad \text{and} \quad \lambda_{j2} = v / (\tilde{\mathbf{a}}_{j2}^\top \mathbf{x}). \quad (7.28)$$

By applying steps 1-3 to all N pixels in the image, we can effectively compute one affine matrix A for each pixel, without yet knowing the segmentation of the image measurements. Since in our model we only have $n \ll N$ different affine

matrices, we only need to apply steps 1-3 to n pixels corresponding to each one of the n models. We can automatically choose the n pixels at which to perform the computation using the same methodology proposed for 2-D translational motions. Once the $\{A_j\}_{j=1}^n$ are calculated we can cluster the data as

$$i = \arg \min_{\ell=1,\dots,n} \frac{|\mathbf{y}_j^T A_\ell \mathbf{x}_j|^2}{\|A_\ell \mathbf{x}_j\|^2}, \quad (7.29)$$

and then refine the affine motion model parameters by solving the linear equation $\mathbf{y}^\top A_j \mathbf{x} = 0$ for each separate cluster.

7.5 Segmentation of Motions Models of Different Type

Consider now the most challenging case in which the 2-D motion of the scene is generated from n_t 2-D translational motion models $\{\mathbf{u}_j\}_{j=1}^{n_t}$ and from n_a 2-D affine motion models $\{A_j\}_{j=1}^{n_a}$. Then, the MBCC (7.5) takes the form of a bi-homogeneous polynomial of degree n_a in \mathbf{x} and $n_a + n_t$ in \mathbf{y}

$$\text{MBCC}(\mathbf{x}, \mathbf{y}) = \prod_{j=1}^{n_t} (\mathbf{y}^\top \mathbf{u}_j) \prod_{j=1}^{n_a} (\mathbf{y}^\top A_j \mathbf{x}) = 0. \quad (7.30)$$

Thanks to the MBCC, the problem of segmenting both 2-D translational and 2-D affine motion models is mathematically equivalent to segmenting data lying on a collection of both linear and quadratic surfaces. If we knew the type of motion model associated with each pixel (translational or affine), we could immediately separate the data into two groups and then apply the algorithms for 2-D translational and 2-D affine motions developed in the previous two sections to each one of the groups. Since in practice we do not know the type of motion associated with each pixel, one alternative is to use the QSA algorithm described in Chapter 12 for clustering both linear and quadratic surfaces. However, the fact that $\mathbf{y}^\top A_j \mathbf{x} = 0$ is not a general quadratic surface, but rather a bilinear surface, gives us some additional algebraic structure which we can exploit to solve the 2-D motion segmentation problem more efficiently.

In the following subsections, we present an algebraic algorithm for segmenting 2-D translational and 2-D affine motion models. We first derive a rank constraint on the image measurements from which one can compute the number of translational and affine motion models using a simple one-dimensional search. We then demonstrate that a sub-matrix of the Hessian of the MBCC encodes information about the type of motion models: The matrix is rank-1 for 2-D translational models and rank-3 for 2-D affine models. Once the type of motion model has been identified, we show that the parameters of each motion model can be obtained by directly applying a subset of the algorithms described in the previous sections.

7.5.1 The Multibody Motion Matrix

Let $\mathcal{U} \in \mathbb{R}^{M_{n_t}(3)}$ be the multibody optical flow associated with $\{\mathbf{u}_j\}_{j=1}^{n_t}$, $\mathcal{A} \in \mathbb{R}^{M_{n_a}(3) \times M_{n_a}(3)}$ be the multibody affine matrix associated with $\{A_j\}_{j=1}^{n_a}$, and $\mathbf{a}_{m_1, m_2, m_3}^T$ be the (m_1, m_2, m_3) th row of \mathcal{A} . We can write the MBCC as

$$\begin{aligned} \text{MBCC}(\mathbf{x}, \mathbf{y}) &= (\nu_{n_t}(\mathbf{y})^\top \mathcal{U})(\nu_{n_a}(\mathbf{y})^\top \mathcal{A} \nu_{n_a}(\mathbf{x})) \\ &= \sum_n y_1^{n_1} y_2^{n_2} y_3^{n_3} \mathcal{U}_{n_1, n_2, n_3} \sum_m y_1^{m_1} y_2^{m_2} y_3^{m_3} \mathbf{a}_{m_1, m_2, m_3}^T \nu_{n_a}(\mathbf{x}) \\ &= \sum_m \sum_n y_1^{n_1+m_1} y_2^{n_2+m_2} y_3^{n_3+m_3} \mathcal{U}_{n_1, n_2, n_3} \mathbf{a}_{m_1, m_2, m_3}^T \nu_{n_a}(\mathbf{x}) \\ &= \sum_k y_1^{k_1} y_2^{k_2} y_3^{k_3} \mathbf{m}_{k_1, k_2, k_3}^T \nu_{n_a}(\mathbf{x}) = \nu_{n_a+n_t}(\mathbf{y})^\top \mathcal{M} \nu_{n_a}(\mathbf{x}) = 0, \end{aligned}$$

where $\mathbf{m}_{k_1, k_2, k_3}^T = \sum_n \mathcal{U}_{n_1, n_2, n_3} \mathbf{a}_{k_1-n_1, k_2-n_2, k_3-n_3}^\top$ is the (k_1, k_2, k_3) th row of $\mathcal{M} \in \mathbb{R}^{M_{n_a+n_t} \times M_{n_a}}$. The matrix \mathcal{M} is called the *multibody motion matrix* and contains information about all the motion models $\{\mathbf{u}_j\}_{j=1}^{n_t}$ and $\{A_j\}_{j=1}^{n_a}$. Note that when $n_a = 0$, \mathcal{M} is equivalent to the multibody optical flow \mathcal{U} and when $n_t = 0$, \mathcal{M} is equivalent to the multibody affine matrix \mathcal{A} .

In order to compute \mathcal{M} , note that the MBCC holds at every image measurement $\{(\mathbf{x}_j, \mathbf{y}_j)\}_{j=1}^N$. Therefore, if the number of translational and affine motions are known, we can compute \mathcal{M} by solving the linear system,

$$\mathbf{V}_{n_a, n_t} \mathbf{m} = 0, \quad (7.31)$$

where the j th row of $\mathbf{V}_{n_a, n_t} \in \mathbb{R}^{N \times M_{n_a+n_t} M_{n_a}}$ is given as $\nu_{n_a}(\mathbf{x}_j) \otimes (\nu_{n_a+n_t}(\mathbf{y}_j))^\top$ and \mathbf{m} is the stack of the columns of \mathcal{M} . The scale of \mathcal{M} is obtained by enforcing the additional constraint $\mathcal{M}(M_{n_a+n_t}, M_{n_a}) = 1$, because $\mathbf{u}_j(3) = A_j(3, 3) = 1$ for all $j = 1, \dots, n_t$ and $j = 1, \dots, n_a$.

7.5.2 Computing the Number of 2-D Motion Models

Note that in order to solve for \mathcal{M} from the linear system $\mathbf{V}_{n_a, n_t} \mathbf{m} = 0$ we need to know the number of translational and affine models, n_t and n_a , respectively. In order to determine the number of models, we assume that the image measurements are non-degenerate, i.e., they do not satisfy any homogeneous polynomial in (\mathbf{x}, \mathbf{y}) of degree less than n_a in \mathbf{x} or less than $n_t + n_a$ in \mathbf{y} . This assumption is analogous to the standard assumption in structure from motion that image measurements should not live in a critical surface. Under this assumption we have the following result:

Theorem 7.6 (Number of Translational and Affine Models). *Let $\mathbf{V}_{n'_a, n'_t} \in \mathbb{R}^{N \times M_{n'_t+n'_a} M_{n'_a}}$ be the matrix in (7.31), but computed with the Veronese map of degree n'_a in \mathbf{x} and $n'_a + n'_t \geq 1$ in \mathbf{y} . If $\text{rank}(A_j) \geq 2$ for all $j = 1, \dots, n_a$, and a large enough set of image measurements in general configuration is given,*

then the number of affine and translational motions are, respectively, given by

$$\begin{aligned} n_a &= \arg \min_{n'_a} \{n'_a : \exists n'_t \geq 0 : \mathbf{V}_{n'_a, n'_t} \text{ drops rank by 1}\}, \\ n_t &= \arg \min_{n'_t} \{n'_t : \mathbf{V}_{n'_a, n'_t} \text{ drops rank by 1}\}. \end{aligned} \quad (7.32)$$

Proof. From the non-degeneracy assumption we have that

1. If $n'_a < n_a$ or $n'_t + n'_a < n_t + n_a$, there is no polynomial of degree n'_a in \mathbf{x} or of degree $n'_a + n'_t$ in \mathbf{y} fitting the data, hence $\mathbf{V}_{n'_a, n'_t}$ is of full column rank.
2. If $n'_t + n'_a = n_t + n_a$ and $n'_t \leq n_t$, there is exactly one polynomial fitting the data, namely $\nu_{n'_t + n'_a}(\mathbf{y})^\top \mathcal{M} \nu_{n'_a}(\mathbf{x})$, thus $\mathbf{V}_{n'_a, n'_t}$ drops rank by 1. This is true for all $n'_t \leq n_t$, given $n'_t + n'_a = n_t + n_a$, because each translational motion model can also be interpreted as an affine motion model.
3. If $n'_t + n'_a > n_t + n_a$ and $n'_a \geq n_a$, there are two or more polynomials of degree n'_a in \mathbf{x} and $n'_a + n'_t$ in \mathbf{y} that fit the data, namely any multiple of the MBCC. Therefore, the null space of $\mathbf{V}_{n'_a, n'_t}$ is at least two-dimensional and $\mathbf{V}_{n'_a, n'_t}$ drops rank by more than 1.

From the above cases, we conclude that there are multiple values of (n'_a, n'_t) for which the matrix $\mathbf{V}_{n'_a, n'_t}$ drops rank exactly by 1, i.e., whenever $n'_t + n'_a = n_t + n_a$ and $n'_t \leq n_t$. Therefore the correct number of motions (n_a, n_t) can be obtained as in (7.32). \square

As a consequence of the theorem, we can immediately devise a strategy to search for the correct number of motions. We know that the correct number of motions (n_a, n_t) occurs for the minimum value of n'_a such that $n'_t + n'_a = n_t + n_a$ and $\mathbf{V}_{n'_a, n'_t}$ drops rank by 1. Thus we can initially set $(n'_a, n'_t) = (0, 1)$ and if $\mathbf{V}_{n'_a, n'_t}$ does not drop rank we can increase n'_a while keeping $n'_a + n'_t$ constant until $\mathbf{V}_{n'_a, n'_t}$ drops rank. If $\mathbf{V}_{n'_a, n'_t}$ does not drop rank for this value of $n'_a + n'_t$, we increase $n'_a + n'_t$ by one, reset $n'_a = 0$ and repeat the process until $\mathbf{V}_{n'_a, n'_t}$ drops rank by 1 for the first time. This process will stop at the true (n_a, n_t) .

Figure 7.2 illustrates our method for searching for the number of motions (n_a, n_t) in the particular case of $n_a = 3$ affine motions and $n_t = 2$ translational motions. In this case, we search for the correct (n_a, n_t) in the following order $(0, 1), (1, 0), (0, 2), (1, 1), (2, 0), (0, 3), \dots, (0, 5), (1, 4), (2, 3), (3, 2)$.

Notice that the proposed search strategy will give the correct number of motions with perfect data, but will fail with noisy data, because the matrix $\mathbf{V}_{n'_a, n'_t}$ will be full rank for all (n'_a, n'_t) . Inspired by the criterion (2.14) for determining the rank of a noisy matrix given in Chapter 2, we find (n_a, n_t) as the pair that minimizes the cost function

$$\frac{\sigma_{M_{n'_t+n'_a} M_{n'_a}}^2(\mathbf{V}_{n'_a, n'_t})}{\sum_{j=1}^{M_{n'_t+n'_a} M_{n'_a}-1} \sigma_j^2(\mathbf{V}_{n'_a, n'_t})} + \kappa_1(n'_a + n'_t) + \kappa_2 n'_a. \quad (7.33)$$

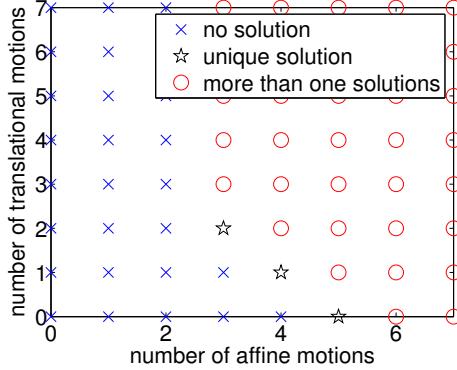


Figure 7.2. Plot of the possible pairs of (n'_a, n'_t) that give a unique solution for the MBCC. The actual pair is $(3, 2)$.

In (7.33) $\sigma_j(L)$ is the j th singular value of L , and κ_1 and κ_2 are parameters that penalize increasing the complexity of the multibody motion model \mathcal{M} , either by increasing the number of affine motions, or by increasing the total number of motions. As before, this two-dimensional optimization problem is reduced to a one-dimensional search by evaluating the cost function for values of (n'_a, n'_t) chosen in the order $(0, 1), (1, 0), (0, 2), (1, 1), (2, 0), (0, 3), \dots$.

7.5.3 Computing the Type of 2-D Motion at Each Pixel

Given the number of motion models (n_a, n_t) and the multibody motion model \mathcal{M} , we now show how to determine the type of motion model associated with each pixel: 2-D translational or 2-D affine. As it turns out, this can be done in a remarkably simple way by looking at the rank of the matrix

$$\mathcal{H}(\mathbf{x}, \mathbf{y}) = \frac{\partial \text{MBCC}(\mathbf{x}, \mathbf{y})}{\partial \mathbf{y} \partial \mathbf{x}} \in \mathbb{R}^{3 \times 3}. \quad (7.34)$$

For the sake of simplicity, consider a scene whose optical flow at every pixel can be modeled by one translational and one affine motion model, \mathbf{u} and A , respectively. In this case, the MBCC can be written as $\text{MBCC}(\mathbf{x}, \mathbf{y}) = (\mathbf{y}^\top \mathbf{u})(\mathbf{y}^\top A \mathbf{x})$, hence

$$\mathcal{H}(\mathbf{x}, \mathbf{y}) = \mathbf{u} \mathbf{y}^\top A + (\mathbf{y}^\top \mathbf{u}) A. \quad (7.35)$$

Therefore, if an image measurement comes from the translational motion model only, i.e., if $\mathbf{y}_j^\top \mathbf{u} = 0$, then

$$\mathcal{H}(\mathbf{x}_j, \mathbf{y}_j) = \mathbf{u} (\mathbf{y}_j^\top A) \implies \text{rank}(\mathcal{H}(\mathbf{x}_j, \mathbf{y}_j)) = 1. \quad (7.36)$$

Similarly, if the image measurement comes from the affine motion model, i.e., if $\mathbf{y}_j^\top A \mathbf{x}_j = 0$, then

$$\mathcal{H}(\mathbf{x}_j, \mathbf{y}_j) = \mathbf{u} (\mathbf{y}_j^\top A) + (\mathbf{y}_j^\top \mathbf{u}) A \Rightarrow \text{rank}(\mathcal{H}(\mathbf{x}_j, \mathbf{y}_j)) = 3. \quad (7.37)$$

This simple observation for the case $n_a = n_t = 1$ generalizes to any value of n_a and n_t as stated in the following theorem.

Theorem 7.7 (Identification of the Motion Type). *Let $\mathcal{M} \in \mathbb{R}^{M_{n_a+n_t}(3) \times M_{n_a}(3)}$ be the multibody motion model associated with n_t 2-D translational motions $\{\mathbf{u}_j\}_{j=1}^{n_t}$ and n_a 2-D affine motions $\{A_j\}_{j=1}^n$. The type of motion model associated with an image measurement (\mathbf{x}, \mathbf{y}) can be found as follows*

1. 2-D translational if $\text{rank}(\mathcal{H}(\mathbf{x}, \mathbf{y})) = 1$.
2. 2-D affine if $\text{rank}(\mathcal{H}(\mathbf{x}, \mathbf{y})) = 3$.

Thanks to Theorem 7.7, we can automatically determine the type of motion model associated with each image measurements. In the case of noisy image data, we can use equation (2.14) to determine the rank of a matrix. As simpler method applicable in this particular case is to declare a model to be 2-D affine if

$$\frac{\sqrt[3]{|\det(H(\mathbf{x}_j, \mathbf{y}_j))|}}{\text{trace}(H(\mathbf{x}_j, \mathbf{y}_j))} > \epsilon. \quad (7.38)$$

We have found a threshold of $\epsilon = 0.03$ to work well in all our experiments.

7.5.4 Computing the 2-D Motion Model Parameters

Given the number and types of motion models, and the multibody motion model \mathcal{M} , we now show how to compute the individual 2-D translational $\{\mathbf{u}_j\}_{j=1}^{n_t}$ and 2-D affine $\{A_j\}_{j=1}^{n_a}$ motion models. One possible method is to simply separate the data into two groups, 2-D translational data and 2-D affine data, and then solve separately for the 2-D translational and 2-D affine motion models by using the algorithms in the previous two sections. This amounts to solving for the multibody optical flow \mathcal{U} from (7.11) and the multibody affine matrix \mathcal{A} from (7.20), and then applying polynomial differentiation to obtain $\{\mathbf{u}_j\}_{j=1}^{n_t}$ from \mathcal{U} and $\{A_j\}_{j=1}^{n_a}$ from \mathcal{A} .

However, at this point we already have the multibody motion \mathcal{M} which is a matrix representation for $\mathcal{U} \otimes \mathcal{A} + \mathcal{A} \otimes \mathcal{U}$. Therefore, having to recompute \mathcal{U} and \mathcal{A} would be extra unnecessary computation. In this subsection we show that the one can directly compute $\{\mathbf{u}_j\}_{j=1}^{n_t}$ and $\{A_j\}_{j=1}^{n_a}$ from the derivatives of the MBCC defined by the multibody motion model \mathcal{M} .

To this end, recall from Theorem 7.1 that one can compute the optical flow at each pixel \mathbf{x} from the partial derivative of the MBCC with respect of \mathbf{y} at (\mathbf{x}, \mathbf{y}) . Therefore, we can immediately obtain the 2-D translational motions $\{\mathbf{u}_j\}_{j=1}^{n_t}$ by applying steps 2 and 3 of the algorithm in Section 7.3 to the pixels obeying a 2-D translational motion model, as determined in the previous subsection.

In an entirely analogous fashion, recall from Section 7.4 that in the case of 2-D affine motion models the computation of the affine matrices $\{A_j\}_{j=1}^{n_a}$ relies on the fact that the derivatives of the MBCC with respect to \mathbf{x} give a linear combination of the rows of A_j , where A_j is the affine model associated with \mathbf{x} (see equation

(7.24)). It is obvious from equation (7.30) that the vector of partial derivatives of the MBCC with respect to \boldsymbol{x} is not affected by 2-D translational motions, because 2-D translational motions do not depend on \boldsymbol{x} . Therefore, we can immediately obtain the 2-D affine motions $\{A_j\}_{j=1}^{n_a}$ by applying steps 1-3 of the algorithm in Section 7.4.3 to the pixels obeying a 2-D affine motion model, as determined in the previous subsection.

7.6 Algorithm Summary

We can summarize the discussions in sections 7.3-7.5 in the form of Algorithm 7.1. The algorithm outlines the steps to calculate the multibody motion parameters and the individual motion models and then segment the motion of the scene.

Notice that the minimum number of image measurements needed in order to compute the multibody motion model \mathcal{M} is $N \geq M_{n_a+n_t}(3)M_{n_t}(3) - 1$. Table 7.1 gives numeric values of the minimum N as a function of the number of affine and translational models. Notice that in most cases less than 800 pixels are needed, which is feasible even with a 100×100 image.

Table 7.1. Minimum number of measurements as a function of the number of models.

(n_a, n_t)	0	1	2	3	4	5
0	2	5	9	14	20	
1	6	15	27	42	60	81
2	24	48	78	114	156	204
3	64	114	174	244	324	414
4	139	229	334	454	589	739

7.7 Experimental Results

In this section, we evaluate the performance of Algorithm 7.1 for the 2-D affine motion models, as a function of the level of noise and the number of motion models using synthetically generated data. We compare the performance of the following algorithms

1. *Factorization*: given \mathcal{A} , this algorithm solves for the affine motion parameters by bi-homogeneous polynomial factorization of the MBCC as described in [Vidal and Sastry, 2002].
2. *Differentiation*: given \mathcal{A} , this algorithm solves for the affine motion parameters by polynomial differentiation of the MBCC as described in Section 7.4.
3. *Complex differentiation*: given the optical flow computed by the differentiation algorithm, this algorithm transforms the two equations (7.15) and

Algorithm 7.1 (2-D Motion Segmentation from Image Derivatives).

Given N image measurements $\{(\mathbf{x}_j, \mathbf{y}_j)\}_{j=1}^N$ of a scene undergoing n_t 2-D translational and n_a 2-D affine motion models, recover the number of motion models (n_a, n_t) , the optical flow $\mathbf{u}(\mathbf{x})$ at each pixel \mathbf{x} , the type of motion model at each pixel, the parameters $\{\mathcal{M}_j\}_{j=1}^n$ of the $n = n_a + n_t$ motion models, and the model object associated with each image measurement as follows:

1. **Number of motion models:** Apply the Veronese map of various degrees to the data $\{(\mathbf{x}_j, \mathbf{y}_j)\}_{j=1}^N$ to form the embedded data matrix $\mathbf{V}_{n'_a, n'_t}^{\mathcal{M}}$ and compute the number of motions (n_a, n_t) as in (7.33).
2. **Multibody motion model:** Compute \mathcal{M} from the singular vector of $\mathbf{V}_{n_a, n_t}^{\mathcal{M}}$ associated with its smallest singular value and let

$$\text{MBCC}(\mathbf{x}, \mathbf{y}) = \nu_{n_a+n_t}(\mathbf{y})^\top \mathcal{M} \nu_{n_a}(\mathbf{x}).$$

3. **Optical flow:** Compute the optical flow at each pixel as:

$$\mathbf{u}(\mathbf{x}_j) = \frac{\partial \text{MBCC}(\mathbf{x}_j, \mathbf{y}_j)}{\partial \mathbf{y}_j} / \left(e_3^\top \frac{\partial \text{MBCC}(\mathbf{x}_j, \mathbf{y}_j)}{\partial \mathbf{y}_j} \right).$$

4. **Motion type.** Assign point $(\mathbf{x}_j, \mathbf{y}_j)$ to the 2-D translational group G_t if $\text{rank}(\mathcal{H}(\mathbf{x}_j, \mathbf{y}_j))=1$ and to the 2-D affine group G_a if $\text{rank}(\mathcal{H}(\mathbf{x}_j, \mathbf{y}_j))=3$.

5. **Motion segmentation:**

for $j = n_t : 1$

Select one representative optical flow $(\mathbf{x}_j, \mathbf{u}_j = \mathbf{u}(\mathbf{x}_j))$ per translational motion model according to (7.14).

end

for $j = n_a : 1$

Select one representative optical flow $(\mathbf{x}_j, \mathbf{u}_j = \mathbf{u}(\mathbf{x}_j))$ per affine motion model according to (7.14);

$$\tilde{\mathbf{y}}_{j1} = [1, 0, -\mathbf{u}_j]^\top;$$

$$\tilde{\mathbf{y}}_{j2} = [0, 1, -\mathbf{v}_j]^\top;$$

$$\mathbf{b}_{j1} = e_3 \times \frac{\partial \text{MBCC}}{\partial \mathbf{y}}(\mathbf{x}, \tilde{\mathbf{y}}_{j1});$$

$$\mathbf{b}_{j2} = e_3 \times \frac{\partial \text{MBCC}}{\partial \mathbf{y}}(\mathbf{x}_i, \tilde{\mathbf{y}}_{j2});$$

$$\tilde{\mathbf{a}}_{j1} = \frac{\partial \text{MBCC}}{\partial \mathbf{x}}(e_1, \mathbf{b}_{j1});$$

$$\tilde{\mathbf{a}}_{j2} = \frac{\partial \text{MBCC}}{\partial \mathbf{x}}(e_2, \mathbf{b}_{j2});$$

$$A_j = \begin{bmatrix} (e_1^\top \mathbf{u}_j) \tilde{\mathbf{a}}_{j1} & (e_2^\top \mathbf{u}_j) \tilde{\mathbf{a}}_{j1} & e_3 \end{bmatrix}^\top.$$

end

6. **Feature segmentation:** Assign $(\mathbf{x}_j, \mathbf{y}_j)$ to group $\arg \min_{\ell} \frac{|\mathbf{y}_j^\top \mathbf{u}_\ell(\mathbf{x}_j)|^2}{\|\mathbf{u}_\ell(\mathbf{x}_j)\|^2}$.

7. **Refining the motion model parameters:** Given the clustering of the image measurements into n groups, refine the motion model parameters for each separate cluster by from all the points belonging to that particular cluster.

(7.16) into a single equation in the complex domain. Taking the product of these equations for each one of the n models leads to a complex multibody affine model, $\mathcal{A}_c \in \mathbb{C}^{M_n(3)}$, from which the individual affine models are obtained by polynomial differentiation, similarly to the 2-D translational case described in Section 7.3.

4. *K-means*: starting from an initial set of affine matrices, this algorithm alternates between assigning points to clusters using the distance in (7.29) and computing (linearly) the affine models for each motion class.
5. *K-means + Differentiation*: we use our differentiation algorithm to initialize the K-means algorithm so that it has good estimates of the affine matrices to start with rather than choosing the initial values randomly.

We then present experimental results for various indoor and outdoor sequences for both, the translational and affine motion models. For the real sequences, we obtain the image derivatives at each frame using derivative of Gaussian filters of order 6 and then apply Algorithm 7.1 to each frame.

7.7.1 Simulation Results

We first test the algorithm on synthetic data. We randomly pick $n = 2$ collections of $N = 200$ pixel coordinates and apply a different (randomly chosen) affine motion model to each collection of pixels to generate their optical flow. From the optical flow associated with each pixel, we randomly choose a vector \mathbf{y} of spatial and temporal image derivatives satisfying the brightness constancy constraint (7.4). The coordinates of \mathbf{y} are constrained to be in $[-1, 1]$ to simulate image intensities in the $[0, 1]$ range. Zero-mean Gaussian noise with standard deviation $\sigma \in [0, 0.05]$ is added to the partial derivatives \mathbf{y} . We run 1000 trials for each noise level. For each trial the error between the true affine motions $\{A_j\}_{j=1}^n$ and the estimates $\{\hat{A}_j\}_{j=1}^n$ is computed as

$$\text{Affine error} = \frac{1}{n} \sum_{j=1}^n \frac{\|A_j - \hat{A}_j\|}{\|A_j\|} \quad (\%). \quad (7.39)$$

Figure 7.3 plots the mean error and the mean percentage of correctly classified pixels as a function of σ , by using the polynomial differentiation approach. In all trials the number of motions was correctly estimated from equation (7.33) as $n = 2$.¹ Notice that K-means randomly initialized usually converges to a local minima. The average number of iterations is 20. The factorization algorithm performs better than K-means for a small level of noise, but its performance deteriorates quickly as noise increases. The differentiation algorithm's estimates are within 6% of the true affine motions, with a percentage of correct classification of over 90%, even for a noise level of 5% in the image derivatives. The best results

¹We use $\kappa = 10^{-6}$ to determine the number of motions.

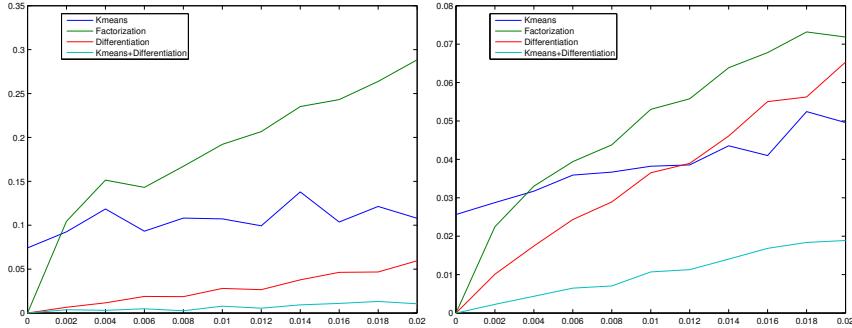


Figure 7.3. Error in the estimation of the affine model parameters and percentage of correctly classified points as a function of noise.

are obtained by using the differentiation algorithm to initialize K-means. Then the error reduces to about 2%, the average number of iterations reduces to 5, and the percentage of correctly classified points increases to 95%.

7.7.2 2-D Translational Motions

Figure 7.4 shows segmentation results for a 240×320 sequence of a car leaving a parking lot. The top row shows the pixels associated with the camera's downward motion and the bottom row shows the pixels associated with the car's right-downward motion. In each row, pixels that do not correspond to the group are colored black. Figure 7.5 shows another example on the segmentation of a 240×320 sequence of a person's head rotating from right to left in front of a lab background. The top row shows the pixels associated with the camera's fronto-parallel motion and the bottom row shows the pixels associated with the head motion. In each row, pixels that do not correspond to the group are colored red.



Figure 7.4. Segmentation of 4 frames from the car-parking lot sequence. Top: pixels associated with the camera motion. Bottom: pixels associated with the car motion.

The results in Figure 7.4 and 7.5, were obtained by applying Algorithm 7.1 directly to all the image data, without any pre or post processing. Therefore, many pixels can be potentially misclassified, such as pixels in low textured regions, e.g., parts of the body of each car, the road, the wall in the lab, as well as pixels in highly specular regions where the BCC is not satisfied. In addition, nearby pixels need not belong to the same group, because motion is the only cue used for segmentation.



Figure 7.5. Segmentation of 4 frames from the head-lab sequence. Top: pixels associated with the camera motion. Bottom: pixels associated with the head motion.

The segmentation results are encouraging. Although we are using a simple mixture of two 2-D translational motion models, i.e., the optical flow is assumed to be piecewise constant, most of the pixels corresponding to the moving car or the moving head are correctly segmented from those of the moving background. Most of the errors occur precisely at regions with low texture, such as the road in Figure 7.4 and the black sweater and white wall regions in Figure 7.5. Overall, about 85% of the image pixels are correctly classified with respect to ground truth manual segmentation. These results can be used as an initial segmentation for any more computationally intense nonlinear iterative refinement scheme.

7.7.3 2-D Affine Motions

Figure 7.6 shows an aerial view of a robot rotating and translating on the ground. The camera is fronto-parallel to the ground and undergoing both rotational and translational motion. Notice that various regions in the image correspond to pieces of the robots made of aluminum, which are highly specular and have little texture. Nevertheless, the algorithm correctly classifies about 85% of the pixels in the image with respect to ground truth manual segmentation.

7.7.4 2-D Translational and 2-D Affine Motions

Figure 7.7 shows segmentation results for a 240×320 sequence of a car leaving a parking lot. The sequence has 2 motions, the camera's downward motion which

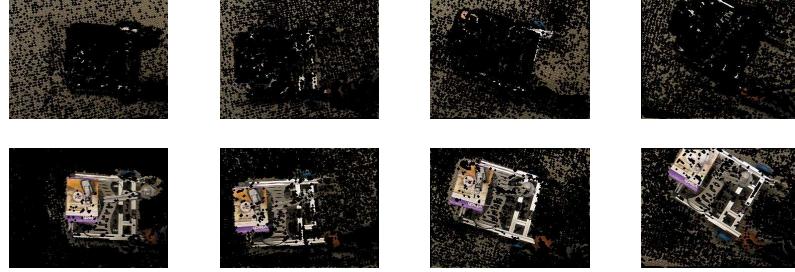


Figure 7.6. Segmenting four frames from robot sequence.

can be modeled as an affine motion and the car's right-downward motion which can be modeled as a translation. The first and second columns in the figure show the segmentation obtained assuming that the scene has 2 translations and 2 affine motions, respectively. The third column is the segmentation obtained assuming that the scene has 1 translation and 1 affine motion. In each image, pixels that do not correspond to the group are colored black. Notice that when both the motion models are assumed to be translational, the segmentation of the car is noisy, and when both the motion models are assumed to be affine, a portion of the parking lot gets segmented along with the car.

Figure 7.8 shows another example of segmentation of a 240×320 sequence of a person's head rotating from right to left in front of a lab background. This sequence too has 2 motions, the camera's fronto-parallel motion which can be modeled as a translation and the motion of the head which can be modeled as an affine motion. The first and second columns in the figure show the segmentation obtained assuming that the scene has 2 translations and 2 affine motions respectively. The third column is the segmentation obtained assuming that the scene has 1 translation and 1 affine motion. In each row, pixels that do not correspond to the group are colored red. We see that using motion models of different type helps segment the head more cleanly as compared to using motion models of the same type.

Figure 7.9 shows the the segmentation results of a 240×320 sequence of a helicopter landing. The sequence has 1 translational motion corresponding to the hovering motion of the helicopter and an affine motion corresponding to the fronto-parallel motion of the camera. The first column indicates pixels belonging to the motion of the camera and the second column indicates pixels belonging to the motion of the helicopter. In each image, pixels that do not correspond to the group are colored black. The results show that we obtain a very good segmentation of the helicopter, in spite of the fact that it constitutes a very small part of the image.

While applying Algorithm 7.1 to the sequences in Figure 7.9, we *pre-assigned* the type of motion model to the pixels corresponding to areas with low texture(*i.e.*, pixels corresponding to the sky) from the prior knowledge of the type of motion model that they should obey. The idea behind this is that the matrix



Figure 7.7. Segmenting 4 frames from the car-parking lot sequence.

$\mathcal{H}(\mathbf{x}, \mathbf{y})$ at such pixels would have rank 0 and this would result in their misclassification as translational points. However, we would like to emphasize that the results show that Algorithm 7.1 does give good segmentation results in the other areas having texture variation.



Figure 7.8. Segmenting 4 frames from the head-lab sequence.

7.8 Bibliographic Notes

Classical approaches to 2-D motion segmentation separate the image flow into different regions by looking for flow discontinuities [Spoerri and Ullman, 1987, Black and Anandan, 1991], fit a mixture of parametric models through successive computation of dominant motions [Irani et al., 1992] or use a layered representation of the motion field [Darrel and Pentland, 1991]. The problem has also



Figure 7.9. Segmenting 4 frames from a helicopter landing sequence. **Left:** pixels associated with the camera motion. **Right:** pixels associated with the helicopters motion

been formalized in a maximum likelihood framework [Jepson and Black, 1993, Ayer and Sawhney, 1995, Weiss, 1996, Weiss, 1997, Torr et al., 2001] in which the estimation of the motion models and their regions of support is done by alternating between the segmentation of the image measurements and the estimation of the motion parameters using the Expectation Maximization (EM) algorithm. EM-like approaches provide robust motion estimates by combining information over large regions in the image. However, their convergence to the optimal solution strongly depends on correct initialization [Shi and Malik, 1998, Torr et al., 2001]. Existing initialization techniques estimate a 2-D motion representation from local patches and cluster this representation using normalized cuts [Shi and Malik, 1998], or K-means [Wang and Adelson, 1993]. The drawback of

these approaches is that they are based on a local computation of 2-D motion, which is subject to the aperture problem and to the estimation of a single 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 [Ke and Kanade, 2002]. The only existing algebraic solution to 2-D motion segmentation is based on bi-homogeneous polynomial factorization and can be found in [Vidal and Sastry, 2002].

Classical approaches to 2-D motion segmentation are based on separating the image flow into different regions by looking for flow discontinuities [Spoerri and Ullman, 1987]. Due to the aperture problem, such techniques have trouble dealing with noisy flow estimates, especially in regions with low texture. Black and Anandan [Black and Anandan, 1991] 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. [Irani et al., 1992] 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 [Black and Anandan, 1996] and intensity information [Ayer et al., 1994], 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 [Darrel and Pentland, 1991] 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 [Jepson and Black, 1993, Ayer and Sawhney, 1995, Weiss, 1996, Weiss, 1997, Torr et al., 2001]. 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 model parameters (M-step). Jepson and Black [Jepson and Black, 1993] assume that the number of models is known and estimate the motion parameters using least squares. Ayer and Sawhney [Ayer and Sawhney, 1995] use MDL to determine the number of models and robust M-estimation to estimate the motion parameters. Weiss [Weiss, 1996] 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 [Weiss, 1997] and Torr et al. [Torr et al., 2001] noticed that the assumption of a parametric motion model (translational, affine or projective) is too restrictive for scenes which are non planar. [Weiss, 1997] proposes a non-parametric mixture model based on a probability distribution that favors smooth motion fields. [Torr et al., 2001] proposes a parametric model that includes some 3-D information by associating a disparity with each pixel, similar to the plane+parallax model [Irani and Anandan, 1999]. The model is initialized 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 their convergence to the optimal solution strongly depends on correct initialization [Shi and Malik, 1998, Torr et al., 2001]. To deal with the initialization problem, various techniques have been proposed. [Wang and Adelson, 1993] 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 drawbacks 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 [Ke and Kanade, 2002].

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* [Irani and Anandan, 1999]. Shi and Malik [Shi and Malik, 1998] 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.

7.9 Exercises

Exercise 7.1 Show that 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

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

Exercise 7.2 Let $\{A_j\}_{j=1}^n$ be a collection of n 2-D affine motion models, and recall that the entries (3,1) and (3,2) of each affine matrix A_j are zero.

1. Show that the entries $(n_1, n_2, n_3), (m_1, m_2, m_3)$ of the multibody affine matrix \mathcal{A} , with $0 \leq m_3 < n_3 \leq n$, are zero as well.
2. Show that the number of zeros in the multibody affine matrix is given by

$$Z_n \doteq n(n+1)(n+2)(3n+5)/24.$$

3. Let $\mathbf{V}_n^{\mathcal{A}} \in \mathbb{R}^{N \times M_n(3)^2}$ and $\mathbf{a} \in \mathbb{R}^{M_n(3)^2}$ be defined as in (7.20). Show that by enforcing the fact that Z_n entries of \mathcal{A} are zero one can solve for the entries of the multibody affine matrix \mathcal{A} from

$$\tilde{\mathbf{V}}_n^{\mathcal{A}} \tilde{\mathbf{a}} = 0, \quad (7.40)$$

where $\tilde{\mathbf{a}} \in \mathbb{R}^{M_n(3)^2 - Z_n}$ is the same as \mathbf{a} , but with the zero entries removed, and $\tilde{\mathbf{V}}_n^{\mathcal{A}} \in \mathbb{R}^{N \times (M_n(3)^2 - Z_n)}$ is the same as $\mathbf{V}_n^{\mathcal{A}} \in \mathbb{R}^{N \times M_n(3)^2}$, but with the corresponding Z_n columns removed.

4. Show that the number of affine motions n can also be computed as

$$n \doteq \min\{j : \text{rank}(\tilde{\mathbf{V}}_j^{\mathcal{A}}) = M_j(3)^2 - 1\}. \quad (7.41)$$

Proof. Since the entries (1,3) and (2,3) of A_j are zero, the monomials of $\mathbf{y}^\top A_j \mathbf{x}$ involving y_3 must also involve x_3 . Therefore, the coefficients of monomials in MBCC(\mathbf{x}, \mathbf{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 (y_1, y_2) , $(n-i+1)$, times the number of polynomials of degree $(n-j)$ in (x_1, x_2) , $(n-j+1)$, then $Z_n = \sum_{i=1}^n \sum_{j=0}^{i-1} (n-i+1)(n-j+1)$. \square

Exercise 7.3 Let $\{\mathbf{u}_j\}_{j=1}^{n_t}$ and $\{A_j\}_{j=1}^{n_a}$ be a collection of n_a 2-D affine motion and n_t 2-D translational motion models, and recall that the entries (3,1) and (3,2) of each affine matrix A_j are zero. If $n_a > 0$, show that Z_{n_a, n_t} entries of the multibody motion matrix

\mathcal{M} are zero. Find the number and location of such zero entries. Let \mathbf{V}_n and \mathbf{m} be defined as in 7.31. Show that one can solve for \mathcal{M} from

$$\tilde{\mathbf{V}}_{n_a, n_t} \tilde{\mathbf{m}} = 0, \quad (7.42)$$

where $\tilde{\mathbf{m}} \in \mathbb{R}^{M_{n_t} + n_a M_{n_a} - Z_{n_a, n_t}}$ is the same as \mathbf{m} , but with the Z_{n_a, n_t} zero entries removed, and $\tilde{\mathbf{V}}_{n_a, n_t} \in \mathbb{R}^{N \times (M_{n_a} + n_t M_{n_a} - Z_{n_a, n_t})}$ is the same as \mathbf{V}_{n_a, n_t} , but with the corresponding Z_{n_a, n_t} columns removed.

Chapter 8

3-D Motion Segmentation from Point Correspondences

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. When the scene is *static*, i.e., when either the camera or a single object move, the problem of fitting a 3-D model compatible with the structure and motion of the scene is well understood [Hartley and Zisserman, 2000, Ma et al., 2003]. For instance, it is well-known that two perspective views of a scene are related by the *epipolar constraint* [Longuet-Higgins, 1981] and that multiple views are related by the *multilinear constraints* [Heyden and Åström, 1997]. These constraints can be used to estimate a motion model for the scene using linear techniques such as the eight-point algorithm and its generalizations.

However, these techniques can not deal with *dynamic scenes* in which both the camera and an unknown number of objects with unknown 3-D structure move independently. In principle, one could model such scenes with a collection of 2-D motion models and segment them using the 2-D motion segmentation techniques developed in the previous chapter. However, because of depth discontinuities, perspective effects, etc, 2-D techniques would tend to interpret a single 3-D motion as multiple 2-D motions, which would result in over segmentation of the scene.

In this chapter, we develop techniques for segmentation of 3-D motion models. In particular, we consider the segmentation of three types of models of increasing complexity: linear, bilinear and trilinear. The segmentation of linear models shows up in motion segmentation from multiple affine views, and can be solved using the GPCA algorithm presented in Chapter 4. The segmentation of bilinear and trilinear models shows up in motion segmentation from point correspondences in two and three perspective views, respectively, and will require the development of extensions of GPCA to certain classes of bilinear and trilinear surfaces.

8.1 The Motion Estimation Problem

Before delving into the details of segmentation of multiple 3-D motion models, we present a brief overview of the classical 3-D motion estimation problem from point correspondences. Sections 8.1.2 and 8.1.3 review the two-view geometry of non-planar and planar scenes, respectively, and Section 8.1.4 reviews the three view geometry of non-planar scenes. We refer the readers to [Hartley and Zisserman, 2000, Ma et al., 2003] for further details.

8.1.1 Rigid-Body Motions and Camera Projection Models

Consider a video sequence taken by a moving camera observing a static scene. We assume that the camera is moving rigidly, so that its pose at frame $f = 1, \dots, F$ can be expressed as $(R_f, T_f) \in SE(3)$, where $R_f \in SO(3)$ is the camera rotation and $T_f \in \mathbb{R}^3$ is the camera translation. Without loss of generality, we assume that the first camera frame coincides with the world frame, so that $(R_1, T_1) = (I, \mathbf{0})$.

Consider a generic point p , with coordinates $\mathbf{X}_1 = (X_1, Y_1, Z_1)^\top \in \mathbb{R}^3$ relative to the world reference frame. As illustrated in Figure 8.1, the coordinates $\mathbf{X}_f = (X_f, Y_f, Z_f)^\top$ of the same point p relative to the f th camera frame are given the rigid-body transformation (R_f, T_f) of \mathbf{X}_1 :

$$\mathbf{X}_f = R_f \mathbf{X}_1 + T_f \quad \in \mathbb{R}^3. \quad (8.1)$$

Adopting the pinhole camera model shown in Figure 8.2 with focal length $d(f)$, the point p with coordinates \mathbf{X}_f is projected onto the image plane at the point

$$\begin{bmatrix} x_f \\ y_f \\ 1 \end{bmatrix} = \frac{1}{Z_f} \begin{bmatrix} d(f) & 0 & 0 \\ 0 & d(f) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_f \\ Y_f \\ Z_f \end{bmatrix}. \quad (8.2)$$

The projection model (8.2) is specified relative to a very particular reference frame centered at the optical center with one axis aligned with the optical axis. In

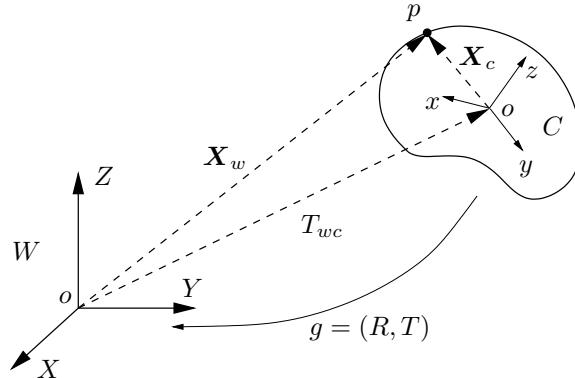


Figure 8.1. A rigid-body motion between a moving frame C and a world frame W .

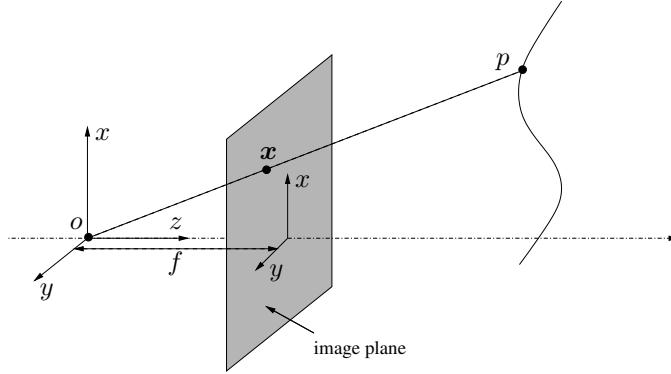


Figure 8.2. Frontal pinhole imaging model: the image of a 3-D point p is the point x at the intersection of the ray going through the optical center o and the image plane at a distance $d(f)$ in front of the optical center.

practice, when one captures digital images the measurements are obtained in pixel coordinates, which are related to the image coordinates by the transformation

$$\mathbf{x}_f \doteq \begin{bmatrix} s_x & s_\theta & o_x \\ 0 & s_y & o_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_f \\ y_f \\ 1 \end{bmatrix}, \quad (8.3)$$

where (s_x, s_y) is a scale factor, s_θ is a skew factor and (o_x, o_y) is a translation so that the origin is in the upper-left corner of the image.

Combining the camera motion model (8.1), the camera projection model (8.2) and the camera calibration model (8.3), leads to the following *camera model*:

$$\lambda_f \mathbf{x}_f = \underbrace{\begin{bmatrix} s_x & s_\theta & o_x \\ 0 & s_y & o_y \\ 0 & 0 & 1 \end{bmatrix}}_{K_f \in \mathbb{R}^{3 \times 3}} \begin{bmatrix} d(f) & 0 & 0 \\ 0 & d(f) & 0 \\ 0 & 0 & 1 \end{bmatrix} [R_f \quad T_f] \begin{bmatrix} X_1 \\ Y_1 \\ Z_1 \\ 1 \end{bmatrix}, \quad (8.4)$$

where $\lambda_f = Z_f$ and K_f are, respectively, the *depth* of the point and the *camera calibration matrix* in the f th frame. When $K_f = I$, we say that the camera is calibrated. We call the 3×4 matrix $\Pi_f = K_f[R_f \ T_f]$ the *projection matrix*.

8.1.2 The Fundamental Matrix

Let \mathbf{x}_1 and \mathbf{x}_2 be images of point p in the first and second frames of a video sequence consisting of $F = 2$ frames. As illustrated in Figure 8.3, the vectors \mathbf{X}_2, T_2 and $R_2 \mathbf{X}_1$ must be coplanar, hence their triple product must be zero, i.e.,

$$\mathbf{X}_2 \cdot (T_2 \times R_2 \mathbf{X}_1) = 0 \iff \mathbf{X}_2^\top \widehat{T_2} R_2 \mathbf{X}_1 = 0. \quad (8.5)$$

where $\widehat{T_2} \in so(3)$ is a skew-symmetric matrix generating the cross product by T_2 .

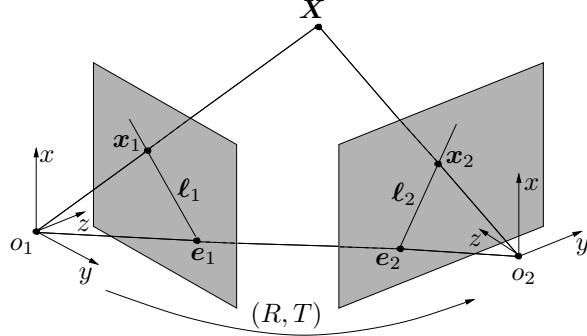


Figure 8.3. Epipolar geometry: Two projections $\mathbf{x}_1, \mathbf{x}_2 \in \mathbb{R}^3$ of a 3-D point \mathbf{X} from two vantage points. The relative Euclidean transformation between the two vantage points is given by $(R, T) \in SE(3)$. The intersections of the line (o_1, o_2) with each image plane are called *epipoles* and are denoted as e_1 and e_2 . The intersections of the plane (o_1, o_2, p) with the two image planes are called *epipolar lines* and are denoted ℓ_1 and ℓ_2 .

It follows from equation (8.4) that $\lambda_1 \mathbf{x}_1 = K_1 \mathbf{X}_1$ and $\lambda_2 \mathbf{x}_2 = K_2 \mathbf{X}_2$. Therefore, the following *epipolar constraint* [Longuet-Higgins, 1981] must be satisfied by the relative camera motion (R_2, T_2) and the image pair $(\mathbf{x}_1, \mathbf{x}_2)$

$$\mathbf{x}_2^\top K_2^{-\top} \widehat{T}_2 R_2 K_1^{-1} \mathbf{x}_1 = 0 \iff \mathbf{x}_2^\top F \mathbf{x}_1 = 0. \quad (8.6)$$

The matrix $F = K_2^{-\top} \widehat{T}_2 R_2 K_1^{-1} \in \mathbb{R}^{3 \times 3}$ is called the *fundamental matrix* and is defined up to an indeterminate scale. By construction, F is a rank-2 matrix having $e_1 = R_2^\top K_1 T_2$ and $e_2 = K_2 T_2$ as its right and left null spaces. The vectors e_1 and e_2 are known as the *epipoles* in the first and second view, respectively.

Since there are 9 unknowns in the fundamental matrix F (up to a scale), one can linearly solve for F from the epipolar constraint (8.6) from $N \geq 8$ point correspondences $\{(\mathbf{x}_{1i}, \mathbf{x}_{2i})\}_{i=1}^N$ in general configuration. Given some additional knowledge about the camera calibration K_1 and K_2 , one can solve for the camera motion (R_2, T_2) from F using the *eight-point algorithm* [Longuet-Higgins, 1981].

8.1.3 The Homography Matrix

The motion estimation scheme described in the previous subsection assumes that the displacement of the camera between the two views is nonzero, i.e., $T_2 \neq 0$, otherwise the fundamental matrix $F = K_2^{-\top} \widehat{T}_2 R_2 K_1^{-1}$ would be zero. Furthermore, it also requires that the 3-D points be in general configuration, otherwise one cannot uniquely recover F from the epipolar constraint [Hartley and Zisserman, 2000]. The latter case occurs, for example, when the 3-D points lie in a plane $\mathbf{N}^\top \mathbf{X}_1 = d$, where $\mathbf{N} \in \mathbb{S}^2$ is the normal to the plane and d is the distance from the plane to the origin of the first view. It follows from the equations $\mathbf{X}_2 = R_2 \mathbf{X}_1 + T_2$, $\mathbf{N}^\top \mathbf{X}_1 = d$, $\lambda_1 \mathbf{x}_1 = K_1 \mathbf{X}_1$ and $\lambda_2 \mathbf{x}_2 = K_2 \mathbf{X}_2$

that the following *homography constraint* holds

$$\mathbf{X}_2 = \left(R_2 + \frac{1}{d} T_2 \mathbf{N}^\top \right) \mathbf{X}_1 \implies \mathbf{x}_2 \sim K_2 \left(R_2 + \frac{1}{d} T_2 \mathbf{N}^\top \right) K_1^{-1} \mathbf{x}_1 \quad (8.7)$$

The matrix $H = K_2(R_2 + \frac{1}{d}T_2\mathbf{N}^\top)K_1^{-1}$ is called the *homography matrix* and is, in general, defined up to an indeterminate scale. Notice that the homography constraint $\mathbf{x}_2 \sim H\mathbf{x}_1$ also holds for non-planar scenes undergoing pure rotation. In this case we simply have $H = K_2R_2K_1^{-1}$. Since there are 9 unknowns in the homograph matrix H (up to a scale), one can linearly solve for H from the homography constraint (8.7) from $N \geq 8$ point correspondences $\{(\mathbf{x}_{1i}, \mathbf{x}_{2i})\}_{i=1}^N$. Given some additional knowledge about the camera calibration K_1 and K_2 , one can solve for the camera motion (R_2, T_2) from F using linear methods.

8.1.4 The Trifocal Tensor

Let $\mathbf{x}_1 \leftrightarrow \mathbf{x}_2 \leftrightarrow \mathbf{x}_3$ be a point correspondence in three perspective views with 3×4 camera matrices

$$\Pi_1 = [K_1 \ 0], \quad \Pi_2 = [K_2 R_2 \ \mathbf{e}_2] \text{ and } \Pi_3 = [K_3 R_3 \ \mathbf{e}_3], \quad (8.8)$$

where $\mathbf{e}_2 \in \mathbb{P}^2$ and $\mathbf{e}_3 \in \mathbb{P}^2$ are the epipoles in the 2nd and 3rd views, respectively. Let ℓ_2 be any line passing through \mathbf{x}_2 , i.e., $\ell_2^\top \mathbf{x}_2 = 0$, and ℓ_3 be any line passing through \mathbf{x}_3 , i.e., $\ell_3^\top \mathbf{x}_3 = 0$. Then, the multiple view matrix [Ma et al., 2004]

$$\begin{bmatrix} \ell_2^\top K_2 R_2 \mathbf{x}_1 & \ell_2^\top \mathbf{e}_2 \\ \ell_3^\top K_3 R_3 \mathbf{x}_1 & \ell_3^\top \mathbf{e}_3 \end{bmatrix} \in \mathbb{R}^{2 \times 2} \quad (8.9)$$

must have rank 1, hence its determinant must be zero, i.e.,

$$\ell_2^\top (K_2 R_2 \mathbf{x}_1 \mathbf{e}_3^\top - \mathbf{e}_2 \mathbf{x}_1^\top R_3^\top K_3^\top) \ell_3 = 0. \quad (8.10)$$

This is the well-known point-line-line *trilinear constraint* among the three views [Hartley and Zisserman, 2000], which we will denote as

$$T(\mathbf{x}_1, \ell_2, \ell_3) = \sum_{p,q,r} T_{pqr} x_{1p} \ell_{2q} \ell_{3r} = 0 \quad (8.11)$$

where $T \in \mathbb{R}^{3 \times 3 \times 3}$ is the so-called *trifocal tensor*.

Computing the trifocal tensor

Since there are 27 unknowns in the trifocal tensor T (up to a scale factor), one can linearly solve for T from the trilinear constraint (8.11) given at least 26 point-line-line correspondences. However, if we are given point-point-point correspondences, then for each point in the 2nd view \mathbf{x}_2 , we can obtain two lines ℓ_{21} and ℓ_{22} passing through \mathbf{x}_2 , and similarly for the 3rd view. Therefore, each point correspondence gives $2^2 = 4$ linearly independent equations on T and we only need 7 point correspondences to linearly estimate T .

Computing epipoles, epipolar lines and camera matrices

Given the trifocal tensor T , it is well known how to compute the epipolar lines in the 2nd and 3rd views of a point \mathbf{x}_1 in the 1st view [Hartley and Zisserman, 2000]. Specifically, notice from (8.11) that the matrix

$$(K_2 R_2 \mathbf{x}_1 \mathbf{e}_3^\top - \mathbf{e}_2 \mathbf{x}_1^\top R_3^\top K_3^\top) \in \mathbb{R}^{3 \times 3} \quad (8.12)$$

has rank 2. In fact its left null-space is $\ell_2(\mathbf{x}_1) = \mathbf{e}_2 \times K_2 R_2 \mathbf{x}_1$ and its right null-space is $\ell_3(\mathbf{x}_1) = \mathbf{e}_3 \times K_3 R_3 \mathbf{x}_1$, i.e., the epipolar lines of \mathbf{x}_1 in the second and third views, respectively.

The epipoles in the second and third views \mathbf{e}_2 and \mathbf{e}_3 must lie in the epipolar lines in the second and third views, $\{\ell_2(\mathbf{x}_{1i})\}_{i=1}^N$ and $\{\ell_3(\mathbf{x}_{1i})\}_{i=1}^N$, respectively. Thus we can obtain the epipoles from

$$\mathbf{e}_2^\top [\ell_2(\mathbf{x}_{11}), \dots, \ell_2(\mathbf{x}_{1N})] = 0 \text{ and } \mathbf{e}_3^\top [\ell_3(\mathbf{x}_{11}), \dots, \ell_3(\mathbf{x}_{1N})] = 0. \quad (8.13)$$

Clearly, we only need 2 epipolar lines to determine the epipoles, hence we do not need to compute the epipolar lines for all points in the first view. However, it is better to use more than two lines in the presence of noise.

Finally, given T , \mathbf{e}_2 and \mathbf{e}_3 , one can solve for the camera matrices Π_1 , Π_2 and Π_3 using linear techniques [Hartley and Zisserman, 2000].

8.2 The Motion Segmentation Problem

Consider a moving camera with pose $g_0(t) \in SE(3)$ at time t observing a scene containing n moving objects with poses $\{g_j(t) \in SE(3)\}_{j=1}^n$ at time t . The motion of object j relative to the camera between the zeroth and f th frame is given by $(R_{fj}, T_{fj}) = g_j(f)g_0(f)^{-1}g_0(0)g_j(0)^{-1} \in SE(3)$. Let $\{\mathbf{X}_i \in \mathbb{R}^3\}_{i=1}^N$ be a collection of points in 3-D space lying in the n moving objects. The projection of a point \mathbf{X}_i lying in the j th object onto the f th camera frame is given by

$$\mathbf{x}_{fi} = \pi_f(R_{fj}\mathbf{X}_i + T_{fj}), \quad (8.14)$$

where $\pi_f : \mathbb{R}^3 \mapsto I$ is the camera projection model (orthographic, perspective, etc.). In this chapter, we will consider the following problem.

Problem 8.1 (3D Motion Segmentation from Point Correspondences)

Given N image points $\{\mathbf{x}_{fi}\}_{i=1,\dots,N}^{f=1,\dots,F}$ taken from F views of a motion sequence related by a collection of n 3-D motion models, estimate the number of motion models n and their parameters $\{\mathcal{M}_j\}_{j=1}^n$ without knowing which measurements correspond to which motion model.

In some cases, the camera model is such that the 3-D motion models are linear on the image measurements, thus Problem 8.1 is a direct application of GPCA. In other cases, the motion models are more complex, e.g., bilinear or trilinear. We develop extensions of GPCA to deal with such classes of segmentation problems.

8.3 Segmentation of Linear Motion Models

In this section, we consider the 3-D motion segmentation problem (Problem 8.1) in cases in which the projection model is such that the resulting 3-D motion model is *linear* in the image measurements. In particular, we consider the segmentation of rigid-body motions from point correspondences in multiple affine views and show that the motion segmentation problem boils down to segmenting low-dimensional subspaces of a high-dimensional space.

8.3.1 The Affine Motion Subspaces

Let $\{\mathbf{x}_{fi} \in \mathbb{R}^2\}_{i=1,\dots,N}^{f=1,\dots,F}$ be the images of N 3-D points $\{\mathbf{X}_i \in \mathbb{P}^3\}_{i=1}^N$ seen by a rigidly moving camera in F frames. Under the affine projection model, which generalizes orthographic, weak perspective, and paraperspective projection [Hartley and Zisserman, 2000], the images satisfy the equation

$$\mathbf{x}_{fi} = A_f \mathbf{X}_i, \quad (8.15)$$

where $A_f = K_f \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} [R_f \ T_f] \in \mathbb{R}^{2 \times 4}$ is the so-called *affine camera matrix* at frame f and depends on the pose of the camera relative to the world $(R_f, T_f) \in SE(3)$ and the internal camera calibration parameters $K_f \in SL(2)$.

When the set of points $\{\mathbf{X}_i\}_{i=1}^N$ all correspond to a single rigidly moving object, we can stack all the image measurements $\{\mathbf{x}_{fi}\}$ into a $2F \times N$ matrix W , which can be decomposed into a *motion matrix* M and *structure matrix* S as

$$W = MS$$

$$\begin{bmatrix} \mathbf{x}_{11} & \cdots & \mathbf{x}_{1N} \\ \vdots & & \vdots \\ \mathbf{x}_{F1} & \cdots & \mathbf{x}_{FN} \end{bmatrix}_{2F \times N} = \begin{bmatrix} A_1 \\ \vdots \\ A_F \end{bmatrix}_{2F \times 4} [\mathbf{X}_1 \ \cdots \ \mathbf{X}_N]_{4 \times N}. \quad (8.16)$$

It follows from equation (8.16) that $\text{rank}(W) \leq 4$. In addition, notice that the two rows of each A_f are linear combinations of the first two rows of a rotation matrix R_f , hence $\text{rank}(W) \geq \text{rank}(A_f) \geq 2$. Therefore, the 2-D point trajectories of 3-D points lying in a single rigidly moving object (the columns of the data matrix W) live in a subspace of \mathbb{R}^{2F} of dimension $d = 2, 3$ or 4 .¹

8.3.2 Segmentation of Motion Affine Subspaces

Consider now the case in which the set of points $\{\mathbf{X}_i\}_{i=1}^N$ corresponds to n rigid objects moving independently. It follows from our analysis in the previous section that, if we knew the segmentation of the feature points, then we could write the

¹This rank constraint was derived in [Tomasi and Kanade, 1992], and was used to propose the first multi-frame algorithm for estimating the motion of an affine camera observing a static scene.

measurement matrix as $W = [W_1, W_2, \dots, W_n]$, where the columns of $W_j \in \mathbb{R}^{2F \times N_j}$ are the N_j measurements associated with the j th moving object, so that $\sum_{j=1}^n N_j = N$. It also follows from our analysis in the previous section that each measurement matrix W_j satisfies

$$W_j = M_j S_j \quad j = 1, \dots, n, \quad (8.17)$$

where $M_j \in \mathbb{R}^{2F \times 4}$ and $S_j \in \mathbb{R}^{4 \times N_j}$ are, respectively, the motion and structure matrices associated with the j th moving object.

In reality, the segmentation of the feature points is *unknown*, and so the measurement matrix is given by $W = [W_1, W_2, \dots, W_n]P$, where $P \in \mathbb{R}^{N \times N}$ is an unknown permutation matrix. Nevertheless, the columns of W still live in a union of n motion subspaces $\{S_j \subset \mathbb{R}^{2F}\}_{j=1}^n$ of dimensions $d_j \in \{2, 3, 4\}$ for $j = 1, \dots, n$. Therefore, 3-D motion segmentation from point correspondences in multiple affine views is equivalent to segmenting subspaces of \mathbb{R}^{2F} of dimensions $d_1, \dots, d_n \leq d_{\max} = 4$. As discussed in Chapter 4, we can solve this problem by applying GPCA to the $2F$ -dimensional point trajectories projected onto a subspace of \mathbb{R}^{2F} of dimension $D = d_{\max} + 1 = 5$. That is, if $W = U\Sigma V^\top$ is the SVD of the data matrix, then we can solve the motion segmentation problem by applying GPCA (Algorithm 4.4) to the first 5 columns of V^\top .

Segmentation of Independent Motion Subspaces

It is worth noting that, under certain additional assumptions, the motion segmentation problem can be solved using a simpler algorithm that depends only on the SVD of the data matrix $W = U\Sigma V^\top$. For example, when the motion subspaces are fully dimensional, i.e., $\dim(S_j) = 4$, and fully independent, i.e., $\dim(S_j \cup S_k) = \dim(S_j) + \dim(S_k)$ or equivalently $S_j \cap S_k = \{0\}$, one can apply the Costeira and Kanade (CK) algorithm [Costeira and Kanade, 1998] to segment the motion subspaces. The CK algorithm is based on thresholding the entries of the so-called *shape interaction* matrix

$$Q = VV^\top, \quad (8.18)$$

which has the property that [Kanatani, 2001]

$$Q_{ij} = 0 \quad \text{if } i \text{ and } j \text{ correspond to different motions.} \quad (8.19)$$

This property has been the basis for most existing motion segmentation algorithms, such as [Costeira and Kanade, 1998, Kanatani, 2001, Kanatani, 2002, Kanatani and Matsunaga, 2002b, Wu et al., 2001].

In many applications, however, the motions need not be fully dimensional. In ground robot navigation, for example, the motion of each robot relative to the camera is constrained to be planar, which reduces the dimension of the motion subspaces to $d = 3$. In addition, the motion subspaces may be partially dependent, i.e., $\max\{\dim(S_j), \dim(S_k)\} < \dim(S_j \cup S_k) < \dim(S_j) + \dim(S_k)$ or equivalently $S_j \cap S_k \neq \{0\}$, $S_j \cap S_k \neq S_j$ and $S_j \cap S_k \neq S_k$, which happens for instance when two objects move with the same rotation but different translation relative to the camera. As reported in [Zelnik-Manor and Irani, 2003,

Kanatani and Sugaya, 2003, Vidal and Hartley, 2004], most existing motion segmentation algorithms show poor performance in the presence of degenerate or partially dependent motions, because they cannot deal with intersecting motion subspaces. The GPCA algorithm discussed in Chapter 4 does not impose any restriction on either the intersection or the dimensionality of the subspaces, hence it can deal with all the spectrum of affine motions: from two-dimensional and partially dependent to four-dimensional and fully independent.

8.4 Segmentation of Bilinear Motion Models

In this section, we consider the 3-D motion segmentation problem (Problem 8.1) in cases in which the projection model is such that the resulting 3-D motion model is *bilinear* in the image measurements. In particular, we consider the segmentation of rigid-body motions from point correspondences in two perspective views of nonplanar (Section 8.4.1) and planar (Section 8.4.2) scenes. In both cases, we show that the motion segmentation problem can be solved using extensions of GPCA to certain classes of bilinear surfaces.

8.4.1 Segmentation of Fundamental Matrices

In this subsection, we consider the problem of segmenting n 3-D rigid-body motions $\{(R_j, T_j) \in SE(3)\}_{j=1}^n$ from point correspondences in two perspective views. We assume that the 3-D scene is nonplanar and that the individual translations T_i are all nonzero. In this case, the motion of the objects relative to the camera between the two views can be modeled as a mixture of fundamental matrices $\{F_j\}_{j=1}^n$. In order for the problem to be well posed, we assume that the fundamental matrices are different from each other (up to a scale factor).

The multibody epipolar constraint and the multibody fundamental matrix

As shown in Section 8.1.2, if $(\mathbf{x}_1, \mathbf{x}_2)$ is an image pair associated with any of the n moving objects, then exists a fundamental matrix F_j such that $\mathbf{x}_2^\top F_j \mathbf{x}_1 = 0$. Therefore, the following *multibody epipolar constraint* must be satisfied by the number of independent motions n , the fundamental matrices $\{F_j\}_{j=1}^n$ and the image pair $(\mathbf{x}_1, \mathbf{x}_2)$, regardless of which motion is associated with the image pair

$$p_n(\mathbf{x}_1, \mathbf{x}_2) \doteq \prod_{j=1}^n (\mathbf{x}_2^\top F_j \mathbf{x}_1) = 0. \quad (8.20)$$

The multibody epipolar constraint (8.20) and the multibody brightness constancy constraint (MBCC) for affine motions (7.18) are both bi-homogeneous polynomials of degree n that factor as a product of n bilinear forms. Therefore, as shown in Theorem 7.4, the multibody epipolar constraint can be written in bilinear

form as

$$\prod_{j=1}^n (\mathbf{x}_2^\top F_j \mathbf{x}_1) = \nu_n(\mathbf{x}_2)^\top \mathcal{F} \nu_n(\mathbf{x}_1) = 0. \quad (8.21)$$

We call the matrix $\mathcal{F} \in \mathbb{R}^{M_n(3) \times M_n(3)}$ the *multibody fundamental matrix* as it is a natural generalization of the fundamental matrix to the case of multiple moving objects. Also, since equation (8.21) clearly resembles the bilinear form of the epipolar constraint for a single rigid-body motion, we will refer to both equations (8.20) and (8.21) as the *multibody epipolar constraint* from now on.

Although the multibody fundamental matrix \mathcal{F} seems a complicated mixture of all the individual fundamental matrices F_1, \dots, F_n , it is still possible to recover all the individual fundamental matrices from \mathcal{F} , under some mild conditions (e.g., the F_j 's are different). The rest of the section is devoted to providing a constructive proof for this. We first show how to recover n and \mathcal{F} from data, and then show how to recover $\{F_j\}_{j=1}^n$ from \mathcal{F} .

Estimating the number of motions n and the multibody fundamental matrix \mathcal{F}

Both the MBCC for affine motions (7.19) and the multibody epipolar constraint (8.21) are bilinear expressions on the embedded image measurements and linear expressions on the multibody motion parameters. Therefore, given $N \geq M_n(3)^2 - 1 \sim O(n^4)$ generic point correspondences $\{(\mathbf{x}_{1i}, \mathbf{x}_{2i})\}_{i=1}^N$, we can solve for the stack of the columns of the multibody fundamental matrix \mathcal{F} , $\text{vec}(\mathcal{F}) \in \mathbb{R}^{M_n(3)^2}$, from the linear system (see equation (7.20))

$$\mathbf{V}_n^{\mathcal{F}} \text{vec}(\mathcal{F}) \doteq [\nu_n(\mathbf{x}_{11}) \otimes \nu_n(\mathbf{x}_{21}), \dots, \nu_n(\mathbf{x}_{1N}) \otimes \nu_n(\mathbf{x}_{2N})]^\top \text{vec}(\mathcal{F}) = \mathbf{0}, \quad (8.22)$$

and for the number of independent motions n from (see Theorem 7.5)

$$n \doteq \min\{j : \text{rank}(\mathbf{V}_j^{\mathcal{F}}) = M_j(3)^2 - 1\}. \quad (8.23)$$

Factorization of the multibody fundamental matrix

Given the multibody fundamental matrix \mathcal{F} and the number of independent motions n , we now show how to recover the fundamental matrices and the segmentation of the image points. We first show that the gradients of the multibody epipolar constraint at the point correspondences lie in a collection of hyperplanes in \mathbb{R}^3 whose normal vectors are the n epipoles. Therefore, one can apply GPCA to these gradients in order to obtain the epipoles as well as the segmentation of the data. Once the data has been segmented, computing a fundamental matrix for each group is a linear problem.

- Given an image pair $(\mathbf{x}_1, \mathbf{x}_2)$ associated with the j th motion, its *epipolar line* in the second view (see Figure 8.3) is defined as $\ell_j \doteq F_j \mathbf{x}_1 \in \mathbb{P}^2$. Since $\mathbf{x}_2^\top F_j \mathbf{x}_1 = 0$, we can compute ℓ_j as the partial derivative of the multibody

epipolar constraint with respect to \mathbf{x}_2 evaluated at $(\mathbf{x}_1, \mathbf{x}_2)$, because

$$\frac{\partial}{\partial \mathbf{x}_2} (\nu_n(\mathbf{x}_2)^\top \mathcal{F} \nu_n(\mathbf{x}_1)) = \sum_{j=1}^n \prod_{k \neq j} (\mathbf{x}_2^\top F_k \mathbf{x}_1) (F_j \mathbf{x}_1) \quad (8.24)$$

$$= \prod_{k \neq j} (\mathbf{x}_2^\top F_k \mathbf{x}_1) (F_j \mathbf{x}_1) \sim \ell_j. \quad (8.25)$$

Therefore, given a set of point correspondences $\{(\mathbf{x}_{1i}, \mathbf{x}_{2i})\}_{i=1}^N$, we can compute its associated set of epipolar lines $\{\ell_i\}_{i=1}^N$ as the gradient of the multibody epipolar constraint at the correspondences.

2. Given an epipolar line ℓ associated with the j th motion, there exists an epipole \mathbf{e}_j such that $\mathbf{e}_j^\top \ell = \mathbf{e}_j^\top F_j \mathbf{x}_1 = 0$, because \mathbf{e}_j is the left null space of F_j . Therefore, the set of N epipolar lines can be interpreted as a set of points in \mathbb{R}^3 lying in n hyperplanes with normal vectors $\{\mathbf{e}_j\}_{j=1}^n$. We can apply the GPCA algorithm (Algorithm 4.4) to estimate the n epipoles $\{\mathbf{e}_j\}_{j=1}^n$ up to a scale factor. If the n epipoles are different,² we can immediately segment the data into n groups by assigning the image pair $(\mathbf{x}_{1i}, \mathbf{x}_{2i})$ to group j if

$$j = \arg \min_{k=1, \dots, n} (\mathbf{e}_k^\top \ell_i)^2 \quad (8.26)$$

3. Once the data has been clustered, solving for the fundamental matrix F_j from the epipolar constraint $\mathbf{x}_2^\top F_j \mathbf{x}_1 = 0$ is a linear problem of the form (see equation (8.22))

$$[w_{1j} \mathbf{x}_{11} \otimes \mathbf{x}_{21}, w_{2j} \dots, w_{Nj} \mathbf{x}_{1N} \otimes \mathbf{x}_{2N}]^\top \text{vec}(F_i) = \mathbf{0}, \quad (8.27)$$

where $w_{ij} = 1$ if the i th point belongs to the j th group, and $w_{ij} = 0$ otherwise.

Algorithm 8.1 summarizes the algorithm for segmenting n fundamental matrices. Table 8.1 gives the minimum number of point correspondences required by the algorithm as a function of the number of motions.

8.4.2 Segmentation of Homography Matrices

The motion segmentation scheme described in the previous subsection assumes that the displacement of each object between the two views relative to the camera is nonzero, i.e., $T \neq 0$, otherwise the individual fundamental matrices $F = \widehat{T}R$ would be zero. Furthermore, it also requires that the 3-D points be in general configuration, otherwise one cannot uniquely recover each fundamental matrix from

²This 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 [Vidal et al., 2005].

Algorithm 8.1 (Segmentation of Fundamental Matrices).

Given two perspective views $\{(\mathbf{x}_{1i}, \mathbf{x}_{2i})\}_{i=1}^N$ of a set of N 3-D points undergoing n different rigid-body motions, recover the number of independent motions n , the fundamental matrix F_j associated with each motion, and the motion model associated with each image pair as follows:

1. **Number of motions.** Form the embedded data matrix of degree $j \geq 1$, $\mathbf{V}_j^{\mathcal{F}} \in \mathbb{R}^{N \times M_j(3)^2}$, as in (8.22). Compute the number of independent motions n from (8.23) or else from

$$n = \arg \min_{j \geq 1} \frac{\sigma_{M_j(3)^2}^2(\mathbf{V}_j^{\mathcal{F}})}{\sum_{k=1}^{M_j(3)^2-1} \sigma_k^2(\mathbf{V}_j^{\mathcal{F}})} + \mu M_j(3)^2. \quad (8.28)$$

2. **Multibody fundamental matrix.** Compute the multibody fundamental matrix \mathcal{F} as the least-squares solution to the linear system $\mathbf{V}_n^{\mathcal{F}} \text{vec}(\mathcal{F}) = \mathbf{0}$ in (8.22), where $\mathbf{V}_n^{\mathcal{F}}$ is computed using the Veronese map ν_n of degree n .
3. **Epipolar lines.** Compute the epipolar lines $\{\ell_i\}_{i=1}^N$ in the second view associated with each image pair $\{(\mathbf{x}_{1i}, \mathbf{x}_{2i})\}_{i=1}^N$ as the gradient of the multibody epipolar constraint with respect to \mathbf{x}_2 evaluated at each image pair.
4. **Epipoles.** Apply GPCA to the epipolar lines $\{\ell_i\}_{i=1}^N$ to obtain the individual epipoles $\{e_j\}_{j=1}^n$.
5. **Feature segmentation.** Assign image pair $(\mathbf{x}_{1i}, \mathbf{x}_{2i})$ to motion $j = \arg \min_{k=1, \dots, n} (e_k^T \ell_i)^2$.
6. **Fundamental matrices.** Obtain the individual fundamental matrices $\{F_j\}_{j=1}^n$ by applying the eight-point algorithm to each group.

its epipolar constraint. The latter case occurs, for example, in the case of planar structures, i.e., when the 3-D points lie in a plane [Hartley and Zisserman, 2000].

Both in the case of purely rotating objects (relative to the camera) or in the case of a planar 3-D structure, the motion (R, T) between the two views $\mathbf{x}_1 \in \mathbb{P}^2$ and $\mathbf{x}_2 \in \mathbb{P}^2$ is described by a homography matrix $H \in \mathbb{R}^{3 \times 3}$. If $N \in \mathbb{R}^3$ is the (unit) normal to the plane and d is the distance from the plane to the origin, the homography matrix $H = R + \frac{1}{d}TN^\top$ is such that [Hartley and Zisserman, 2000]

$$\mathbf{x}_2 \sim H\mathbf{x}_1 = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \mathbf{x}_1. \quad (8.29)$$

When the camera calibration $K \in \mathbb{R}^{3 \times 3}$ is also unknown, the homography matrix is written as $H = K(R + \frac{1}{d}TN^\top)K^{-1}$.

The multibody homography

Consider now a scene that can be modeled with n different homographies $\{H_j\}_{j=1}^n$. Note that the n homographies do *not* necessarily correspond to n different rigid-body motions, as one rigidly moving object could consist of two or more planes whose motion is described by two or more homographies. Therefore, the n homographies can represent anything from 1 up to n rigid-body motions.

It is evident from the form of equation (8.29) that in order to eliminate the segmentation of the data we cannot take the product of all the equations, as we did with the epipolar constraints, because in this case we have two linearly independent equations per image pair. In Chapter 4 we dealt with this issue by using multiple polynomials to represent multiple subspaces of co-dimension more than one. In the case of homographies, one can avoid using multiple polynomials by exploiting properties of the cross product in \mathbb{R}^3 , as we show now.

We start by noticing that if ℓ is a line passing through \mathbf{x}_2 , i.e., ℓ is such that $\ell^\top \mathbf{x}_2 = 0$, then it follows from (8.29) that there is a motion i such that $\ell^\top H_j \mathbf{x}_1 = 0$. Therefore, the following *multibody homography constraint* must hold

$$p_n(\mathbf{x}_1, \ell) = \prod_{j=1}^n (\ell^\top H_j \mathbf{x}_1) = \nu_n(\ell)^\top \mathcal{H} \nu_n(\mathbf{x}_1) = 0. \quad (8.30)$$

We call the matrix $\mathcal{H} \in \mathbb{R}^{M_n(3) \times M_n(3)}$ the *multibody homography*. Notice that \mathcal{H} plays the analogous role of the multibody affine matrix \mathcal{A} , or the multibody fundamental matrix \mathcal{F} .

Computing the number of homographies n and the multibody homography \mathcal{H}

In order to compute n and \mathcal{H} from a given a set of point P correspondences $\{(\mathbf{x}_{1p}, \mathbf{x}_{2p})\}_{p=1}^P$, notice that given an image pair $(\mathbf{x}_1, \mathbf{x}_2)$, we can generate two linearly independent lines ℓ_1 and ℓ_2 passing through \mathbf{x}_2 . This may lead us to conclude that each correspondence gives two linearly independent constraints on the entries of \mathcal{H} . In reality, each correspondence gives $n + 1$ constraints on \mathcal{H} .

To see this, notice that equation (8.30) must hold for any line passing through \mathbf{x}_2 . Since the family of lines $\alpha\ell_1 + \ell_2$ passes through \mathbf{x}_2 for all $\alpha \in \mathbb{R}$, we have

$$q(\alpha) = \prod_{j=1}^n ((\alpha\ell_1 + \ell_2)^\top H_j \mathbf{x}_1) = \nu_n(\alpha\ell_1 + \ell_2)^\top \mathcal{H} \nu_n(\mathbf{x}_1) = 0. \quad (8.31)$$

One can show that (see Exercise 8.2)

$$\nu_n(\alpha\ell_1 + \ell_2) = \sum_{j=0}^n \alpha^j f_j(\ell_1, \ell_2), \quad (8.32)$$

where $f_j(\ell_1, \ell_2) \in \mathbb{R}^{M_n(3)}$ a polynomial of degree j in ℓ_1 and $(n - j)$ in ℓ_2 for $j = 0, \dots, n$. Therefore, $q(\alpha)$ is a polynomial of degree n in α such that $q(\alpha) = 0$ for all α , and so each one of its $n + 1$ coefficients must be zero, i.e.,

$$f_j(\ell_1, \ell_2)^\top \mathcal{H} \nu_n(\mathbf{x}_1) = 0, \quad j = 0, \dots, n. \quad (8.33)$$

This gives $n + 1$ constraints per point correspondence on the entries of \mathcal{H} . Therefore, given $P \geq \frac{M_n(3)^3 - 1}{n+1} \sim O(n^3)$ generic point correspondences $\{(\mathbf{x}_{1p}, \mathbf{x}_{2p})\}_{p=1}^P$, we can solve for the stack of the columns of the multibody homography \mathcal{H} , $\mathbf{h} \in \mathbb{R}^{M_n(3)^2}$, from the linear system

$$\mathbf{V}_n^{\mathcal{H}} \mathbf{h} = \mathbf{0}, \quad (8.34)$$

where the i th row of $\mathbf{V}_n^{\mathcal{H}} \in \mathbb{R}^{P(n+1) \times M_n(3)^2}$ is given by $\nu_n(\mathbf{x}_1) \otimes f_j(\ell_1, \ell_2)$.

Finally, similarly to (7.22) and (8.23), the number of homographies is given by

$$n \doteq \min\{j : \text{rank}(\mathbf{V}_j^{\mathcal{H}}) = M_j(3)^2 - 1\}. \quad (8.35)$$

Factorization of the multibody homography

Once \mathcal{H} is known, computing the homographies $\{H_j\}_{j=1}^n$ is equivalent to factorizing the multibody homography constraint (8.30) into a product of n bilinear factors. In general, solving such a factorization problem is difficult, unless some structure about the matrices H_j is known. In the case of fundamental matrices discussed in Section 8.4.1, we exploited the fact that epipoles are the left null spaces of the fundamental matrices. In the case of affine matrices discussed in Section 7.4, we exploited the fact that the 3rd row of each affine matrix is known.

Unfortunately, homographies are usually full rank and none of their rows are known. Hence, the factorization of \mathcal{H} is more challenging than the factorization of the multibody affine matrix \mathcal{A} or the multibody fundamental matrix \mathcal{F} . In fact, we will show that the factorization of \mathcal{H} requires a combination of the methods for factorizing \mathcal{A} and \mathcal{F} , according to the following four steps:

1. Compute derivatives of $p_n(\mathbf{x}_1, \ell)$ with respect to \mathbf{x}_1 to obtain linear combinations of the rows of each H_j .
2. Obtain three vectors orthogonal to the three pairs of rows of each H_j by solving three hyperplane clustering problems.
3. Obtain the rows of each H_j up to a scale factor from the cross products of these vectors.
4. Solve linearly for the unknown scales of the rows of H_j from the homography constraint.

For step 1, notice from (7.24) that if the image pair $(\mathbf{x}_1, \mathbf{x}_2)$ is associated with the i th motion and ℓ is any line passing through \mathbf{x}_2 , then the derivative of $p_n(\mathbf{x}_1, \ell)$ with respect to \mathbf{x}_1 at (\mathbf{x}_1, ℓ) gives $\ell^\top H_j$. Thus, by properly choosing ℓ we can get different linear combinations of the rows of $H_j = [\mathbf{h}_{j1} \mathbf{h}_{j2} \mathbf{h}_{j3}]^\top$. If we choose $\ell_{12} = (y_2, -x_2, 0)$, $\ell_{23} = (0, 1, -y_2)$ and $\ell_{31} = (1, 0, -x_2)$ as three lines passing through $\mathbf{x}_2 = (x_2, y_2, 1)$, we can compute the vectors

$$\mathbf{g}_{12} \sim y_2 \mathbf{h}_{j1} - x_2 \mathbf{h}_{j2}, \quad \mathbf{g}_{23} \sim \mathbf{h}_{j2} - y_2 \mathbf{h}_{j3} \quad \text{and} \quad \mathbf{g}_{31} \sim \mathbf{h}_{j1} - x_1 \mathbf{h}_{j3} \quad (8.36)$$

from the derivatives of p_n at $(\mathbf{x}_1, \ell_{12})$, $(\mathbf{x}_1, \ell_{23})$ and $(\mathbf{x}_1, \ell_{31})$, respectively.

For step 2, notice that \mathbf{g}_{12} lives in the plane spanned by \mathbf{h}_{j1} and \mathbf{h}_{j2} , whose normal vector is $\mathbf{b}_{12j} = \mathbf{h}_{j1} \times \mathbf{h}_{j2}$. Therefore, if we evaluate \mathbf{g}_{12} at all the given point correspondences, we obtain a set of N vectors $\{\mathbf{g}_{12i}\}_{i=1}^N$ lying in a union of n planes with normal vectors $\{\mathbf{b}_{12j}\}_{j=1}^n$. Similarly, the vectors $\{\mathbf{g}_{23i}\}_{i=1}^N$ and $\{\mathbf{g}_{31i}\}_{i=1}^N$ lie in a union of n planes with normal vectors $\{\mathbf{b}_{23j}\}_{j=1}^n$ and $\{\mathbf{b}_{31j}\}_{j=1}^n$, respectively. In principle we can obtain the vectors $\{\mathbf{b}_{12j}\}_{j=1}^n$, $\{\mathbf{b}_{23j}\}_{j=1}^n$ and $\{\mathbf{b}_{31j}\}_{j=1}^n$ by applying the GPCA algorithm (Algorithm 4.4) to each one of the three sets of points $\{\mathbf{g}_{12i}\}$, $\{\mathbf{g}_{23i}\}$ and $\{\mathbf{g}_{31i}\}$ separately. However, if we do so we would not know which \mathbf{b}_{12j} correspond to which \mathbf{b}_{13j} . Exercise 4.1 shows how to resolve this difficulty by exploiting the fact that the correspondences among the data points $\{\mathbf{g}_{12i}\}$, $\{\mathbf{g}_{23i}\}$ and $\{\mathbf{g}_{31i}\}$ are known, because these three vectors are computed as a triplet associated with the same point correspondence $(\mathbf{x}_{1i}, \mathbf{x}_{2i})$. The main idea is to compute three polynomials p_{12} , p_{23} and p_{31} fitting the points $\{\mathbf{g}_{12i}\}$, $\{\mathbf{g}_{23i}\}$ and $\{\mathbf{g}_{31i}\}$, respectively, and then obtain the normal vector for the j th group from the gradients of these polynomials. In order for the normal vectors \mathbf{b}_{12i} , \mathbf{b}_{23i} and \mathbf{b}_{31i} to correspond, we can choose the points at which the gradients by minimizing the sum of the squared distances to all hyperplanes, i.e., we compute the normal vectors as

$$\mathbf{b}_{12j} \sim \nabla p_{12}(\mathbf{g}_{12i_j}), \quad \mathbf{b}_{23j} \sim \nabla p_{23}(\mathbf{g}_{23i_j}) \text{ and } \mathbf{b}_{31j} \sim \nabla p_{31}(\mathbf{g}_{31i_j}), \quad (8.37)$$

where

$$i_j = \arg \min_{j=1, \dots, N} \frac{\frac{|p_{12}(\mathbf{g}_{12j})|}{\|\nabla p_{12}(\mathbf{g}_{12j})\|}}{\prod_{i=1}^{k=n} |\mathbf{b}_{12k}^T \mathbf{g}_{12j}|} + \frac{\frac{|p_{23}(\mathbf{g}_{23j})|}{\|\nabla p_{23}(\mathbf{g}_{23j})\|}}{\prod_{i=1}^{k=n} |\mathbf{b}_{23k}^T \mathbf{g}_{23j}|} + \frac{\frac{|p_{31}(\mathbf{g}_{31j})|}{\|\nabla p_{31}(\mathbf{g}_{31j})\|}}{\prod_{i=1}^{k=n} |\mathbf{b}_{31k}^T \mathbf{g}_{31j}|} \quad (8.38)$$

For step 3, given $\mathbf{b}_{12} = \mathbf{h}_{i1} \times \mathbf{h}_{i2}$, $\mathbf{b}_{23} = \mathbf{h}_{i2} \times \mathbf{h}_{i3}$ and $\mathbf{b}_{31} = \mathbf{h}_{i3} \times \mathbf{h}_{i1}$, we can immediately obtain the rows of H_j up to a scale factor as

$$\mathbf{h}_{i1} \sim \tilde{\mathbf{h}}_{i1} \doteq \mathbf{b}_{12} \times \mathbf{b}_{31}, \quad \mathbf{h}_{i2} \sim \tilde{\mathbf{h}}_{i2} \doteq \mathbf{b}_{23} \times \mathbf{b}_{12}, \quad \mathbf{h}_{i3} \sim \tilde{\mathbf{h}}_{i3} \doteq \mathbf{b}_{31} \times \mathbf{b}_{23}. \quad (8.39)$$

For step 4, we know that $\mathbf{x}_2 \sim H_j \mathbf{x}_1$. Therefore, we can obtain the n homographies as

$$H_j = \begin{bmatrix} \frac{x_2}{\tilde{\mathbf{h}}_{j1}^T \mathbf{x}_1} \tilde{\mathbf{h}}_{j1} & \frac{y_2}{\tilde{\mathbf{h}}_{j2}^T \mathbf{x}_1} \tilde{\mathbf{h}}_{j2} & \frac{1}{\tilde{\mathbf{h}}_{j3}^T \mathbf{x}_1} \tilde{\mathbf{h}}_{j3} \end{bmatrix}^\top \quad j = 1, \dots, n. \quad (8.40)$$

Algorithm 8.2 summarizes the algorithm for segmenting homography matrices. Table 8.1 gives the minimum number of point correspondences required by the algorithm as a function of the number of motions.

8.5 Segmentation of Trilinear Motion Models

In this section, we consider the problem of segmenting n 3-D rigid-body motions $\{(R_j, T_j) \in SE(3)\}_{j=1}^n$ from point correspondences in three perspective views. As shown in Section 8.1.4, in this case the motion of the each object relative to the camera among the three views can be modeled as a mixture of trifocal tensors

Algorithm 8.2 (Segmentation of Homography Matrices).

Given two perspective views $\{(\mathbf{x}_{1i}, \mathbf{x}_{2i})\}_{i=1}^N$ of a set of N 3-D points whose motion can be modeled with n homography matrices $\{H_j\}_{j=1}^n$, recover the number of independent motions n , the homography matrix $\{H_j\}_{j=1}^n$ associated with each motion, and the motion model associated with each image pair as follows:

1. **Number of motions.** Compute two lines (ℓ_{21i}, ℓ_{22i}) passing through \mathbf{x}_{2i} . Form the embedded data matrix of degree $j \geq 1$, $\mathbf{V}_i^{\mathcal{H}} \in \mathbb{R}^{N(j+1) \times M_n(3)}$, as in (8.34). Compute the number of motions from (8.35), or else from

$$n = \arg \min_{j \geq 1} \frac{\sigma_{M_j(3)^2}^2(\mathbf{V}_i^{\mathcal{H}})}{\sum_{k=1}^{M_j(3)^2-1} \sigma_k^2(\mathbf{V}_i^{\mathcal{H}})} + \mu M_j(3)^2. \quad (8.41)$$

2. **Multibody homography matrix.** Compute the multibody homography matrix \mathcal{H} as the least-squares solution to the linear system $\mathbf{V}_n^{\mathcal{H}} \text{vec}(\mathcal{H}) = \mathbf{0}$ in (8.34), and let

$$p_n(\mathbf{x}_1, \boldsymbol{\ell}) = \nu_n(\boldsymbol{\ell})^T \mathcal{H} \nu_n(\mathbf{x}_1).$$

3. **Homography matrices.**

- (a) For all $i = 1, \dots, N$, let $\ell_{12i} = (y_{2i}, -x_{2i}, 0)^T$, $\ell_{23i} = (0, 1, y_{2i})^T$ and $\ell_{31i} = (1, 0, -x_{2i})^T$ be three lines passing through $\mathbf{x}_{2i} = (x_{2i}, y_{2i}, 1)^T$. Compute a linear combination of rows 1 & 2, 2 & 3, and 3 & 1 of the homography matrix at each point correspondence as

$$\mathbf{g}_{12i} = \frac{\partial p_n}{\partial \mathbf{x}_1}(\mathbf{x}_{1i}, \ell_{12i}); \quad \mathbf{g}_{23i} = \frac{\partial p_n}{\partial \mathbf{x}_1}(\mathbf{x}_{1i}, \ell_{23i}); \quad \mathbf{g}_{31i} = \frac{\partial p_n}{\partial \mathbf{x}_1}(\mathbf{x}_{1i}, \ell_{31i})$$

- (b) Solve for the coefficients \mathbf{c}_{jk} of the polynomials $p_{12}(\mathbf{g}) = \mathbf{c}_{12}^T \nu_n(\mathbf{g})$, $p_{23}(\mathbf{g}) = \mathbf{c}_{23}^T \nu_n(\mathbf{g})$ and $p_{31}(\mathbf{g}) = \mathbf{c}_{31}^T \nu_n(\mathbf{g})$, from the linear system

$$[\nu_n(\mathbf{g}_{jk1}), \nu_n(\mathbf{g}_{jk2}), \dots, \nu_n(\mathbf{g}_{jkN})]^T \mathbf{c}_{jk} = \mathbf{0}$$

- (c) Compute the homography matrices from the cross products of the gradients of p_{12} , p_{23} and p_{31} as follows:

for $j = n : 1$

$$i_j = \arg \min_{j=1, \dots, N} \frac{|p_{12}(\mathbf{g}_{12j})|}{\|\nabla p_{12}(\mathbf{g}_{12j})\|} + \frac{|p_{23}(\mathbf{g}_{23j})|}{\|\nabla p_{23}(\mathbf{g}_{23j})\|} + \frac{|p_{31}(\mathbf{g}_{31j})|}{\|\nabla p_{31}(\mathbf{g}_{31j})\|};$$

$$\mathbf{b}_{12j} = \nabla p_{12}(\mathbf{g}_{12j}); \quad \mathbf{b}_{23j} = \nabla p_{23}(\mathbf{g}_{23j}); \quad \mathbf{b}_{31j} = \nabla p_{31}(\mathbf{g}_{31j});$$

$$H_j = \begin{bmatrix} \frac{x_{2i_j}(\mathbf{b}_{12j} \times \mathbf{b}_{31j})}{\det([\mathbf{b}_{12j} \ \mathbf{b}_{31j} \ \mathbf{x}_{1i_j}])} & \frac{y_{2i_j}(\mathbf{b}_{23j} \times \mathbf{b}_{12j})}{\det([\mathbf{b}_{23j} \ \mathbf{b}_{12j} \ \mathbf{x}_{1i_j}])} & \frac{(\mathbf{b}_{31j} \times \mathbf{b}_{23j})}{\det([\mathbf{b}_{31j} \ \mathbf{b}_{23j} \ \mathbf{x}_{1i_j}])} \end{bmatrix}^T.$$

end

4. **Feature segmentation.** Assign the image pair $(\mathbf{x}_{1i}, \mathbf{x}_{2i})$ to group j if $j = \arg \min_{k=1, \dots, n} \|\mathbf{x}_2 - \frac{H_k \mathbf{x}_1}{e_3^T H_k \mathbf{x}_1}\|^2$.

$\{T_j \in \mathbb{R}^{3 \times 3 \times 3}\}_{j=1}^n$ relating a point, a line and a line in the first, second and third views. To avoid degenerate situations, we assume that the 3-D scene is nonplanar and that the trifocal tensors are different from each other (up to a scale factor).

8.5.1 The Multibody Trifocal Tensor

The multibody trilinear constraint and the multibody trifocal tensor

Let $\mathbf{x}_1 \leftrightarrow \mathbf{x}_2 \leftrightarrow \mathbf{x}_3$ be an arbitrary point correspondence. Then, there exists a trifocal tensor T_j that relates the point in the first view $\mathbf{x}_1 = (x_{11}, x_{12}, x_{13})^\top$, a line in the second view $\ell_2 = (\ell_{21}, \ell_{22}, \ell_{23})^\top$ passing through \mathbf{x}_2 and a line in the third view $\ell_3 = (\ell_{31}, \ell_{32}, \ell_{33})^\top$ passing through \mathbf{x}_3 via the *trilinear constraint*:

$$T_j(\mathbf{x}_1, \ell_2, \ell_3) = \sum_{p,q,r} T_{j,pqr} x_{1p} \ell_{2q} \ell_{3r} = 0. \quad (8.42)$$

Therefore, regardless of which motion is associated with the correspondence, the following constraint must be satisfied by the number of independent motions n , the trifocal tensors $\{T_j\}_{j=1}^n$ and the point-line-line correspondence $\mathbf{x}_1 \leftrightarrow \ell_2 \leftrightarrow \ell_3$

$$\prod_{j=1}^n T_j(\mathbf{x}_1, \ell_2, \ell_3) = 0. \quad (8.43)$$

This multibody constraint is a homogeneous polynomial of degree n in each of \mathbf{x}_1 , ℓ_2 or ℓ_3 . Thus, we can write it as a sum of monomials of degree n in each of \mathbf{x}_1 , ℓ_2 and ℓ_3 . By collecting the coefficients of these monomial in a 3-dimensional tensor $\mathcal{T} \in \mathbb{R}^{M_n(3) \times M_n(3) \times M_n(3)}$, we can write the constraint (8.43) as

$$\mathcal{T}(\nu_n(\mathbf{x}_1), \nu_n(\ell_2), \nu_n(\ell_3)) = 0. \quad (8.44)$$

We call the array \mathcal{T} the *multibody trifocal tensor* and equation (8.44) the *multibody trilinear constraint*, as they are natural generalizations of the trifocal tensor and the trilinear constraint, respectively, valid for $n = 1$.

Computing the number of motions and the multibody trifocal tensor

Notice that, although (8.44) has degree n in the entries of \mathbf{x}_1 , ℓ_2 and ℓ_3 , it is in fact *linear* in the entries of $\nu_n(\mathbf{x})$, $\nu_n(\ell_2)$ and $\nu_n(\ell_3)$. Hence, given a point-line-line correspondence $\mathbf{x}_1 \leftrightarrow \ell_2 \leftrightarrow \ell_3$, we can compute the entries of the vectors $\nu_n(\mathbf{x})$, $\nu_n(\ell_2)$ and $\nu_n(\ell_3)$ and use the multibody trilinear constraint (8.44) to obtain a linear relationship in the entries of \mathcal{T} . Therefore, we may estimate \mathcal{T} linearly from $M_n(3)^3 - 1 \sim O(n^6)$ point-line-line correspondences. That is 26 correspondences for one motion, 215 for two motions, 999 for three motions, etc.

Fortunately, as in the case of $n = 1$ motion, one may significantly reduce the data requirements by working with point-point-point correspondences $\mathbf{x}_1 \leftrightarrow \mathbf{x}_2 \leftrightarrow \mathbf{x}_3$. Since each point in the second view \mathbf{x}_2 gives two independent lines ℓ_{21} and ℓ_{22} passing through it and each point in the third view \mathbf{x}_3 gives two independent lines ℓ_{31} and ℓ_{32} passing through it, a naive calculation would give

$2^2 = 4$ constraints per point-point-point correspondence. However, due to the algebraic properties of the Veronese map, each correspondence provides in general $(n+1)^2$ independent constraints on the multibody trifocal tensor.

To see this, remember from Section 8.1.4 that the trilinear constraint is satisfied by all lines $\ell_2 = \alpha\ell_{21} + \ell_{22}$ and $\ell_3 = \beta\ell_{31} + \ell_{32}$ passing through x_2 and x_3 , respectively. Therefore, for all $\alpha \in \mathbb{R}$ and $\beta \in \mathbb{R}$ we must have

$$\prod_{j=1}^n T_j(x_1, \alpha\ell_{21} + \ell_{22}, \beta\ell_{31} + \ell_{32}) = 0. \quad (8.45)$$

The above equation, viewed as a function of α , is a polynomial of degree n , hence its $n+1$ coefficients must be zero. Each coefficient is in turn a polynomial of degree n in β , whose $n+1$ coefficients must be zero. Therefore, each point-point-point correspondence gives $(n+1)^2$ constraints on the multibody trifocal tensor \mathcal{T} , and we need only $(M_n(3)^3 - 1)/(n+1)^2 \sim O(n^4)$ point-point-point correspondences to estimate \mathcal{T} . That is 7, 24 and 63 correspondences for one, two and three motions, respectively. This reduction on the number of required correspondences represents a significant improvement, not only with respect to the case of point-line-line correspondences, as explained above, but also with respect to the case of two perspective views. As discussed in Section 8.4.1, one needs $M_n(3)^2 - 1$ point-point correspondences for linearly estimating the multibody fundamental matrix \mathcal{F} , i.e., 8, 35 and 99 correspondences for one, two and three motions, respectively, as shown in Table 8.1.

Given a correspondence $x_1 \leftrightarrow x_2 \leftrightarrow x_3$, we generate the $(n+1)^2$ linear equations in the entries of \mathcal{T} by choosing $\ell_{21}, \ell_{22}, \ell_{31}$ and ℓ_{32} passing through x_2 and x_3 , respectively, and then computing the coefficient of $\alpha^i \beta^j$ in (8.45). As shown in Exercise 8.3, these $(n+1)^2$ coefficients are given by

$$\begin{aligned} \mathcal{T}(\nu_n(x_1), f_j(\ell_{21}, \ell_{22}), f_k(\ell_{31}, \ell_{32})) \\ = (\nu_n(x_1) \otimes f_j(\ell_{21}, \ell_{22}) \otimes f_k(\ell_{31}, \ell_{32}))^\top \text{vec}(\mathcal{T}), \quad j, k = 1, \dots, n, \end{aligned}$$

where $\text{vec}(\mathcal{T}) \in \mathbb{R}^{M_n(3)^3}$ is the stack of all the entries of \mathcal{T} and f_j is defined in (8.32). Therefore, we can solve for \mathcal{T} from the linear system

$$\mathbf{V}_n^\mathcal{T} \text{vec}(\mathcal{T}) = \mathbf{0}, \quad (8.46)$$

where the rows of the matrix $\mathbf{V}_n^\mathcal{T} \in \mathbb{R}^{P(n+1)^2 \times M_n(3)^3}$ are of the form $\nu_n(x_{1i}) \otimes f_j(\ell_{21i}, \ell_{22i}) \otimes f_k(\ell_{31i}, \ell_{32i})$, for $j, k = 1, \dots, n$ and $i = 1, \dots, N$.

Finally, similarly to (7.22), (8.23) and (8.35), the number of trifocal tensors can be computed from

$$n \doteq \min\{j : \text{rank}(\mathbf{V}_j^\mathcal{T}) = M_j(3)^3 - 1\}. \quad (8.47)$$

8.5.2 Segmentation of Trifocal Tensors

Computing the epipolar lines

Given the trifocal tensor T , it is well known how to compute the epipolar lines $\ell_2(\mathbf{x}_1)$ and $\ell_3(\mathbf{x}_1)$ in the 2nd and 3rd views associated with a point \mathbf{x}_1 in the 1st view. For example, as shown in Section 8.1.4, the epipolar line in the second view $\ell_2(\mathbf{x}_1)$ must satisfy the relationship

$$\forall \ell_3 \in \mathbb{R}^3 \quad T(\mathbf{x}_1, \ell_2(\mathbf{x}_1), \ell_3) = 0. \quad (8.48)$$

In the case of multiple motions, we are faced with the more challenging problem of computing the epipolar lines $\ell_2(\mathbf{x}_1)$ and $\ell_3(\mathbf{x}_1)$ without knowing the individual trifocal tensors $\{T_j\}_{j=1}^n$ or the segmentation of the correspondences. The question is then how to compute such epipolar lines from the multibody trifocal tensor T . To this end, notice that with each point in the first view \mathbf{x}_1 we can associate n epipolar lines corresponding to the n motions between the 1st and 2nd views. If $\ell_2(\mathbf{x}_1)$ is an epipolar line of \mathbf{x}_1 according to the j th motion, then from equation (8.48) we have that for all $\ell_3 \in \mathbb{R}^3$, $T_j(\mathbf{x}_1, \ell_2(\mathbf{x}_1), \ell_3) = 0$. This implies that

$$\forall \ell_3 \in \mathbb{R}^3 \prod_{j=1}^n T_j(\mathbf{x}_1, \ell_2(\mathbf{x}_1), \ell_3) = T(\nu_n(\mathbf{x}_1), \nu_n(\ell_2(\mathbf{x}_1)), \nu_n(\ell_3)) = 0. \quad (8.49)$$

As this equation holds for *any* of the n epipolar lines, the question of determining *the* epipolar line of a point \mathbf{x}_1 is not well posed as such, because *the* epipolar line depends on which of the n motions the point \mathbf{x}_1 belongs to, which cannot be determined without additional information. We therefore pose the question a little differently, and suppose that we know the point \mathbf{x}_2 in the second view corresponding to \mathbf{x}_1 . Since the epipolar line $\ell_2(\mathbf{x}_1)$ must of course pass through \mathbf{x}_2 , we can parameterize it as

$$\ell_2(\mathbf{x}_1) = \alpha \ell_{21} + \ell_{22}, \quad (8.50)$$

where, as before, ℓ_{21} and ℓ_{22} are two different lines passing through \mathbf{x}_2 . Replacing (8.50) in equation (8.49) gives

$$\forall \ell_3 \in \mathbb{R}^3 \quad T(\nu_n(\mathbf{x}_1), \nu_n(\alpha \ell_{21} + \ell_{22}), \nu_n(\ell_3)) = 0. \quad (8.51)$$

As ℓ_3 ranges over all of \mathbb{R}^3 , this gives a total of up to $M_n(3)$ linearly independent equations. Each one of such equations is a polynomial of degree n in α . These polynomials must have a common root α^* for which all the polynomials vanish. The epipolar line of \mathbf{x}_1 in the second view is then $\ell_2(\mathbf{x}_1) = \alpha^* \ell_{21} + \ell_{22}$. The epipolar line of \mathbf{x}_1 in the third view can be obtained in an analogous fashion.

We may apply this process to all N correspondences $\{\mathbf{x}_{1i} \leftrightarrow \mathbf{x}_{2i} \leftrightarrow \mathbf{x}_{3i}\}_{i=1}^N$ to obtain the set of all N epipolar lines in the second and third views, $\{\ell_2(\mathbf{x}_{1i})\}_{i=1}^N$ and $\{\ell_3(\mathbf{x}_{1i})\}_{i=1}^N$, according to their individual motion models. Notice, again, that this is done from the multibody trifocal tensor T only, without knowing the individual trifocal tensors or the segmentation of the correspondences.

Computing the epipoles

As shown in Section 8.1.4, in the case of one rigid-body motion, the epipoles in the second and third views, \mathbf{e}_2 and \mathbf{e}_3 , can be computed from the epipolar lines in the second and third views, $\{\ell_2(\mathbf{x}_{1i})\}_{i=1}^N$ and $\{\ell_3(\mathbf{x}_{1i})\}_{i=1}^N$, respectively, by solving the linear systems

$$\mathbf{e}_2^\top [\ell_2(\mathbf{x}_{11}), \dots, \ell_2(\mathbf{x}_{1N})] = \mathbf{0}^\top \text{ and } \mathbf{e}_3^\top [\ell_3(\mathbf{x}_{11}), \dots, \ell_3(\mathbf{x}_{1N})] = \mathbf{0}^\top. \quad (8.52)$$

In the case of n motions there exist n epipole pairs, $\{(\mathbf{e}_{2j}, \mathbf{e}_{3j})\}_{j=1}^n$, where \mathbf{e}_{2j} and \mathbf{e}_{3j} are epipoles in the second and third views corresponding to the j th motion. Given a set of correspondences $\{\mathbf{x}_{2i} \leftrightarrow \mathbf{x}_{3i} \leftrightarrow \mathbf{x}_{3i}\}$ we may compute the multibody trifocal tensor \mathcal{T} and determine the epipolar lines $\ell_2(\mathbf{x}_{1i})$ and $\ell_3(\mathbf{x}_{1i})$ associated with each correspondence by the method described in the previous subsection. Then, for each pair of epipolar lines $(\ell_2(\mathbf{x}_{1i}), \ell_3(\mathbf{x}_{1i}))$ there exists an epipole pair $(\mathbf{e}_{2j}, \mathbf{e}_{3j})$ such that

$$\mathbf{e}_{2j}^\top \ell_2(\mathbf{x}_{1i}) = 0 \quad \text{and} \quad \mathbf{e}_{3j}^\top \ell_3(\mathbf{x}_{1i}) = 0. \quad (8.53)$$

Therefore, the problem of finding the epipole pairs $\{(\mathbf{e}_{2j}, \mathbf{e}_{3j})\}_{j=1}^n$ is equivalent to one of segmenting two sets of points lying in two collections of hyperplanes whose normal vectors are the epipole pairs.

Exercise 4.1 provides a solution to this problem based on a simple extension of the GPCA (Algorithm 4.4). The first step is to fit two polynomials,

$$\begin{aligned} p_2(\ell_2) &= \prod_{j=1}^n (\mathbf{e}_{2j}^\top \ell_2) = \mathbf{c}_2^\top \nu_n(\ell_2) = 0, \\ p_3(\ell_3) &= \prod_{j=1}^n (\mathbf{e}_{3j}^\top \ell_3) = \mathbf{c}_3^\top \nu_n(\ell_3) = 0, \end{aligned} \quad (8.54)$$

to the epipolar lines $\{\ell_2(\mathbf{x}_{1i})\}_{i=1}^N$ and $\{\ell_3(\mathbf{x}_{1i})\}_{i=1}^N$, respectively. Similarly to (8.52), we may find the coefficients of p_2 and p_3 by solving the linear systems

$$\begin{aligned} \mathbf{c}_2^\top [\nu_n(\ell_2(\mathbf{x}_{11})), \dots, \nu_n(\ell_2(\mathbf{x}_{1N}))] &= \mathbf{0}^\top, \\ \mathbf{c}_3^\top [\nu_n(\ell_3(\mathbf{x}_{11})), \dots, \nu_n(\ell_3(\mathbf{x}_{1N}))] &= \mathbf{0}^\top. \end{aligned} \quad (8.55)$$

The second step is to compute the epipoles as the gradients of these polynomials at a collection of n pairs of epipolar lines $\{(\ell_{2j}, \ell_{3j})\}_{j=1}^n$ passing through each one of the epipole pairs, i.e.,

$$\mathbf{e}_{2j} \sim \nabla p_2(\ell_{2j}) \quad \text{and} \quad \mathbf{e}_{3j} \sim \nabla p_3(\ell_{3j}), \quad j = 1, \dots, n. \quad (8.56)$$

The pairs of epipolar lines $\{(\ell_{2j}, \ell_{3j})\}_{j=1}^n$ are chosen as the n pairs of epipolar lines in $\{(\ell_2(\mathbf{x}_{1i}), \ell_3(\mathbf{x}_{1i}))\}_{i=1}^N$ that minimize a certain distance to their respective epipoles. More specifically, for $j = n, \dots, 2, 1$ set $\ell_{2j} = \ell_2(\mathbf{x}_{1i_j})$ and $\ell_{2j} = \ell_2(\mathbf{x}_{1i_j})$, where

$$i_j = \arg \min_{i=1, \dots, N} \frac{\frac{p_2(\ell_2(\mathbf{x}_{1i}))^2}{\|\nabla p_2(\ell_2(\mathbf{x}_{1i}))\|^2}}{\prod_{k=j+1}^n (\mathbf{e}_{2k}^\top \ell_2(\mathbf{x}_{1i}))^2} + \frac{\frac{p_3(\ell_3(\mathbf{x}_{1i}))^2}{\|\nabla p_3(\ell_3(\mathbf{x}_{1i}))\|^2}}{\prod_{k=j+1}^n (\mathbf{e}_{3k}^\top \ell_3(\mathbf{x}_{1i}))^2}. \quad (8.57)$$

Computing the trifocal tensors

Once the n epipole pairs $\{\mathbf{e}_{2j}, \mathbf{e}_{3j}\}_{j=1}^n$ have been computed, we can segment the data into n groups by assigning the image pair $(\mathbf{x}_{1i}, \mathbf{x}_{2i})$ to group j if

$$j = \arg \min_{k=1, \dots, n} (\mathbf{e}_{2k}^\top \boldsymbol{\ell}_2(\mathbf{x}_{1i}))^2 + (\mathbf{e}_{3k}^\top \boldsymbol{\ell}_3(\mathbf{x}_{1i}))^2 \quad (8.58)$$

provided that the n epipoles pairs are different. Given the segmentation of the correspondences, one may obtain trifocal tensors, fundamental matrices and camera matrices using the algebraic methods described in Section 8.1.4.

Algorithm summary

Algorithm 8.3 summarizes the main steps of the algorithm for segmenting trifocal tensors described in this section. Table 8.1 gives the minimum number of point correspondences required by the algorithm as a function of the number of motions.

Table 8.1. Number of correspondences required to linearly solve for the different multibody motion models.

	Correspondence	1	2	3	4
Multibody fundamental matrix \mathcal{F}	point-point	8	35	99	
Multibody homography matrix \mathcal{H}	point-point	4	12	25	
Multibody trifocal tensor \mathcal{T}	point-point-point	7	24	63	
Multibody trifocal tensor \mathcal{T}	point-line-line	26	215	999	

8.6 Experimental Results

8.6.1 Segmentation of Affine Motion Subspaces

We tested the GPCA algorithm on three different sequences shown in Figure 8.4. The data for these sequences consist of point correspondences in multiple views, which are available at <http://www.suri.it.okayama-u.ac.jp/data.html>. Sequence A consists of 30 frames of an outdoor sequence taken by a moving camera tracking a car moving in front of a parking lot. Sequence B consists of 17 frames of an outdoor sequence taken by a moving camera tracking a car moving in front of a building. Sequence C consists of 100 frames of an indoor sequence taken by a moving camera tracking a person moving his head.

For all sequences, we first projected the point trajectories onto a 5-dimensional subspace of \mathbb{R}^{2F} , where F is the number of frames in the sequence. We assumed that the motion subspaces are 4-dimensional, so that the motion segmentation problem is reduced to segmenting 4-dimensional hyperplanes in \mathbb{R}^5 . The number of motions is correctly estimated from (4.19) as $n = 2$. We used the criterion (2.14) with $\kappa \in [2, 20] 10^{-7}$ to determine the rank of the embedded data matrix.

Algorithm 8.3 (Segmentation of Trifocal Tensors).

Given a set of points $\{(\mathbf{x}_{1i}, \mathbf{x}_{2i}, \mathbf{x}_{3i})\}_{i=1}^N$ corresponding to N points undergoing n different rigid-body motions relative to a moving perspective camera, recover the number of independent motions n , the trifocal tensors $\{T_j\}_{j=1}^n$ associated with each motion, and the motion associated with each correspondence as follows:

1. **Number of motions.** Compute two lines (ℓ_{21i}, ℓ_{22i}) passing through \mathbf{x}_{2i} , and two lines (ℓ_{31i}, ℓ_{32i}) passing through \mathbf{x}_{3i} . Form the embedded data matrix of degree $j = 1, \dots, n$, $\mathbf{V}_j^T \in \mathbb{R}^{N(j+1)^2 \times M_j(3)^3}$, as defined in (8.46). Compute the number of independent motions n from

$$n = \arg \min_{j \geq 1} \frac{\sigma_{M_j(3)^3}^2(\mathbf{V}_j^T)}{\sum_{k=1}^{M_j(3)^3-1} \sigma_k^2(\mathbf{V}_j^T)} + \mu M_j(3)^3. \quad (8.59)$$

2. **Multibody trifocal tensor.** Compute the multibody trifocal tensor \mathcal{T} as the least-squares solution to the linear system $\mathbf{V}_n^T \text{vec}(\mathcal{T}) = \mathbf{0}$ in (8.46).
3. **Epipolar lines.** For all $i = 1, \dots, N$, compute the epipolar lines of \mathbf{x}_{1i} in the second and third views, $\ell_2(\mathbf{x}_{1i})$ and $\ell_3(\mathbf{x}_{1i})$, as follows
 - (a) Let $q_k(\alpha) = \sum_{j=0}^n \mathcal{T}(\nu_n(\mathbf{x}_{1i}), f_j(\ell_{21i}, \ell_{22i}), e_k) \alpha^j$, for all $k = 1, \dots, M_n(3)$. Compute the common root α^* of these $M_n(3)$ polynomials as the value of α that minimizes $q(\alpha) = \sum_{k=1}^{M_n(3)} q_k(\alpha)^2$. The epipolar line of \mathbf{x}_{1i} in the second view is $\ell_2(\mathbf{x}_{1i}) = \alpha^* \ell_{21i} + \ell_{22i}$.
 - (b) Compute the epipolar line of \mathbf{x}_{1i} in the third view as $\ell_3(\mathbf{x}_{1i}) = \beta^* \ell_{31i} + \ell_{32i}$, where β^* is the common roots of the polynomials $q_k(\beta) = \sum_{j=0}^n \mathcal{T}(\nu_n(\mathbf{x}_{1i}), e_k, f_j(\ell_{31i}, \ell_{32i})) \beta^j$.
4. **Epipoles.** Given a set of epipolar lines $\{(\ell_2(\mathbf{x}_{1i}), \ell_3(\mathbf{x}_{1i}))\}_{i=1}^N$,
 - (a) Compute the multibody epipoles $\mathbf{c}_2 \in \mathbb{R}^{M_n(3)}$ and $\mathbf{c}_3 \in \mathbb{R}^{M_n(3)}$ from (8.55), and let $p_2(\ell_2) = \mathbf{c}_2^\top \nu_n(\ell_2)$ and $p_3(\ell_3) = \mathbf{c}_3^\top \nu_n(\ell_3)$.
 - (b) Compute the epipole pairs from the gradients of p_2 and p_3 as follows:
for $j = n : 1$

$$i_j = \arg \min_{i=1, \dots, N} \frac{|p_2(\ell_2(\mathbf{x}_{1i}))|}{\prod_{k=j+1}^n |\mathbf{e}_{2k}^\top \ell_2(\mathbf{x}_{1i})|} + \frac{|p_3(\ell_3(\mathbf{x}_{1i}))|}{\prod_{k=j+1}^n |\mathbf{e}_{3k}^\top \ell_2(\mathbf{x}_{1i})|};$$

$$\mathbf{e}_{2j} \sim \nabla p_2(\ell_2(\mathbf{x}_{1i_j})) \quad \text{and} \quad \mathbf{e}_{3j} \sim \nabla p_3(\ell_3(\mathbf{x}_{1i_j})).$$

end
5. **Feature segmentation.** Assign point correspondence $(\mathbf{x}_{1i}, \mathbf{x}_{2i}, \mathbf{x}_{3i})$ to motion $j = \arg \min_{k=1, \dots, n} (\mathbf{e}_{2k}^\top \ell_2(\mathbf{x}_{1i}))^2 + (\mathbf{e}_{3k}^\top \ell_3(\mathbf{x}_{1i}))^2$.
6. **Trifocal tensors.** Obtain the individual trifocal tensors $\{T_j\}_{j=1}^n$ from the trilinear constraint for each group.

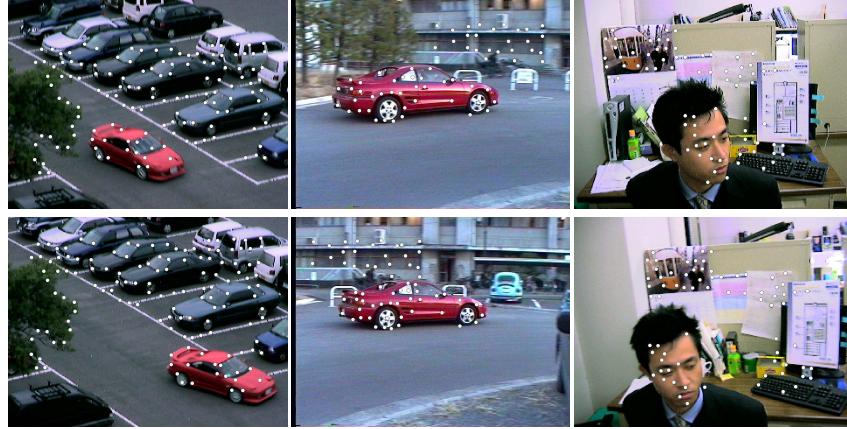


Figure 8.4. Segmenting the point correspondences of sequences A (left), B (center) and C (right) in [Kanatani and Sugaya, 2003] by clustering subspaces in \mathbb{R}^5 . First row: first frame of the sequence with point correspondences superimposed. Second row: last frame of the sequence with point correspondences superimposed.

As shown in Table 8.2, GPCA gives a percentage of correct classification of 100.0% for all three sequences. The table also shows results reported in [Kanatani and Sugaya, 2003] from existing *multiframe* algorithms for motion segmentation. The only algorithm having a comparable performance to GPCA is Kanatani's multi-stage optimization algorithm, which is based on solving a series of EM-like iterative optimization problems, at the expense of a significant increase in computation.

Table 8.2. Classification rates given by various subspace segmentation algorithms for sequences A, B, C in [Kanatani and Sugaya, 2003].

Sequence	A	B	C
Number of points	136	63	73
Number of frames	30	17	100
Costeira-Kanade	60.3%	71.3%	58.8%
Ichimura	92.6%	80.1%	68.3%
Kanatani: subspace separation	59.3%	99.5%	98.9%
Kanatani: affine subspace separation	81.8%	99.7%	67.5%
Kanatani: multi-stage optimization	100.0%	100.0%	100.0%
GPCA	100.0%	100.0%	100.0%

8.6.2 Segmentation of Fundamental Matrices

We first test Algorithm 8.1 on synthetic data. We randomly pick $n = 2$ collections of $N = 100n$ feature points and apply a different (randomly chosen) rigid body motion $(R_i, T_i) \in SE(3)$, with $R_i \in SO(3)$ the rotation and $T_i \in \mathbb{R}^3$ the translation. We add zero-mean Gaussian noise with standard deviation (std) from 0 to 1 pixels to the images \mathbf{x}_1 and \mathbf{x}_2 . The image size is 1000×1000 pixels. We run 1000 trials for each noise level. For each trial, the classification error is computed as the percentage of misclassified points, and the error between the true motions $\{(R_i, T_i)\}_{i=1}^n$ and their estimates $\{(\hat{R}_i, \hat{T}_i)\}_{i=1}^n$ are computed as

$$\text{Rotation error} = \frac{1}{n} \sum_{i=1}^n \text{acos}\left(\frac{\text{trace}(R_i \hat{R}_i^T) - 1}{2}\right) \quad (\text{degrees}).$$

$$\text{Translation error} = \frac{1}{n} \sum_{i=1}^n \text{acos}\left(\frac{T_i^T \hat{T}_i}{\|T_i\| \|\hat{T}_i\|}\right) \quad (\text{degrees}).$$

Figure 8.5 plots the mean classification error (%), the rotation error (degrees) and the translation error (degrees) as a function of noise. In all trials the number of motions was correctly estimated from equation (8.23) as $n = 2$.³ The mean classification error is less than 7% using an assignment based on epipoles and epipolar lines, and can be reduced to about 3.25% using an assignment based on the Sampson error. The rotation error is less than 0.38° and the translation error is less than 0.83° .

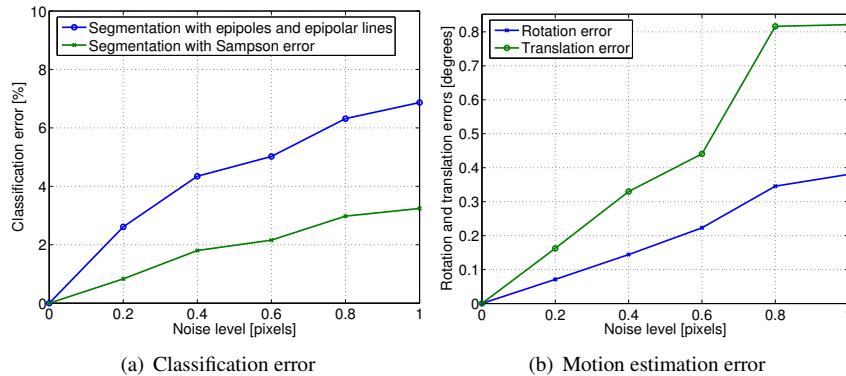


Figure 8.5. Percentage of misclassification and error in the estimation of rotation and translation (in degrees) as a function of the level of noise for $n = 2$ moving objects.

We also tested the proposed approach by segmenting a real sequence in which a moving camera observes a can moving in front of a static background consisting of a T-shirt and a book. We manually extracted a total of $N = 170$ correspon-

³We use $\kappa = 5 \times 10^{-3}$ in equation (2.14) for computing the number of motions.

dences: 70 for the can and 100 for the background. For comparison purposes, we estimated the ground truth motion (R_i, T_i) by applying the eight-point algorithm to manually segmented correspondences.

Figure 8.6 shows the first frame of the sequence as well as the relative displacement of the correspondences between the two frames. We applied Algorithm 8.1 to estimate the number of motions as $n = 2$.⁴ We obtained a misclassification error of 5.88% when the clustering is obtained using epipolar lines and epipoles only. We used this segmentation to obtain the motion parameters for each group. The error in rotation was 0.07° for the background and 4.12° for the can. The error in translation was 0.21° for the background and 4.51° for the can. Given the motion parameters for each group, we re-clustered the features using the Sampson error (??). The misclassification error reduced to 0%.

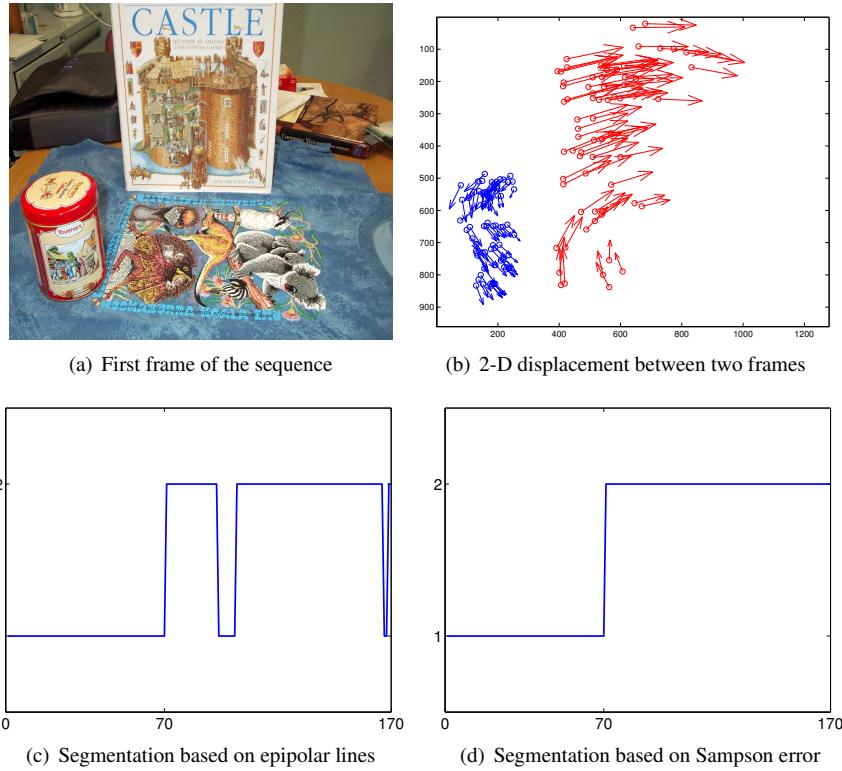


Figure 8.6. Segmenting two frames of a video sequence with two different rigid-body motions – the can and the background. (a) First frame of the sequence. (b) 2-D displacements of the 170 correspondences from the first view ('o') to the second ('→'). (c) Segmentation of the 170 correspondences using epipoles and epipolar lines. (d) Segmentation of the 170 correspondences using Sampson distance.

⁴We use $\kappa = 5 \times 10^{-3}$ in equation (2.14) for computing the number of motions.

8.6.3 Segmentation of Homography Matrices

We applied Algorithm 8.2 to segment two frames of a 2048×1536 video sequence, shown in Figure 8.7(a)-(b), with two moving objects – a cube and a checkerboard. Notice that although there are only *two* rigid-body motions, the scene contains *three* different homographies, each one associated with each one of the three visible planar structures. Furthermore, notice that the top side of the cube and the checkerboard have approximately the same normals. We manually tracked a total of $N = 147$ features: 98 in the cube (49 in each of the two visible sides) and 49 in the checkerboard. We applied Algorithm 8.2 to segment the image data and obtained a 97% of correct classification, as shown in Figure 8.7(c).

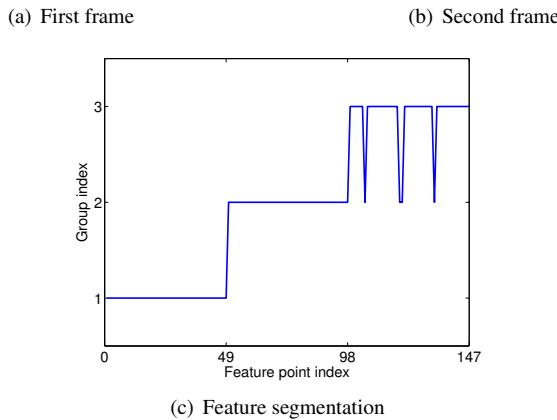


Figure 8.7. Segmenting two different rigid-body motions, a cube and a plane, according to three different homography models corresponding to the three planes in the scene.

We then added zero-mean Gaussian noise with standard deviation between 0 and 1 pixels to the features, after rectifying the features in the second view in order to simulate the noise free case. Figure 8.7(c) shows the mean percentage of correct classification for 1000 trials per level of noise. The percentage of correct classification of our algorithm is between 80% and 100%, which gives a very good

initial estimate for any of the existing iterative/optimization/EM based motion segmentation schemes.

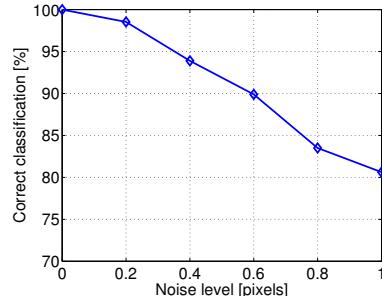


Figure 8.8. Percentage of correct classification as a function of noise synthetically added to the point correspondences of the scene in Figure 8.7.

8.7 Bibliographical Notes

3-D motion estimation and segmentation has been an active topic of research in the computer vision community over the past few years. Earlier work [Feng and Perona, 1998] solves this problem by first clustering the features corresponding to the same motion using e.g., K-means or spectral clustering, and then estimating a single motion model for each group. This can also be done in a probabilistic framework [Torr, 1998] in which a maximum-likelihood estimate of the parameters of each motion model is sought by alternating between feature clustering and single-body motion estimation using the Expectation Maximization (EM) algorithm. However, the convergence of EM to the global maximum depends strongly on initialization [Torr et al., 2001].

In order to deal with the initialization problem of EM-like approaches, recent work has concentrated on the study of the geometry of dynamic scenes, including the analysis of multiple points moving linearly with constant speed [Han and Kanade, 2000, Shashua and Levin, 2001] or in a conic section [Avidan and Shashua, 2000], multiple points moving in a plane [Sturm, 2002], multiple translating planes [Wolf and Shashua, 2001a], self-calibration from multiple motions [Fitzgibbon and Zisserman, 2000, Han and Kanade, 2001], multiple moving objects seen by an affine camera [Boult and Brown, 1991, Costeira and Kanade, 1998, Kanatani, 2001, Wu et al., 2001, Kanatani and Matsunaga, 2002b, Zelnik-Manor and Irani, 2003, Kanatani and Sugaya, 2003, Vidal and Hartley, 2004], and two-object segmentation from two perspective views [Wolf and Shashua, 2001b]. The case of multiple moving objects seen by two perspective views was recently studied in [Vidal et al., 2002b, Vidal and Sastry, 2003, Vidal and Ma, 2004, Vidal et al., 2005], and has been extended to three perspective views via the so-called *multibody trifocal tensor* [Hartley and Vidal, 2004]. Such works have been the basis for the

material presented in this chapter. Recent extensions omnidirectional cameras can be found in [Shakernia et al., 2003, ?].

8.8 Exercises

Exercise 8.1 Motion Segmentation from Optical Flow in Multiple Perspective Views.

Let $\Omega_f = (\omega_{1f}, \omega_{2f}, \omega_{3f})^\top$ and $V_f = (v_{1f}, v_{2f}, v_{3f})^\top$ be, respectively, the rotational and translational velocities of one of the moving objects relative to the camera at frame $f = 1, \dots, F$. Under the perspective projection model, the projection of point $\mathbf{X}_i = (X_i, Y_i, Z_i, 1)^\top \in \mathbb{P}^3$ on the zeroth frame is $(x_i, y_i)^\top = (X_i, Y_i)^\top / Z_i$, and its optical flow $\mathbf{u}_{fp} \in \mathbb{R}^2$ in the f th frame relative to the zeroth is:

$$\mathbf{u}_{fp} = \begin{bmatrix} x_i y_i & -(1+x_i^2) & -y_i & 1/Z_i & 0 & x_i/Z_i \\ (1+y_i^2) & -x_i y_i & x_i & 0 & 1/Z_i & y_i/Z_i \end{bmatrix} \begin{bmatrix} \Omega_f \\ V_f \end{bmatrix}.$$

Given measurements for the optical flow $\{\mathbf{u}_{fp}\}$ of P pixels in F frames, we can stack all the image measurements into a $2F \times P$ matrix W

$$W = \begin{bmatrix} \mathbf{u}_{11} & \cdots & \mathbf{u}_{1P} \\ \vdots & & \vdots \\ \mathbf{u}_{F1} & \cdots & \mathbf{u}_{FP} \end{bmatrix}_{2F \times P} \quad (8.60)$$

that can be factored into its motion and structure components as $W = MS^\top$, where

$$M = \begin{bmatrix} \omega_{11} & \omega_{21} & -\omega_{31} & 0 & 0 & v_{11} & v_{31} & 0 \\ -\omega_{21} & 0 & 0 & \omega_{11} & \omega_{31} & v_{21} & 0 & v_{31} \\ \vdots & & & & & & \vdots & \\ \omega_{1F} & \omega_{2F} & -\omega_{3F} & 0 & 0 & v_{1F} & v_{3F} & 0 \\ -\omega_{2F} & 0 & 0 & \omega_{1F} & \omega_{3F} & v_{2F} & 0 & v_{3F} \end{bmatrix}_{2F \times 8} \quad (8.61)$$

$$S = \begin{bmatrix} x_1 y_1 & z_1 - x_1^2 & y_1 & y_1^2 - z_1 & x_1 & \frac{1}{\lambda_1} & \frac{x_1}{\lambda_1} & \frac{y_1}{\lambda_1} \\ \vdots & & & & & & \vdots & \\ x_N y_N & z_N - x_N^2 & y_N & y_N^2 - z_N & x_N & \frac{1}{\lambda_N} & \frac{x_N}{\lambda_N} & \frac{y_N}{\lambda_N} \end{bmatrix}_{N \times 8}.$$

Therefore, $\text{rank}(W) \leq 8$, hence the vector containing the optical flow of a point between the zeroth and the f th frame for $f = 1, \dots, F$ lives in a subspace of \mathbb{R}^{2F} of dimension at most 8. This rank constraint, among others, was derived in [Irani, 1999] and was used to derive a multi-frame algorithm for the estimation of the optical flow of a moving camera observing a static scene.

Exercise 8.2 Show that for all $\ell_1, \ell_2 \in \mathbb{R}^3$ and $\alpha \in \mathbb{R}$

$$\nu_n(\alpha \ell_1 + \ell_2) = \sum_{i=0}^n \alpha^i f_i(\ell_1, \ell_2), \quad (8.62)$$

where $f_i(\ell_1, \ell_2) \in \mathbb{R}^{M_n(3)}$ is a bi-homogeneous polynomial of degree i in ℓ_1 and $(n-i)$ in ℓ_2 for $i = 0, \dots, n$.

Exercise 8.3 Show that

$$\begin{aligned} & \mathcal{T}(\nu_n(\mathbf{x}_1), \nu_n(\alpha\ell_{21} + \ell_{22}), \nu_n(\beta\ell_{31} + \ell_{32})) \\ &= \mathcal{T}(\nu_n(\mathbf{x}_1), \sum_{i=0}^n \alpha^i f_i(\ell_{21}, \ell_{22}), \sum_{j=0}^n \beta^j f_j(\ell_{31}, \ell_{32})) \\ &= \sum_{i=0}^n \sum_{j=0}^n \alpha^i \beta^j \mathcal{T}(\nu_n(\mathbf{x}_1), f_i(\ell_{21}, \ell_{22}), f_j(\ell_{31}, \ell_{32})) \end{aligned} \quad (8.63)$$

Exercise 8.4 We interpret the second image $\mathbf{x}_2 \in \mathbb{P}^2$ as a point in \mathbb{CP} by considering the first two coordinates in \mathbf{x}_2 as a complex number and appending a one to it. However, we still think of \mathbf{x}_1 as a point in \mathbb{P}^2 . With this interpretation, we can rewrite (??) as

$$\mathbf{x}_2 \sim H\mathbf{x}_1 \doteq \begin{bmatrix} h_{11} + h_{21}\sqrt{-1} & h_{12} + h_{22}\sqrt{-1} & h_{13} + h_{23}\sqrt{-1} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \mathbf{x}_1, \quad (8.64)$$

where $H \in \mathbb{C}^{2 \times 3}$ now represents a *complex homography*⁵. Let \mathbf{w}_2 be the vector in \mathbb{CP} perpendicular to \mathbf{x}_2 , i.e., if $\mathbf{x}_2 = (z, 1)$ then $\mathbf{w}_2 = (1, -z)$. Then we can rewrite (8.64) as the following *complex bilinear constraint*

$$\mathbf{w}_2^\top H\mathbf{x}_1 = 0, \quad (8.65)$$

which we call the *complex homography constraint*. We can therefore interpret the motion segmentation problem as one in which we are given image data $\{\mathbf{x}_1^j \in \mathbb{P}^2\}_{j=1}^N$ and $\{\mathbf{w}_2^j \in \mathbb{CP}\}_{j=1}^N$ generated by a collection of n complex homographies $\{H_i \in \mathbb{C}^{2 \times 3}\}_{i=1}^n$. Then each image pair $(\mathbf{x}_1, \mathbf{w}_2)$ has to satisfy the *multibody homography constraint*

$$\prod_{i=1}^n (\mathbf{w}_2^\top H_i \mathbf{x}_1) = \nu_n(\mathbf{w}_2)^\top \mathcal{H} \nu_n(\mathbf{x}_1) = 0, \quad (8.66)$$

regardless of which one of the n complex homographies is associated with the image pair. We call the matrix $\mathcal{H} \in \mathbb{C}^{M_n(2) \times M_n(3)}$ the *multibody homography*. Now, since the multibody homography constraint (8.66) is linear in the multibody homography \mathcal{H} , we can linearly solve for \mathcal{H} from (8.66) given $N \geq M_n(2)M_n(3) - (M_n(3) + 1)/2 \sim O(n^3)$ image pairs in general position⁶ with at least 4 pairs per moving object.

Given the multibody homography $\mathcal{H} \in \mathbb{C}^{M_n(2) \times M_n(3)}$, the rest of the problem is to recover the individual homographies $\{H_i\}_{i=1}^n$. In the case of fundamental matrices discussed in Section ??, the key for solving the problem was the fact that fundamental matrices are of rank 2, hence one can cluster epipolar lines based on the epipoles. In principle, we cannot do the same with real homographies $H_i \in \mathbb{R}^{3 \times 3}$, because in general they are full rank. However, if we work with complex homographies $H_i \in \mathbb{C}^{2 \times 3}$ they automatically have a right null space which we call the *complex epipole* $\mathbf{e}_i \in \mathbb{C}^3$. Then, similarly to (8.24), we can associate a *complex epipolar line*

$$\ell^j \sim \frac{\partial \nu_n(\mathbf{w}_2)^\top \mathcal{H} \nu_n(\mathbf{x}_1)}{\partial \mathbf{x}_1} \Big|_{(\mathbf{x}_1, \mathbf{w}_2) = (\mathbf{x}_1^j, \mathbf{w}_2^j)} \in \mathbb{CP}^2 \quad (8.67)$$

⁵Strictly speaking, we embed each real homography matrix into an affine complex matrix.

⁶The multibody homography constraint gives two equations per image pair, and there are $(M_n(2) - 1)M_n(3)$ complex entries in \mathcal{H} and $M_n(3)$ real entries (the last row).

with each image pair $(\mathbf{x}_1^j, \mathbf{w}_2^j)$. Given this set of $N \geq M_n(3) - 1$ complex epipolar lines $\{\ell^j\}_{j=1}^N$, with at least 2 lines per moving object, we can apply Algorithm ?? with $K = 3$ and $\Pi = I$ to estimate the n complex epipoles $\{e_i \in \mathbb{C}^3\}_{i=1}^n$ up to a scale factor, as in equation (8.52). Therefore, if the n complex epipoles are different, we can cluster the original image measurements by assigning image pair $(\mathbf{x}_1^j, \mathbf{x}_2^j)$ to group i if $i = \arg \min_{k=1, \dots, n} |e_k^\top \ell^j|^2$. Once the image pairs have been clustered, the estimation of each homography, either real or complex, becomes a simple linear problem.

Remark 8.1 (Direct extraction of homographies from \mathcal{H}). *There is yet another way to obtain individual H_i from \mathcal{H} without segmenting the image pairs first. Once the complex epipoles e_i are known, one can compute the following linear combination of the rows of H_i (up to scale) from the derivatives of the multibody homography constraint at e_i*

$$\mathbf{w}^\top H_i \sim \left. \frac{\partial \nu_n(\mathbf{w})^\top \mathcal{H} \nu_n(\mathbf{x})}{\partial \mathbf{x}} \right|_{\mathbf{x}=e_i} \in \mathbb{CP}^2, \quad \forall \mathbf{w} \in \mathbb{C}^2. \quad (8.68)$$

In particular, if we take $\mathbf{w} = [1, 0]^\top$ and $\mathbf{w} = [0, 1]^\top$ we obtain the first and second row of H_i (up to scale), respectively. By choosing additional \mathbf{w} 's one obtains more linear combinations from which the rows of H_i can be linearly and uniquely determined.

Remark 8.2 (Independent homographies). *The above solution assumes that the complex epipoles are different (up to a scale factor). We take this assumption as our definition of independent homographies, even though it is more restrictive than saying that the real homographies $H_i \in \mathbb{R}^{3 \times 3}$ are different (up to a scale factor). However, one can show that, under mild conditions, e.g., the third rows of each H_i are different, the null spaces of the complex homographies are indeed different for different real homographies.⁷*

Remark 8.3 (One rigid-body motion versus multiple ones). *A homography is generally of the form $H = R + T\pi^\top$ where π is the plane normal. If the homographies come from different planes (different π) undergoing the same rigid-body motion, the proposed scheme would work just fine since different normal vectors π will cause the complex epipoles to be different. However, if multiple planes with the same normal vector $\pi = [0, 0, 1]^\top$ undergo pure translational motions of the form $T_i = [T_{xi}, T_{yi}, T_{zi}]^\top$, then all the complex epipoles are equal to $e_i = [\sqrt{-1}, -1, 0]^\top$. To avoid this problem, one can complexify the first and third rows of H instead of the first two. The new complex epipoles are $e_i = [T_{xi} + T_{zi}\sqrt{-1}, T_{yi}, -1]^\top$, which are different for different translations.*

Exercise 8.5 In this section, we demonstrate that one can estimate the individual trifocal tensors *without* first clustering the image correspondences. The key is to look at second order derivatives of the multibody trilinear constraint. Therefore, we contend that *all* the geometric information about the multiple motions is already encoded in the multibody trifocal tensor.

Let \mathbf{x} be an arbitrary point in \mathbb{P}^2 (not necessarily a point in the first view). Since the i^{th} epipole e'_i is known, we can compute two lines ℓ'_{i1} and ℓ'_{i2} passing through e'_i and apply Algorithm ?? to compute the epipolar line of \mathbf{x} in the second view ℓ'_{ix} according to the i^{th} motion. In a completely analogous fashion, we can compute the epipolar line

⁷The set of complex homographies that share the same null space is a five-dimensional subset (hence a zero-measure subset) of all real homography matrices. Furthermore, one can complexify any other two rows of H instead of the first two. As long as two homography matrices are different, one of the complexifications will give different complex epipoles.

of x in the third view ℓ''_{ix} from two lines passing through e''_i . Given $(\ell'_{ix}, \ell''_{ix})$, a simple calculation shows that the slices of the trifocal tensor T_i can be expressed in terms of the second derivative of the multibody epipolar constraint, as follows:

$$\frac{\partial^2(\tilde{x}\tilde{\ell}\tilde{\ell}''T)}{\partial\ell'\partial\ell''}\Bigg|_{(x,\ell'_{ix},\ell''_{ix})} = M_{ix} \sim xT_i \in \mathbb{R}^{3 \times 3}. \quad (8.69)$$

Thanks to (8.69), we can immediately outline an algorithm for computing the individual trifocal tensors.

Chapter 9

Dynamical Texture and Video Segmentation

Part III

Extensions to Arrangements of Dynamical Systems and Nonlinear Varieties

+ This is page 208
Printer: Opaque this

Chapter 10

Switched ARX Systems

Hybrid systems are mathematical models that are used to describe continuous processes that occasionally exhibit discontinuous behaviors due to sudden changes of dynamics. For instance, the continuous trajectory of a bouncing ball results from alternating between a free fall and an elastic contact with the ground. However, hybrid systems can also be used to describe a complex process or time series that does not itself exhibit discontinuous behaviors, by approximating the process or series with a simpler class of dynamical models. For example, a non-linear dynamical system can be approximated by switching among a set of linear systems, each approximating the nonlinear system in a subset of its state space. As another example, a video sequence can be segmented to different scenes by fitting a piecewise linear dynamical model to the entire sequence.

In recent years, there has been significant interest and progress in the study of the analysis, stability, and control of hybrid systems. Knowing the system parameters, many successful theories have been developed to characterize the behaviors of hybrid systems under different switching mechanisms. However, in practice, the parameters and the switching mechanism of a hybrid system are often not known or derivable from first principles. We are faced with the task of identifying the system from its input and output measurements.

In this chapter, we show how to apply the GPCA method to the problem of identifying a class of discrete-time hybrid systems known as hybrid Auto Regres-

sive eXogenous (ARX) systems.¹ We know from classic identification theory of linear systems that the configuration space of the input/output data generated by a single ARX system, say

$$y_t = \sum_{j=1}^{n_a} a_j y_{t-j} + \sum_{j=1}^{n_c} c_j u_{t-j} + w_t, \quad y_t, u_t, w_t \in \mathbb{R}, \quad (10.1)$$

is a linear subspace. The problem of identifying the system is equivalent to identifying this subspace from a finite number of samples on the subspace (as we will review briefly in Section 10.2). Unfortunately, for a hybrid system that switches among multiple ARX systems, as shown in Figure 10.1, when the orders of the constituent systems are different, depending on the switching sequence λ_t , the configuration space of the hybrid ARX system might *not* simply be a union of the configuration spaces of the constituent ARX systems. Therefore, the problem of identifying the hybrid ARX system is *not* a trivial subspace segmentation problem.

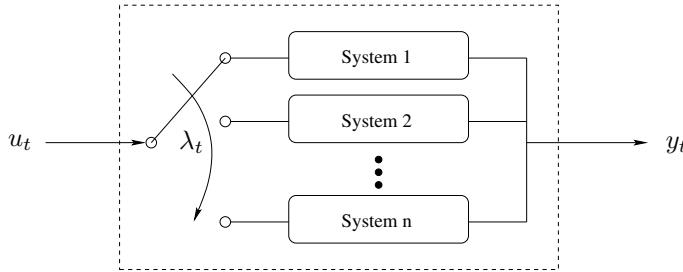


Figure 10.1. The input/output diagram of a hybrid system switching among n constituent systems. The identification problem requires to infer what is in the black box (including the n systems and the switching mechanism λ_t) from its input u_t and output y_t .

In this chapter, we show how to incorporate some special (algebraic and dynamical) structures of a hybrid ARX system so that the identification problem can still be solved by a special version of the GPCA method. In particular, we will show that a hybrid ARX system can still be correctly identified from a special polynomial p that fits the input/output data of the hybrid ARX system – the last nonzero term of p has the lowest degree-lexicographic order in the ideal \mathfrak{a} of polynomials. This polynomial is unique, factorable, and independent of the switching sequence. The non-repeated factors of this polynomial correspond to the constituent ARX systems, hence the number of systems is given by the number of non-repeated factors (Section 10.3).

Although the analysis and algorithm will be developed primarily in a noise-free algebraic setting, the GPCA-based identification algorithm is numerically

¹ARX systems are an extremely popular class of dynamical models that are widely used in control, signal processing, communications, and economics. In image/video processing, they can be used to model videos of dynamical scenes.

stable and works with moderate noises. Simulation and experimental results show that the algorithm performs extremely well for both synthetic and real data, in comparison with the existing iterative (e.g., EM-based) identification algorithms (Section 10.4).

10.1 Problem Statement

Now let us consider a hybrid ARX system – a system that switches among multiple, say n , ARX systems of the type (10.1). Mathematically, such a system can be described as

$$y_t = \sum_{j=1}^{n_a(\lambda_t)} a_j(\lambda_t) y_{t-j} + \sum_{j=1}^{n_c(\lambda_t)} c_j(\lambda_t) u_{t-j} (+ w_t), \quad (10.2)$$

where $u_t \in \mathbb{R}$ is the *input*, $y_t \in \mathbb{R}$ is the *output*, $\lambda_t \in \{1, 2, \dots, n\}$ is the *discrete state*, and $n_a(i)$, $n_c(i)$, $\{a_j(i)\}_{j=1}^{n_a(i)}$ and $\{c_j(i)\}_{j=1}^{n_c(i)}$ are, respectively, the orders and the system parameters of the i th ARX system for $i = 1, \dots, n$. The last term w_t is zero for a deterministic ARX system and a white-noise random process for a stochastic system. The purpose of this paper is to provide an analytic solution to the deterministic case, which approximates the stochastic case when w_t is small.

The discrete state λ_t , also called the *mode* of the system, can evolve due to a variety of mechanisms. In the least restrictive case, $\{\lambda_t\}$ is a deterministic but unknown sequence that can take a finite number of possible values, which we can assume to coincide with a collection of integers:

$$\lambda : t \in \mathbb{Z} \mapsto \lambda_t \in \{1, 2, \dots, n\}.$$

One can further restrict the set of switching sequences by assuming that λ_t is a realization of an irreducible Markov chain, governed by transition probabilities

$$\pi(i, j) \doteq P(\lambda_{t+1} = j | \lambda_t = i).$$

In this case, the system (10.2) is often called a “Jump-Markov Linear System” (JMLS). Alternatively, one can assume that λ_t is a piecewise constant function of the “continuous states” of the system (10.2),

$$\lambda : (y_{t-1}, \dots, y_{t-n_a}) \in \mathbb{R}^{n_a} \mapsto \lambda_t \in \{1, 2, \dots, n\}.$$

In this case, the system (10.2) is often called a “PieceWise ARX” (PWARX) system.

In this chapter we will consider the first scenario, so that our results also apply to other switching mechanisms if that information becomes available. Therefore, our method does not depend on any particular switching mechanism. Once the switching sequence has been identified, the switching mechanism can be further retrieved.

The following problem summarizes the goal of this chapter. In the sequel

Problem 10.1 (Identification of Hybrid Auto Regressive eXogenous Systems).

Given input/output data $\{u_t, y_t\}_{t=0}^T$ generated by an HARX system such as (10.2), identify the number of constituent systems n , the orders of each ARX system $\{n_a(i), n_c(i)\}_{i=1}^n$, the system parameters $\{a_j(i)\}_{j=1}^{n_a(i)}$ and $\{c_j(i)\}_{j=1}^{n_c(i)}$, and the discrete states $\{\lambda_t\}$.

we characterize a set of (sufficient) conditions that allow one to solve the above problem as well as develop an efficient algorithm for it.

10.2 Identification of a Single ARX System

For the sake of completeness and comparison, let us first review some classic results for the identification of a single discrete-time ARX system

$$y_t = a_1 y_{t-1} + \cdots + a_{n_a} y_{t-n_a} + c_1 u_{t-1} + \cdots + c_{n_c} u_{t-n_c}. \quad (10.3)$$

The transfer function $\hat{H}(z) \doteq \hat{y}(z)/\hat{u}(z)$ of the system (10.3) is given by:

$$\begin{aligned} \hat{H}(z) &= z^{\max(n_a - n_c, 0)} \tilde{H}(z) \\ &= \frac{z^{\max(n_a - n_c, 0)} (z^{n_c-1} c_1 + z^{n_c-2} c_2 + \cdots + c_{n_c})}{z^{\max(n_c - n_a, 0)} (z^{n_a} - z^{n_a-1} a_1 - z^{n_a-2} a_2 - \cdots - a_{n_a})}. \end{aligned} \quad (10.4)$$

From the theory of signals and systems, given the infinite sequences of the input $\{y_t\}$ and the output $\{u_t\}$, we can compute their Z-transform $\hat{y}(z)$ and $\hat{u}(z)$, respectively. Then we can identify the parameters of the ARX model by directly computing $\hat{H}(z)$ as $\hat{y}(z)/\hat{u}(z)$.² This requires the ARX model to be *identifiable*, i.e., $\tilde{H}(z)$ must have no pole-zero cancellation,³ and $\hat{u}(z)$ to have no zero in common with a pole of $\hat{H}(z)$ and vice versa.

Alternatively, we may identify the system via the identification of a *subspace* associated with the input/output data. Let us define $D \doteq n_a + n_c + 1$ and the vector of *regressors* to be:

$$\boldsymbol{x}_t \doteq [y_t, y_{t-1}, \dots, y_{t-n_a}, u_{t-1}, u_{t-2}, \dots, u_{t-n_c}]^T \in \mathbb{R}^D. \quad (10.5)$$

Thus, for all time t , the so-defined \boldsymbol{x}_t is orthogonal to the vector that consists of the parameters of the ARX system:

$$\boldsymbol{b} \doteq [1, -a_1, -a_2, \dots, -a_{n_a}, -c_1, -c_2, \dots, -c_{n_c}]^T \in \mathbb{R}^D. \quad (10.6)$$

²Notice that this scheme is not practical since it requires one to obtain the typically infinitely-long output sequence $\{y_t\}$.

³That is, the polynomials $z^{\max(n_c - n_a, 0)} (z^{n_a} - z^{n_a-1} a_1 - z^{n_a-2} a_2 - \cdots - a_{n_a})$ and $z^{n_c-1} c_1 + z^{n_c-2} c_2 + \cdots + c_{n_c}$ are co-prime.

That is, $\forall t$ \mathbf{x}_t and \mathbf{b} satisfy the equation $\mathbf{b}^T \mathbf{x}_t = 0$. In other words, \mathbf{b} is the normal vector to the hyperplane spanned by (the rows of) the following *data matrix*:

$$\mathbf{L}(n_a, n_c) \doteq [\mathbf{x}_{\max(n_a, n_c)}, \dots, \mathbf{x}_{t-1}, \mathbf{x}_t, \mathbf{x}_{t+1}, \dots]^T \in \mathbb{R}^{\infty \times D}. \quad (10.7)$$

When the model orders n_a, n_c are known, we can readily solve for the model parameters \mathbf{b} from the null space of $\mathbf{L}(n_a, n_c)$ via SVD.

In practice, however, the model orders may be unknown, and only upper bounds \bar{n}_a and \bar{n}_c may be available. Thus, the vector of regressors \mathbf{x}_t is

$$\mathbf{x}_t \doteq [y_t, y_{t-1}, y_{t-2}, \dots, y_{t-\bar{n}_a}, u_{t-1}, u_{t-2}, \dots, u_{t-\bar{n}_c}]^T \in \mathbb{R}^D, \quad (10.8)$$

where $D = \bar{n}_a + \bar{n}_c + 1$. Obviously, the following vector

$$\mathbf{b} \doteq [1, -a_1, \dots, -a_{n_a}, \mathbf{0}_{1 \times (\bar{n}_a - n_a)}, -c_1, \dots, -c_{n_c}, \mathbf{0}_{1 \times (\bar{n}_c - n_c)}]^T \quad (10.9)$$

satisfies the equation $\mathbf{x}_t^T \mathbf{b} = 0$ for all t . Notice that here the vector \mathbf{b} is the one in (10.6) with additional $\bar{n}_a - n_a$ and $\bar{n}_c - n_c$ zeros filled in after the terms $-a_{n_a}$ and $-c_{n_c}$, respectively.

Let us define the data matrix $\mathbf{L}(\bar{n}_a, \bar{n}_c)$ in the same way as in equation (10.7). Because of the redundant embedding (10.8), the vector \mathbf{b} is no longer the only one in the null space of \mathbf{L} . It is easy to verify that all the following vectors are also in the null space of \mathbf{L} :

$$\begin{aligned} \mathbf{b}^1 &= [\mathbf{0}_1, 1, -a_1, \dots, -a_{n_a}, \mathbf{0}_{\bar{n}_a - n_a - 1}, \mathbf{0}_1, -c_1, \dots, -c_{n_c}, \mathbf{0}_{\bar{n}_c - n_c - 1}]^T, \\ \mathbf{b}^2 &= [\mathbf{0}_2, 1, -a_1, \dots, -a_{n_a}, \mathbf{0}_{\bar{n}_a - n_a - 2}, \mathbf{0}_2, -c_1, \dots, -c_{n_c}, \mathbf{0}_{\bar{n}_c - n_c - 2}]^T, \\ &\vdots && \vdots \end{aligned} \quad (10.10)$$

Therefore, the data $\{\mathbf{x}_t\}$ span a low-dimensional linear subspace \mathcal{S} in the ambient space \mathbb{R}^D .⁴ Each of the vectors defined above uniquely determines the original system (10.3), including its order and coefficients. However, a vector in the null space of \mathbf{L} is in general a linear combination of all such vectors and it is not necessarily one of the above. Thus, in order to identify the original system from the data matrix \mathbf{L} , we need to seek a vector in its null space that has certain desired structure.

Notice that the last $\bar{n}_c - n_c$ entries of \mathbf{b} in (10.9) are zero, hence the last non-zero entry of \mathbf{b} has the lowest order – in terms of the ordering of the entries of \mathbf{x}_t – among all vectors that are in the null space of \mathbf{L} . Therefore, we can obtain the first $\bar{n}_a + n_c + 1$ entries of \mathbf{b} from the null space of the submatrix of \mathbf{L} defined by its first $\bar{n}_a + n_c + 1$ columns. Since n_c is unknown, we can incrementally take the first $j = 1, 2, \dots$ columns of the matrix \mathbf{L} from the left to the right:

$$\mathbf{L}^1 \doteq \mathbf{L}(:, 1 : 1), \quad \mathbf{L}^2 \doteq \mathbf{L}(:, 1 : 2), \quad \dots, \quad \mathbf{L}^j \doteq \mathbf{L}(:, 1 : j), \quad (10.11)$$

⁴Only when the initial conditions $\{y_{t_0-1}, \dots, y_{t_0-\bar{n}_a}\}$ are arbitrary do the data span a hyperplane in \mathbb{R}^D with \mathbf{b} as the only normal vector.

until the rank of the submatrix L^j stops increasing for the first time for some $j = m$.⁵

Remark 10.1 (Identifying \mathbf{b} and m in the Stochastic Case). *In the stochastic case (i.e., $w_t \neq 0$), the ultimate goal is to minimize the (squared) modeling error $\sum_t w_t^2 = \sum_t (\mathbf{b}^T \mathbf{x}_t)^2$, which corresponds to the maximum-likelihood estimate when w_t is white-noise. Then the optimal solution \mathbf{b}^* can be found in a least-square sense as the singular vector that corresponds to the smallest singular value of L^m . However, in the noisy case, we cannot directly estimate m from the rank of L^j since it might be full rank for all j . Based on model selection techniques, m can be estimated from a noisy L^j as*

$$m = \underset{j=1, \dots, D}{\operatorname{argmin}} \left\{ \frac{\sigma_j^2(L^j)}{\sum_{k=1}^{j-1} \sigma_k^2(L^j)} + \kappa \cdot j \right\}, \quad (10.12)$$

where $\sigma_k(L^j)$ is the k th singular value of L^j and $\kappa \in \mathbb{R}$ is a parameter weighting the two terms. The above criterion minimizes a cost function that consists of a data fitting term and a model complexity term. The data fitting term measures how well the data is approximated by the model – in this case how close the matrix L^j is to dropping rank. The model complexity term penalizes choosing models of high complexity – in this case choosing a large rank.

There is, however, a much more direct way of dealing with the case of unknown orders. The following lemma shows that the system orders n_a and n_c together with the system parameters \mathbf{b} can all be simultaneously and uniquely computed from the data.

Lemma 10.2 (Identifying the Orders of an ARX System). *Suppose we are given data generated by an identifiable ARX model whose input $\hat{u}(z)$ shares no poles or zeros with the zeros or poles, respectively, of the model transfer function $\hat{H}(z)$. If $\bar{n}_a + \bar{n}_c + 1 \leq n_a + n_c + 1$, then*

$$\operatorname{rank}(L(\bar{n}_a, \bar{n}_c)) = \begin{cases} \bar{n}_a + \bar{n}_c & \text{if and only if } \bar{n}_a = n_a \text{ and } \bar{n}_c = n_c, \\ \bar{n}_a + \bar{n}_c + 1 & \text{otherwise.} \end{cases} \quad (10.13)$$

Therefore the systems orders can be computed as:

$$(n_a, n_c) = \arg \min_{(\bar{n}_a, \bar{n}_c) \in \mathbb{Z}^2} \{ \bar{n}_a + \bar{n}_c : \operatorname{rank}(L(\bar{n}_a, \bar{n}_c)) = \bar{n}_a + \bar{n}_c \}. \quad (10.14)$$

The parameter vector \mathbf{b} is the unique vector in the null space of $L(n_a, n_c)$.

Proof. Suppose $\operatorname{rank}(L(\bar{n}_a, \bar{n}_c)) \leq \bar{n}_a + \bar{n}_c$ and $\mathbf{b}' = [1, b'_1, b'_2, \dots, b'_{\bar{n}_a + \bar{n}_c}] \in \mathbb{R}^{\bar{n}_a + \bar{n}_c + 1}$ is a nonzero vector such that $L\mathbf{b}' = 0$. Consider the Z-transform of

⁵If n_c was known, then we would have $m = \bar{n}_a + n_c + 1$.

$L\mathbf{b}' = 0$:

$$\begin{aligned} & \hat{y}(z) + b'_1 z^{-1} \hat{y}(z) + \cdots + b'_{\bar{n}_a} z^{-\bar{n}_a} \hat{y}(z) \\ & + b'_{\bar{n}_a+1} z^{-1} \hat{u}(z) + \cdots + b'_{\bar{n}_a+\bar{n}_c} z^{-\bar{n}_a-\bar{n}_c} \hat{u}(z) = 0. \end{aligned}$$

Since $\hat{u}(z)$ does not have any of the poles or zeros of the transfer function $\hat{H}(z)$ in (10.4), the ratio $\hat{y}(z)/\hat{u}(z)$ derived from the above equation should be a rational function whose numerator and denominator contain those of $\hat{H}(z)$ as factors, respectively. Since $\bar{n}_a + \bar{n}_c \leq n_a + n_c$, this happens only if $\bar{n}_a = n_a$ and $\bar{n}_c = n_c$ and the vector \mathbf{b}' is exactly the same as \mathbf{b} in (10.6). \square

Remark 10.3 (Identifying n_a, n_c in the Stochastic Case). *In the stochastic case (i.e., $w_t \neq 0$), we cannot directly estimate n_a, n_c from the rank of $L(\bar{n}_a, \bar{n}_c)$ since it might be full rank for all \bar{n}_a, \bar{n}_c . From model selection methods, n_a, n_c can be estimated from a noisy L as*

$$(n_a, n_c) = \arg \min_{(\bar{n}_a, \bar{n}_c) \in \mathbb{Z}^2} \left\{ \frac{\sigma_{\bar{n}_a+\bar{n}_c+1}^2(L(\bar{n}_a, \bar{n}_c))}{\sum_{k=1}^{\bar{n}_a+\bar{n}_c} \sigma_k^2(L(\bar{n}_a, \bar{n}_c))} + \kappa \cdot (\bar{n}_a + \bar{n}_c) \right\}, \quad (10.15)$$

where $\sigma_k(L)$ is the k th singular value of L and $\kappa \in \mathbb{R}$ is a parameter weighting the two terms – the first for the model fitting error and the second for the model complexity.

In principle, the above lemma allows us to identify the precise orders n_a, n_c and the vector \mathbf{b} of the ARX system from the (infinite) sequences of input $\{u_t\}$ and output $\{y_t\}$. In practice, we are usually given a finite input/output sequence. In such cases, we need to assume that the sequence of regressors is *sufficiently exciting*, i.e., the $T \times (n_a + n_c + 1)$ submatrix

$$L \doteq [x_{\max(n_a, n_c)}, \dots, x_{\max(n_a, n_c)+T-1}]^T$$

has the same rank $n_a + n_c$ as the “full” L matrix defined in (10.7). Then, the maximum-likelihood estimate for $\mathbf{b} \in \mathbb{R}^{n_a+n_c+1}$ can be identified as the singular vector that corresponds to the smallest singular value of L .

This condition for sufficient exciting for finite data can also be expressed in terms of only the input sequence. As shown in [Anderson and Johnson, 1982], the regressors are sufficiently exciting if the input sequence $\{u_t\}$ is, i.e., if the following vectors

$$\mathbf{u}_t \doteq [u_t, u_{t-1}, \dots, u_{t-n_a-n_c+1}]^T \in \mathbb{R}^{n_a+n_c}, \quad n_a + n_c - 1 \leq t \leq T,$$

span an $(n_a + n_c)$ -dimensional subspace.

10.3 Identification of Hybrid ARX Systems

From our discussion in the previous section, we know that the regressors generated by an identifiable ARX system with sufficiently exciting input live in a linear subspace in \mathbb{R}^D where $D = \bar{n}_a + \bar{n}_c + 1$ and \bar{n}_a, \bar{n}_c are upper bounds on the

orders of the system. The problem of identifying the ARX system becomes one of seeking a vector in the orthogonal complement to this subspace that has certain desired structure. We show in this section how to generalize these concepts to the more challenging problem of identifying a hybrid ARX system (Problem 10.1). Most of our development will focus on the case of single-input single-output (SISO) systems. However, we will discuss at the end of this section how our approach can be easily extended to multiple-input multiple-output (MIMO) systems.

Consider an input/output sequence $\{u_t, y_t\}$ generated by a hybrid ARX system switching among a set of n ARX systems with parameters $\{\mathbf{b}_i\}_{i=1}^n$ and possibly different orders $\{n_a(i), n_c(i)\}_{i=1}^n$. We assume that the HARX system is *identifiable*, i.e., for all $i = 1, \dots, n$, the rational function $\tilde{H}_i(z)$ associated with the i th ARX model has no zero-pole cancellation and the configuration subspaces of all the ARX models do not contain one another.⁶ In general, we also assume that we do not know the exact orders of the systems but know only certain upper bounds of them, i.e.,

$$\bar{n}_a \geq \max\{n_a(1), \dots, n_a(n)\}, \quad \bar{n}_c \geq \max\{n_c(1), \dots, n_c(n)\}.$$

Very often we do not know the exact number of systems involved either but know only an upper bound of it, i.e., $\bar{n} \geq n$.⁷ In this section, we study how to identify the hybrid ARX system despite these uncertainties.

10.3.1 The Hybrid Decoupling Polynomial

One of the difficulties in identifying hybrid ARX systems is that we do not know the switching sequence λ_t , hence we cannot directly apply the subspace identification technique described in the previous section to each of the n ARX systems. As we will soon see, in fact both the number of subspaces and their dimensions depend not only on the number of systems and their orders but also on the switching sequence. This motivates us to look for relationships between the data $\{\mathbf{x}_t \in \mathbb{R}^D\}$ and the system parameters $\{\mathbf{b}_i \in \mathbb{R}^D\}$ that do not depend on the switching sequence. To this end, recall that for every t there exists a state $\lambda_t = i \in \{1, 2, \dots, n\}$ such that $\mathbf{b}_i^T \mathbf{x}_t = 0$. Therefore, the following polynomial equation must be satisfied by the system parameters and the input/output data for any switching sequence and mechanism (JMLS or PWARX):

$$p_n(\mathbf{x}_t) \doteq \prod_{i=1}^n (\mathbf{b}_i^T \mathbf{x}_t) = 0. \quad (10.16)$$

⁶One way to ensure this is to assume that for all $i \neq j = 1, \dots, n$, $\tilde{H}_i(z)$ and $\tilde{H}_j(z)$ do not have all their zeros and poles in common. That is, there is no ARX system that can simulate another ARX system with a smaller order. However, this is not necessary because two ARX systems can have different configuration spaces even if one system's zeros and poles are a subset of the other's.

⁷This is the case when a particular switching sequence visits only a subset of all the discrete states.

We call this polynomial equation the *hybrid decoupling polynomial* (HDP). In the absence of knowledge about the switching mechanism, the HDP encodes all the information about the system parameters that we can obtain from the input/output data.

The HDP eliminates the discrete state by taking the product of the equations defining each one of the ARX systems. While taking the product is not the only way of algebraically eliminating the discrete state, this leads to an algebraic equation with a very nice algebraic structure. The HDP is simply a homogeneous multivariate polynomial of degree n in D variables

$$p_n(\mathbf{z}) \doteq \prod_{i=1}^n (\mathbf{b}_i^T \mathbf{z}) = 0, \quad (10.17)$$

which can be written linearly in terms of its coefficients as

$$p_n(\mathbf{z}) \doteq \sum h_{n_1, \dots, n_D} z_1^{n_1} \cdots z_D^{n_D} = \mathbf{h}_n^T \nu_n(\mathbf{z}) = 0. \quad (10.18)$$

In equation (10.18), $h_{n_1, \dots, n_D} \in \mathbb{R}$ is the coefficient of the monomial $z_1^{n_1} z_2^{n_2} \cdots z_D^{n_D}$. Obviously, the vector $\mathbf{h}_n = (h_{n_1, \dots, n_D})$ encodes the parameters of all the constituent ARX systems. We will show in the sequel how this vector can be correctly recovered from the data and how the parameters of each individual ARX system can be further retrieved from it.

10.3.2 Identifying the Hybrid Decoupling Polynomial

Let us assume for now that we know the number of systems n . We will show later how to relax this assumption. Since the HDP (10.16) – (10.18) is satisfied by all the data points $\{\mathbf{x}_t\}_{t=1}^T$, we can use it to derive the following linear system on the vector \mathbf{h}_n :

$$L_n(\bar{n}_a, \bar{n}_c) \mathbf{h}_n \doteq \begin{bmatrix} \nu_n(\mathbf{x}_{\max\{\bar{n}_a, \bar{n}_c\}})^T \\ \nu_n(\mathbf{x}_{\max\{\bar{n}_a, \bar{n}_c\}+1})^T \\ \vdots \\ \nu_n(\mathbf{x}_{\max\{\bar{n}_a, \bar{n}_c\}+T-1})^T \end{bmatrix} \mathbf{h}_n = \mathbf{0}_{T \times 1}, \quad (10.19)$$

where $L_n(\bar{n}_a, \bar{n}_c) \in \mathbb{R}^{T \times M_n(D)}$ is the matrix of the input/output data embedded via the Veronese map.

Definition 10.4 (Sufficiently Exciting Switching and Input Sequences). A switching and input sequence $\{\lambda_t, u_t\}$ is called sufficiently exciting for a hybrid ARX system, if the data points $\{\mathbf{x}_t\}$ generated by $\{\lambda_t, u_t\}$ are sufficient to determine the union of the subspaces associated with the constituent ARX systems as an algebraic variety, in the sense of Theorem A.10 of Appendix A.

Given the data matrix $L_n(\bar{n}_a, \bar{n}_c)$ from a sufficiently exciting switching and input sequence, we would like to retrieve the coefficient vector \mathbf{h}_n from its null space. There are two potential difficulties. First, since the maximum orders \bar{n}_a, \bar{n}_c

may not be tight for every constituent ARX system, the null space of $L_n(\bar{n}_a, \bar{n}_c)$ may be more than one-dimensional, as we have known from a single ARX system. Second, even if we know the discrete state for each time, the structure of the data associated with each state is not exactly the same as that of the ARX system itself: Suppose we switch to the i th system at time t_0 , then we have $\mathbf{b}_i^T \mathbf{x}_{t_0} = 0$. However, the vectors \mathbf{b} given in equation (10.10) are no longer orthogonal to \mathbf{x}_{t_0} even if the embedding is redundant for the i th system. In a sense, the regressor at a switching time usually lives in a subspace whose dimension is higher than that of the subspace associated with the ARX model generating the regressor. Therefore, the configuration space of the data $\{\mathbf{x}_t\}$ of an HARX system will *not* exactly be the union of all the subspaces associated with the constituent ARX systems. Let us denote the former as an algebraic variety Z' and the latter as Z . Then in general, we have $Z' \supseteq Z$.

In order to retrieve \mathbf{h}_n uniquely from the data matrix L_n , we need to utilize its additional structure.

Lemma 10.5 (Structure of the Hybrid Decoupling Polynomial). *The monomial associated with the last non-zero entry of the coefficient vector \mathbf{h}_n of the hybrid decoupling polynomial $p_n(z) = \mathbf{h}_n^T \nu_n(z)$ has the lowest degree-lexicographic order in all the polynomials in $\mathfrak{a}(Z) \cap S_n$.⁸*

Proof. Any polynomial of degree n in the ideal $\mathfrak{a}(Z)$ is a superposition of the polynomials $\prod_{i=1}^n (\mathbf{b}_{\sigma(i)}^T z)$ where $\mathbf{b}_{\sigma(i)}$ is a normal vector to the subspace associated with the i th ARX system.⁹ Notice that \mathbf{h}_n is the symmetric tensor of $\mathbf{b}_1, \mathbf{b}_2, \dots, \mathbf{b}_n$ defined in (10.9). For the i th ARX system, the last non-zero entry of the vector \mathbf{b}_i always has the lowest degree-lexicographic order among all normal vectors that are orthogonal to the regressors $\mathbf{z} = \mathbf{x}_t$ associated to the i th system. Therefore, the last non-zero entry of \mathbf{h}_n must have the lowest degree-lexicographic order. \square

Theorem 10.6 (Identifying the Hybrid Decoupling Polynomial). *Suppose that $\{u_t, y_t\}_{t=0}^T$ are the input/output data generated by an identifiable HARX system. Let $L_n^j \in \mathbb{R}^{T \times j}$ be the first j columns of the embedded data matrix $L_n(\bar{n}_a, \bar{n}_c)$, and let*

$$m \doteq \min \{j : \text{rank}(L_n^j) = j - 1\}. \quad (10.20)$$

If T is sufficiently large and the input and switching sequences are sufficiently exciting, then the coefficient vector \mathbf{h}_n of the hybrid decoupling polynomial is given by

$$\mathbf{h}_n = [(\mathbf{h}_n^m)^T, \mathbf{0}_{1 \times (M_n(D)-m)}]^T \in \mathbb{R}^{M_n(D)}, \quad (10.21)$$

⁸The set of (homogeneous) polynomial of degree n .

⁹This is easily verifiable from the fact that the derivatives of the polynomials in $\mathfrak{a}(Z)$ are exactly the normal vectors of the subspaces.

where $\mathbf{h}_n^m \in \mathbb{R}^m$ is the unique vector that satisfies

$$L_n^m \mathbf{h}_n^m = \mathbf{0} \quad \text{and} \quad \mathbf{h}_n^m(1) = 1. \quad (10.22)$$

Proof. Let Z to be the union of the subspaces associated with the n constituent ARX systems. Since the input and switching sequence is sufficiently exciting in the sense of Definition 10.4, according to Theorem A.10 of Appendix A, any polynomial of degree less than and equal to n that vanishes on all the data points must be in the set $\mathfrak{a}(Z) \cap S^n$.¹⁰

From our discussion before the theorem, the configuration space Z' of the data $\{\mathbf{x}_t\}$ associated with the hybrid ARX system is in general a superset of Z . The ideal $\mathfrak{a}'(Z')$ of polynomials that vanish on the configuration space Z' is then a sub-ideal of the ideal $\mathfrak{a}(Z)$ associated with the union of the subspaces. Furthermore, regardless of the switching sequence, the hybrid decoupling polynomial $p_n(\mathbf{z})$ is always in $\mathfrak{a}' \cap S_n \subseteq \mathfrak{a} \cap S_n$. According to Lemma 10.5, the last non-zero term of $p_n(\mathbf{z})$ has the lowest degree-lexicographic order among all polynomials of degree n in \mathfrak{a} , so does it in \mathfrak{a}' . Since every solution $L_n \tilde{\mathbf{h}} = 0$ gives a polynomial $\tilde{p}_n(\mathbf{z}) = \tilde{\mathbf{h}}_n^T \nu_n(\mathbf{z}) \in \mathfrak{a} \cap S_n$ of degree n that vanishes on all data points, the last non-zero entry of \mathbf{h}_n given by (10.21) obviously has the lowest degree-lexicographic order. Therefore, we have $p_n(\mathbf{z}) = \mathbf{h}_n^T \nu_n(\mathbf{z})$. \square

In fact to compute the coefficients \mathbf{h}_n of the hybrid decoupling polynomial, we can do better than checking the rank of the submatrix L_n^j for every $j = 1, 2, \dots$. The following corollary provides one alternative scheme.

Corollary 10.7 (Zero Coefficients of the Decoupling Polynomial). *Consider a set of vectors $\mathbf{b}_i \in \mathbb{R}^D$, $i = 1, \dots, n$. Suppose that one of the \mathbf{b}_i has a maximal number of zeros on its right, and without loss of generality, assume it is*

$$\mathbf{b}_1 = [b_{11}, b_{12}, \dots, b_{1n_1}, 0, \dots, 0]^T, \quad \text{with} \quad b_{1n_1} \neq 0.$$

The multivariate polynomial $p_n(\mathbf{z}) \doteq (\mathbf{b}_1^T \mathbf{z})(\mathbf{b}_2^T \mathbf{z}) \cdots (\mathbf{b}_n^T \mathbf{z})$ has zero coefficients for all the monomials of $\nu_n([z_{n_1+1}, z_{n_1+2}, \dots, z_D])$; but the coefficients cannot all be zeros for the monomials of $\nu_n([z_{n_1}, z_{n_1+1}, \dots, z_D])$.

This corollary allows us to narrow down the range for m (where L_n^j first drops rank) because m must fall between two consecutive values of the following:

$$1, \quad M_n(D) - M_n(D-1), \quad M_n(D) - M_n(D-2), \quad \dots, \quad M_n(D) - 1.$$

Remark 10.8 (Sub-Optimality in the Stochastic Case). *In the stochastic case (i.e., $w_t \neq 0$), we can still solve for \mathbf{h}_n^m in (10.22) in a least-squares sense as the singular vector of L_n^m associated with its smallest singular value, using a similar model selection criterion for m as in Remark 10.1. However, unlike the single system case, the so-found \mathbf{h}_n no longer minimizes the sum of least-square errors $\sum_t w_t^2 = \sum_t (\mathbf{b}_{\lambda_t}^T \mathbf{x}_t)^2$. Instead, it minimizes (in a least-square sense) a*

¹⁰ S^n is the set of polynomials of degree up to n .

“weighted version” of this objective:

$$\sum_t \alpha_t (\mathbf{b}_{\lambda_t}^T \mathbf{x}_t)^2 \doteq \sum_t \prod_{i \neq \lambda_t} (\mathbf{b}_i^T \mathbf{x}_t)^2 (\mathbf{b}_{\lambda_t}^T \mathbf{x}_t)^2, \quad (10.23)$$

where the weight α_t is conveniently chosen to be $\prod_{i \neq \lambda_t} (\mathbf{b}_i^T \mathbf{x}_t)^2$. Such a “softening” of the objective function allows a global algebraic solution. It offers a sub-optimal approximation for the original stochastic objective when the variance of w_t is small. One can use the solution as the initialization for any other (local) nonlinear optimization scheme (such as Expectation Maximization) to further minimize the original stochastic objective.

Notice that in the above theorem, we have assumed that the switching sequence is such that all the ARX systems are sufficiently visited. What if only a subset of the n systems are sufficiently visited? Furthermore, we sometimes do not even know the correct number of systems involved and only know an upper bound for it. The question is whether the above theorem still applies when the degree n we choose for the Veronese embedding is strictly larger than the actually number of systems. This is answered by the following corollary whose proof is straightforward.

Corollary 10.9 (Identifying the Number of ARX Systems). *Let $\{u_t, y_t\}_{t=0}^T$ be the input/output data generated by an HARX system with $n < \bar{n}$ discrete states. If T is sufficiently large and the input and switching sequences are sufficiently exciting, then the vector $\mathbf{h}_{\bar{n}}$ found by Theorem 10.6 is the symmetric tensor product*

$$\mathbf{h}_{\bar{n}} = \text{Sym}(\mathbf{b}_1 \otimes \mathbf{b}_2 \cdots \otimes \mathbf{b}_n \otimes \underbrace{\mathbf{e}_1 \otimes \cdots \otimes \mathbf{e}_1}_{\bar{n}-n}), \quad (10.24)$$

where $\mathbf{e}_1 \doteq [1, 0, \dots, 0]^T \in \mathbb{R}^D$, i.e., $\mathbf{h}_{\bar{n}}$ is the coefficients of the polynomial:

$$p_{\bar{n}}(\mathbf{z}) = \mathbf{h}_{\bar{n}}^T \nu_{\bar{n}}(\mathbf{z}) = (\mathbf{b}_1^T \mathbf{z})(\mathbf{b}_2^T \mathbf{z}) \cdots (\mathbf{b}_{\bar{n}}^T \mathbf{z}) z^{\bar{n}-n}. \quad (10.25)$$

Therefore, even if we may over-estimate the number of constituent systems or the switching sequence does not visit all the systems, the solution given by Theorem 10.6 will simply treat the nonexistent (or not visited) systems as if they had zero order¹¹ and the information about the rest of the systems will be conveniently recovered.

10.3.3 Identifying System Parameters and Discrete States

Theorem 10.6 allows us to determine the hybrid decoupling polynomial $p_n(\mathbf{z}) = \mathbf{h}_n^T \nu_n(\mathbf{z})$, from input/output data $\{u_t, y_t\}_{t=0}^T$. The rest of the problem is to recover the system parameters $\{\mathbf{b}_i\}_{i=1}^n$ from \mathbf{h}_n . To this end, recall from Chapter 4 that

¹¹That is, the coefficient vector $\mathbf{b} = \mathbf{e}_1$ corresponds to the “system” $y_t = 0$ with $n_a = n_c = 0$, which is a trivial ARX system.

given \mathbf{h}_n one can recover the model parameters by looking at the partial derivative of $p_n(\mathbf{z})$ given in (10.17)

$$Dp_n(\mathbf{z}) \doteq \frac{\partial p_n(\mathbf{z})}{\partial \mathbf{z}} = \sum_{i=1}^n \prod_{\ell \neq i} (\mathbf{b}_\ell^T \mathbf{z}) \mathbf{b}_i. \quad (10.26)$$

If \mathbf{z} belongs to the hyperplane $\mathcal{H}_i = \{\mathbf{z} : \mathbf{b}_i^T \mathbf{z} = 0\}$, then, since the 1st entry of \mathbf{b}_i by definition is equal to one, after replacing $\mathbf{b}_i^T \mathbf{z} = 0$ into (10.26) we obtain

$$\mathbf{b}_i = \left. \frac{Dp_n(\mathbf{z})}{\mathbf{e}_1^T Dp_n(\mathbf{z})} \right|_{\mathbf{z} \in \mathcal{H}_i} \in \mathbb{R}^D, \quad (10.27)$$

where $\mathbf{e}_1 = [1, 0, \dots, 0]^T \in \mathbb{R}^D$. Therefore, we can estimate the system parameters directly from the derivatives of $p_n(\mathbf{z})$ at a collection of n points $\{\mathbf{z}_i \in \mathcal{H}_i\}_{i=1}^n$ lying on the n hyperplanes, respectively.

In order to find the set of points $\{\mathbf{z}_i \in \mathcal{H}_i\}_{i=1}^n$, let us consider a line with base point \mathbf{z}_0 and direction \mathbf{v} , $\mathcal{L} = \{\mathbf{z}_0 + \alpha \mathbf{v}, \alpha \in \mathbb{R}\}$. If $\mathbf{z}_0 \neq 0$, \mathbf{z}_0 is not parallel to \mathbf{v} , and $\mathbf{b}_i^T \mathbf{v} \neq 0$, then the line \mathcal{L} in general intersects the n hyperplanes $\cup_{i=1}^n \mathcal{H}_i = \{\mathbf{z} : p_n(\mathbf{z}) = 0\}$ at n distinct points

$$\mathbf{z}_i = \mathbf{z}_0 + \alpha_i \mathbf{v} \in \mathcal{H}_i \cap \mathcal{L}, \quad i = 1, \dots, n, \quad (10.28)$$

where $\{\alpha_i\}$ are the roots of the univariate polynomial

$$q_n(\alpha) = p_n(\mathbf{z}_0 + \alpha \mathbf{v}). \quad (10.29)$$

We are left with choosing the parameters \mathbf{x}_0 and \mathbf{v} for the line \mathcal{L} . The base point \mathbf{z}_0 can be chosen as any nonzero vector in \mathbb{R}^D . Given \mathbf{z}_0 , the direction \mathbf{v} must be chosen not parallel to \mathbf{z}_0 and such that $\mathbf{b}_i^T \mathbf{v} \neq 0$, for all $i = 1, \dots, n$. Since the latter constraint is equivalent to $p_n(\mathbf{v}) \neq 0$, and p_n is known, we can immediately choose \mathbf{v} even though we do not know the system parameters $\{\mathbf{b}_i\}_{i=1}^n$.

Be aware that if we have chosen for the Veronese embedding a number \bar{n} that is strictly larger than n , the polynomial $p_{\bar{n}}(\mathbf{z})$ will be of the form $(\mathbf{b}_1^T \mathbf{z})(\mathbf{b}_2^T \mathbf{z}) \cdots (\mathbf{b}_n^T \mathbf{z}) z_1^{\bar{n}-n}$. Then the line \mathcal{L} will have only $n+1$ intersections with the n hyperplanes $\mathcal{H}_1, \dots, \mathcal{H}_n$ and the hyperplane $\mathcal{H}_0 \doteq \{\mathbf{z} : \mathbf{e}_1^T \mathbf{z} = z_1 = 0\}$. The intersection $\mathbf{z}_0 = \mathcal{H}_0 \cap \mathcal{L}$ has a multiplicity of $\bar{n}-n$; and $Dp_{\bar{n}}(\mathbf{z}_0) \sim \mathbf{e}_1$ if $\bar{n}-n = 1$ or $Dp_{\bar{n}}(\mathbf{z}_0) = 0$ if $\bar{n}-n > 1$. We have essentially proven the following theorem.

Theorem 10.10 (Identifying the Constituent System Parameters). *Given the input/output data $\{u_t, y_t\}_{t=0}^T$ generated by an HARX system with n discrete states, the system parameters $\{\mathbf{b}_i\}_{i=1}^n$ can be computed from the the hybrid decoupling polynomial $p_{\bar{n}}(\mathbf{z}) = \mathbf{h}_{\bar{n}}^T \nu_{\bar{n}}(\mathbf{z})$ for any $\bar{n} \geq n$ as follows:*

1. Choose $\mathbf{z}_0 \neq 0$ and \mathbf{v} such that $\mathbf{v} \neq \gamma \mathbf{z}_0$ and $p_{\bar{n}}(\mathbf{v}) \neq 0$.
2. Solve for the \bar{n} roots $\{\alpha_i\}_{i=1}^{\bar{n}}$ of $q_{\bar{n}}(\alpha) = p_{\bar{n}}(\mathbf{z}_0 + \alpha \mathbf{v}) = 0$.

3. For all the roots $\mathbf{z}_i = \mathbf{z}_0 + \alpha_i \mathbf{v}$ with $z_1 \neq 0$, compute the system parameters $\{\mathbf{b}_i\}_{i=1}^n$ as

$$\mathbf{b}_i = \frac{Dp_{\bar{n}}(\mathbf{z}_i)}{\mathbf{e}_1^T Dp_{\bar{n}}(\mathbf{z}_i)} \in \mathbb{R}^D, \quad i = 1, 2, \dots, n. \quad (10.30)$$

Remark 10.11 (Alternative Ways of Identifying $\{\mathbf{b}_i\}_{i=1}^n$ from Noisy Data). In the presence of noise, we can still estimate the normal vectors $\{\mathbf{b}_i\}_{i=1}^n$ as in Theorem 10.10. However, the quality of the estimates will depend on the choice of the parameters \mathbf{z}_0 and \mathbf{v} . In this case, one can choose multiple $(\mathbf{z}_0, \mathbf{v})$ satisfying the above conditions, obtain the system parameters for each choice, and let $\{\mathbf{b}_i\}_{i=1}^n$ be the parameters that better reconstruct \mathbf{h}_n . Alternatively, one can directly choose $\{\mathbf{z}_i\}_{i=1}^n$ from points in the data set that fit the decoupling polynomial in an optimal way. That allows us to bypass the problem of solving the (real) roots of the real polynomial $q_n(\alpha)$.

Once the system parameters $\{\mathbf{b}_i\}_{i=1}^n$ are recovered, we can then reconstruct the orders $n_a(i), n_c(i)$ of each constituent ARX system as well as the discrete state trajectory $\{\lambda_t\}$ from the input/output data $\{\mathbf{x}_t\}_{t=0}^T$. Notice that for each time t there exists a generally unique i such that $\mathbf{b}_i^T \mathbf{x}_t = 0$. Therefore, the discrete state λ_t can be easily identified as:

$$\lambda_t = \arg \min_{i=1, \dots, n} (\mathbf{b}_i^T \mathbf{x}_t)^2. \quad (10.31)$$

There will be ambiguity in the value of λ_t only if \mathbf{x}_t happens to be at (or close to) the intersection of more than one subspace associated to the constituent ARX systems. However, the set of all such points is a zero measure set of the variety $Z \subseteq \{\mathbf{z} : p_n(\mathbf{z}) = 0\}$.

10.3.4 The Basic Algorithm and its Extensions

Based on the results that we have derived so far, we summarize the main steps for solving the identification of an HARX system (Problem 10.1) as the following Algorithm 10.1. Notice that the algorithm is different from the general-purpose GPCA algorithm given in Chapter 4. By utilizing the structure in the system parameters $\{\mathbf{b}_i\}$ and subsequently in their symmetric tensor product \mathbf{h}_n , the algorithm guarantees that the so found polynomial p_n is the desired hybrid decoupling polynomial.

Different Embedding Orders.

The order of stacking $\{y_t\}$ and $\{u_t\}$ in the vector \mathbf{x}_t in (10.8) is more efficient for the algorithm when $n_a(i)$ are approximately the same for all the constituent systems and $n_c(i)$ are much smaller than $n_a(i)$. However, if $n_a(i)$ are rather different for different systems and $n_c(i)$ and $n_a(i)$ are roughly the same, the following ordering in time t

$$\mathbf{x}_t \doteq [y_t, y_{t-1}, u_{t-1}, y_{t-2}, u_{t-2}, \dots, y_{t-\bar{n}_a}, u_{t-\bar{n}_a}]^T \in \mathbb{R}^D \quad (10.32)$$

Algorithm 10.1 (Identification of an SISO HARX System).

Given the input/output data $\{y_t, u_t\}$ from a sufficiently excited hybrid ARX system, and the upper bound on the number \bar{n} and maximum orders (\bar{n}_a, \bar{n}_c) of its constituent ARX systems:

1. **Veronese Embedding.** Construct the data matrix $L_{\bar{n}}(\bar{n}_a, \bar{n}_c)$ via the Veronese map based on the given number \bar{n} of systems and the maximum orders (\bar{n}_a, \bar{n}_c) .
 2. **Hybrid Decoupling Polynomial.** Compute the coefficients of the polynomial $p_{\bar{n}}(z) \doteq h_{\bar{n}}^T \nu_{\bar{n}}(z) = \prod_{i=1}^n (b_i^T z)^{\bar{n}-n} = 0$ from the data matrix $L_{\bar{n}}$ according to Theorem 10.6 and Corollary 10.9. In the stochastic case, comply with Remarks 10.1 and 10.8.
 3. **Constituent System Parameters.** Retrieve the parameters $\{b_i\}_{i=1}^n$ of each constituent ARX system from $p_{\bar{n}}(z)$ according to Theorem 10.10. In the noisy case, comply with Remark 10.11.
 4. **Key System Parameters.** The correct number of system n is the number of $b_i \neq e_1$; The correct orders $n_a(i), n_c(i)$ are determined from such b_i according to their definition (10.9); The discrete state λ_t for each time t is given by equation (10.31).
-

results in less non-zero leading coefficients in h_n . Thus the above algorithm becomes more efficient. Nevertheless, if all the systems have the same $n_a = n_c$, both embeddings have the same efficiency.

Inferring the Switching Mechanisms.

Once the system parameters and the discrete state have been identified, the problem of estimating the switching mechanisms, e.g., the partition of the state space for PWARX or the parameters of the jump Markov process for JMLS, becomes a simpler problem. We refer interested readers to [Bemporad et al., 2003, Ferrari-Trecate et al., 2003] for specific algorithms.

10.4 Simulations and Experiments

In this section we evaluate the performance of the proposed algorithm with respect to the model orders and the amount of noise. We also present experiments on real data from a component placement process in a pick-and-place machine.

10.4.1 Error in the Estimation of the Model Parameters

Consider the PWARX system taken from [Niessen and A.Juloski, 2004]

$$y_t = \begin{cases} 0.5u_{t-1} + 0.5 + w_{t-1} & \text{if } u_{t-1} \in [-2.5, 0], \\ -u_{t-1} + 2 + w_{t-1} & \text{if } u_{t-1} \in (0, 2.5]. \end{cases} \quad (10.33)$$

The input sequence u_t consists of 100 points, 80% uniformly distributed in $[-2.5, 2.5]$ and 20% uniformly distributed in $[0.85, 1.15]$. The noise is $w_t \stackrel{\text{i.i.d.}}{\sim} \mathcal{N}(0, 0.005)$. The error between the estimated parameters $\hat{\mathbf{b}}$ and the true parameters \mathbf{b} is defined as

$$\text{error} = \max_{i=1, \dots, m.} \min_{j=1, \dots, n.} \frac{\|\hat{\mathbf{b}}_i - \mathbf{b}_j\|}{\|[\mathbf{0}_{(D-1) \times 1} \ I_{D-1}] \mathbf{b}_j\|}.$$

We applied our algorithm with known parameters $n = 2$, $n_a = 0$ and $n_c = 1$. Our algorithm gives an estimate for the ARX model parameters of $[0.5047, 0.5102]^T$ and $[-0.9646, 1.9496]^T$, which corresponds to an error of 0.0276. Table 10.1 compares our results with those reported in [Niessen and A.Juloski, 2004] for the algorithms of [Ferrari-Trecate et al., 2003] and [Bemporad et al., 2003]. Notice that our algorithm provides a purely algebraic solution to the problem which does not perform iterative refinement. Nevertheless it provides a comparable error with the other algorithms which are based on iterative refinement.

Table 10.1. Comparison of error in the estimation of the model parameters.

Algorithms	Errors
Ferrari-Trecate et. al.	0.0045
Bemporad et. al.	0.0334
Algorithm 10.1	0.0276

10.4.2 Error as a Function of the Model Orders

Consider the PWAR system taken from [Niessen and A.Juloski, 2004]

$$y_t = \begin{cases} 2y_{t-1} + 0u_{t-1} + 10 + w_t & \text{if } y_{t-1} \in [-10, 0], \\ -1.5y_{t-1} + 0u_{t-1} + 10 + w_t & \text{if } y_{t-1} \in (0, 10], \end{cases} \quad (10.34)$$

with initial condition $y_0 = -10$, input $u_t \stackrel{\text{i.i.d.}}{\sim} \mathcal{U}(-10, 10)$ and noise $w_t \stackrel{\text{i.i.d.}}{\sim} \mathcal{N}(0, 0.01)$.

We applied our algorithm¹² with known number of models $n = 2$, but unknown model orders (n_a, n_c) . We evaluated the performance of our algorithm as a function of the orders (n_a, n_c) . We used a fixed value for (n_a, n_c) and search for the polynomial in the null space of $L_n(n_a, n_c)$ with the smallest degree-lexicographic order. We repeated the experiment for multiple values of $n_a = 1, \dots, 4$ and $n_c = 1, \dots, 10$, to evaluate the effectiveness of equation (10.12) at finding the “correct” null space of $L_n(n_a, n_c)$. Figure 10.2 shows the results for $\kappa = 10^{-5}$. Notice that for all the range of values of n_a and n_c , the algorithm gives an error that is very close to the theoretical bound of 0.01 (the noise variance).

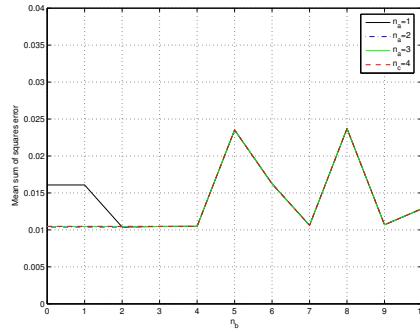


Figure 10.2. Mean sum of squares error for various orders of the ARX models.

For the correct system orders $n_a = 1$ and $n_c = 0$, the estimates of the ARX model parameters from our algorithm are $[1.9878, 0, 10.0161]^T$ and $[-1.4810, 0, 10.0052]^T$, which have an error of 0.0020. These results are significantly better than those reported in [Niessen and A.Juloski, 2004] for the Ferrari-Trecate and Bemporad’s algorithms.

10.4.3 Error as a Function of Noise

Consider the PWAR model taken from [Niessen and A.Juloski, 2004]

$$y_t = \begin{cases} 2u_{t-1} + 10 + w_t & \text{if } u_{t-1} \in [-10, 0], \\ -1.5u_{t-1} + 10 + w_t & \text{if } u_{t-1} \in (0, 10], \end{cases} \quad (10.35)$$

with input $u_t \stackrel{\text{i.i.d.}}{\sim} \mathcal{U}(-10, 10)$ and noise $w_t \stackrel{\text{i.i.d.}}{\sim} \mathcal{N}(0, \sigma_\eta^2)$. We run our algorithm with $n = 2$, $n_a = 0$ and $n_c = 1$ for 10 different values of σ_η and compute the mean and the variance of the error in the estimated model parameters, as shown in Figure 10.3. The algorithm estimates the parameters with an error of less than 3.7% for the levels of noise considered. Again, the errors provided by the

¹²Since here the system is an affine ARX model with a constant input, we need to slightly modify our algorithm by using the homogeneous representation for the regressor x_t , i.e., appending an entry of “1.”

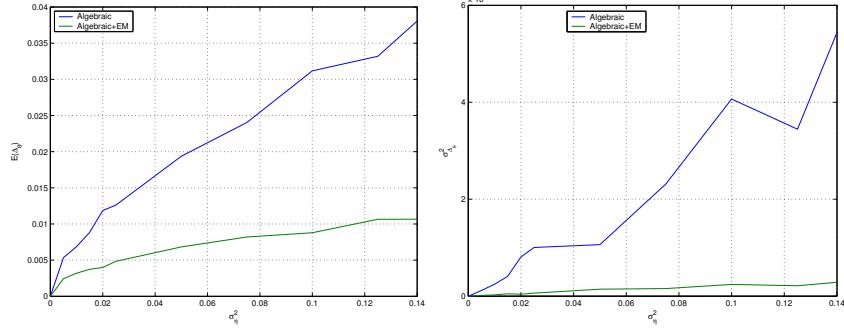


Figure 10.3. Means (left) and variances (right) of the error in the estimation of the model parameters for different levels of noise. Blue curves are for the purely algebraic Algorithm 10.1; Green curves are for the EM algorithm initialized with the solutions from Algorithm 10.1.

purely algebraic algorithm (Algorithm 10.1) without any iterative refinement are comparable to those of the Ferrari-Trecate and Bemporad's algorithms reported in [Niessen and A.Juloski, 2004] which are about $2 \sim 3\%$. Furthermore, if we use the solutions offered by our algebraic algorithm to initialize other iterative refinement algorithms such as the Expectation and Maximization (EM) algorithm, then the error is reduced significantly to about 1% (see Figure 10.3 left).

10.4.4 Experimental Results on Test Datasets

We applied our algorithm with $n = n_a = n_c = 2$ to four datasets of $T = 60,000$ measurements from a component placement process in a pick-and-place machine [Juloski et al., 2004].¹³

Since the methods of [Ferrari-Trecate et al., 2003] and [Bemporad et al., 2003] cannot handle large datasets, for comparison purposes we first report results on a down-sampled dataset of 750 points.¹⁴ The 750 points are separated in two overlapping groups of points. The first 500 points are used for identification, and the last 500 points are used for validation. Table 10.2 shows the average sum of squared residuals (SSR) – one step ahead prediction errors, and the average sum of squared simulation errors (SSE) obtained by our method for all four datasets, as well as the SSE of Ferrari-Trecate's and Bemporad's algorithm for the first dataset as reported in [Niessen and A.Juloski, 2004]. Figure 10.4 shows the true and simulated outputs for dataset 1.

We now report the results of our algorithm tested on the entire datasets. We split the 60,000 measurements in two groups of 30,000 points each. The first 30,000 are used for identification and the last 30,000 for simulation. Table 10.3 shows the average sum of squared residual error (SSR) and the average sum of squared

¹³We thank Prof. A. Juloski for providing us with the datasets

¹⁴We take one out of every 80 samples.

Table 10.2. Training and simulation errors for down-sampled datasets.

Dataset	n	n_a	n_c	GPCA SSR	SSE	F-T SSE	Bem. SSE
1	2	2	2	0.0803	0.1195	1.98	2.15
2	2	2	2	0.4765	0.4678	N/A	N/A
3	2	2	2	0.6692	0.7368	N/A	N/A
4	2	2	2	3.1004	3.8430	N/A	N/A

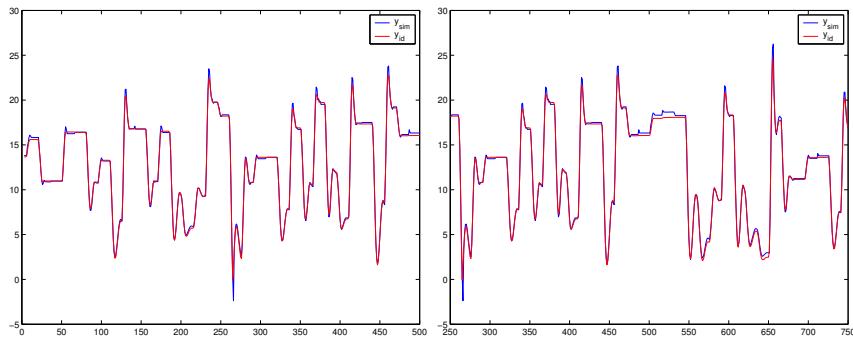


Figure 10.4. Training and simulation sequences for down-sampled dataset 1.

simulation error (SSE) obtained by our method for all four datasets. Figure 10.5 shows the true and simulated outputs for dataset 1.

Overall, the algorithm demonstrates a very good performance in all four datasets. The running time of a MATLAB implementation of our algorithm is 0.15 second for the 500 data points and 0.841 second for 30,000 data points.

Table 10.3. Training and simulation errors for complete datasets.

Dataset	n	n_a	n_c	SSR	SSE
1 with all points	2	2	2	$4.9696 \cdot 10^{-6}$	$5.3426 \cdot 10^{-6}$
2 with all points	2	2	2	$9.2464 \cdot 10^{-6}$	$7.9081 \cdot 10^{-6}$
3 with all points	2	2	2	$2.3010 \cdot 10^{-5}$	$2.5290 \cdot 10^{-5}$
4 with all points	2	2	2	$7.5906 \cdot 10^{-6}$	$9.6362 \cdot 10^{-6}$

10.5 Bibliographic Notes

Work on identification (and filtering) of hybrid systems first appeared in the seventies; a review of the state of the art as of 1982 can be found in [Tugnait, 1982]. After a decade-long hiatus, the problem has recently been enjoying considerable interest [Bemporad et al., 2000, Ezzine and Haddad, 1989, Sun et al., 2002, Szigeti, 1992, Vidal et al., 2002a, Vidal et al., 2003a]. Much related work has also

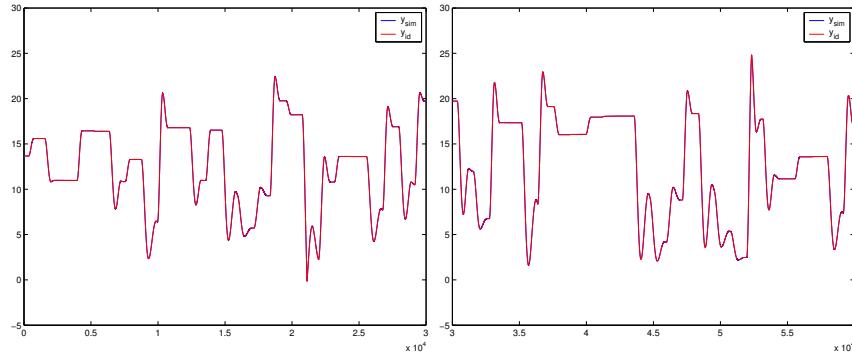


Figure 10.5. Training and simulation sequences for complete datasets – the simulated and the identified sequences overlap almost exactly.

appeared in the machine-learning community [Billio et al., 1999, Blake et al., 1999, Doucet et al., 2000, Ghahramani and Hinton, 1998, Murphy, 1998, Pavlovic et al., 1999].

When the model parameters and the switching mechanism are *known*, the identification problem reduces to the design of observers for the hybrid state [Alessandri and Coletta, 2001, Balluchi et al., 2002, Ferrari-Trecate et al., 2002, Vecchio and Murray, 2004], together with the study of observability conditions under which hybrid observers operate correctly [Babaali and Egerstedt, 2004, Bemporad et al., 2000, Collins and Schuppen, 2004, Vidal et al., 2002a, Vidal et al., 2003a, Hwang et al., 2003, Santis et al., 2003].

When the model parameters and the switching mechanism are both *unknown*, the identification problem becomes much more challenging. Existing work has concentrated on the class of piecewise affine and piecewise ARX systems, i.e., models in which the regressor space is partitioned into polyhedra with affine or ARX submodels for each polyhedron. For instance, [Ferrari-Trecate et al., 2003] assumes that the number of systems is known, and proposes an identification algorithm that combines clustering, regression and classification techniques; [Bemporad et al., 2001] solves for the model parameters and the partition of the state space using mixed-integer linear and quadratic programming; [Bemporad et al., 2003] uses a greedy approach for partitioning a set of infeasible inequalities into a minimum number of feasible subsystems, and then iterates between assigning data points to models and computing the model parameters.

Chapter 11

Switched ARMA Models

Chapter 12

Extensions to Arrangements of Nonlinear Surfaces

In the previous chapter, we have studied how to model data with subspace arrangements and fit piecewise linear models to the data. In practice, data may not always be perfectly piecewise linear, and the piecewise linear models can only approximate the data to certain extent. In many applications, we may actually have information about the form of nonlinearity in the data (e.g., the data lies on quadratic surfaces or certain nonlinear manifolds). It just seems wiser to exploit such nonlinearity than throw the information away. In this chapter, we will extend the GPCA techniques to incorporate nonlinear models, especially quadratic ones.¹ As in the previous chapter, our focus will be on the algebraic and geometric aspect of the problem. Although the proposed methods are numerically stable and tolerate moderate noises in the data, we will leave noise analysis and robustness issues to the next chapter.

12.1 Arrangements of Quadratic Surfaces

In this section, we extend the pool of models to arrangements of both linear subspaces and quadratic surfaces possibly of different dimensions.² As GPCA, we like to simultaneously segment the data into multiple groups and determine a linear or quadratic model for each group. Specifically, we show how the basic al-

¹For readers who are not interested in nonlinear models, they can simply skip this chapter without losing any continuity.

²Linear subspaces can be viewed as degenerate quadratic surfaces.

gebraic techniques that we developed in the previous chapters for GPCA can be extended to this case. To distinguish with GPCA, we refer to the new method as *generalized principal surface analysis* (GPSA).

In GPCA, we have seen that the derivatives of the fitting polynomials play a crucial role in segmenting the data and determining the local dimension. As we will see in this section, to segment and extract quadratic models, one needs to study the properties of *both* the derivatives and Hessians, the second-order derivatives, of the fitting polynomials. These properties lead to a rich spectrum of algebraic signatures for the data points which allow us to effectively segment them into different linear or quadratic models.

12.1.1 Problem Formulation

Notice that a quadratic surface is in general described by a quadratic equation of the form:

$$\mathbf{y}^T B \mathbf{y} + 2c^T \mathbf{y} + d = 0, \quad (12.1)$$

where B is a symmetric matrix. Define the homogeneous coordinates of \mathbf{y} as $\mathbf{x} \doteq [\begin{smallmatrix} \mathbf{y} \\ 1 \end{smallmatrix}]$ and we can rewrite the above equation as

$$\mathbf{x}^T A \mathbf{x} \doteq [\mathbf{y}^T, 1] \begin{bmatrix} B & c \\ c^T & d \end{bmatrix} \begin{bmatrix} \mathbf{y} \\ 1 \end{bmatrix} = 0. \quad (12.2)$$

Therefore, using the homogeneous representation, we can always represent a quadratic surface by a homogeneous quadratic equation (in a space of one dimension higher).

Under this notation, we define the subject of interest for this section. A “quadratic surface” of dimension d^q in \mathbb{R}^D is defined to be³

$$S^q \doteq \{\mathbf{x} : \mathbf{x}^T A_i \mathbf{x} = 0, i = 1, \dots, D - d^q\}, \quad (12.3)$$

where $A_i \in \mathbb{R}^{D \times D}$, $i = 1, \dots, D - d^q$ are a set of linearly independent symmetric matrices (i.e., for any A_i , it cannot be expressed as the linear combination of other matrices A_j 's).⁴ To avoid degenerate cases, we further require that A_i are *not* semi-definite.⁵ The superscript “ q ” indicates “quadratic.” For convenience, we denote the codimension as $r^q \doteq D - d^q$.

As before, we represent a d^l -dimensional ($1 \leq d^l \leq D$) linear subspace by

$$S^l \doteq \{\mathbf{x} : B^T \mathbf{x} = 0, B \in \mathbb{R}^{D \times (D - d^l)}\}.$$

³Note that our notion of “quadratic surfaces” is more general than the traditional definition. A quadratic surface here will be an algebraic surface that satisfies a set of quadratic equations. Such a surface, strictly speaking, could be an algebraic surface of order higher than two.

⁴The representation of S^q by A_i 's is not unique since it can also be described by any other set of linearly independent symmetric matrices $A'_1, A'_2, \dots, A'_{D-d^q}$ whose span is the same as that of A_i 's.

⁵If A is either positive or negative semi-definite, then we have $A = BB^T$ or $A = -BB^T$ for some B . Then $\mathbf{x}^T A \mathbf{x} = \mathbf{x}^T BB^T \mathbf{x} = \|B^T \mathbf{x}\|^2 = 0$ defines a linear subspace.

The superscript “ l ” indicates “linear.” We denote the codimension as $r^l \doteq D - d^l$.

We then formulate the problem of segmenting arrangements of linear subspaces and quadratic surfaces as the following:

Problem 12.1 (Segmentation of Linear Subspaces and Quadratic Surfaces).

Let $\mathbf{X} = \{\mathbf{x}_i\} \subset \mathbb{R}^D$ be the homogeneous coordinates of a set of N data points that are sampled from m unknown quadratic varieties S_1^q, \dots, S_m^q of dimensions d_i^q ($i = 1, \dots, m$) and n linear varieties S_1^l, \dots, S_n^l with dimensions d_j^l ($j = 1, \dots, n$). Assume that the i th quadratic surface is defined by $D - d_i^q$ symmetric matrices A_{ik} ($k = 1, \dots, D - d_i^q$). Similarly, the j th linear subspace is defined by a matrix $B_j = [\mathbf{b}_{j1}, \dots, \mathbf{b}_{j(D-d_j^l)}] \in \mathbb{R}^{D \times (D - d_j^l)}$. From the samples, we want to

1. determine the number of varieties m and n and segment the data points in \mathbf{X} into the $m + n$ subspaces and surfaces;
 2. identify, for each subspace and surface, the corresponding matrix (matrices) A_i or B_i .
-

12.1.2 Properties of the Fitting Polynomials

As in GPCA, we first fit all the data points in \mathbf{X} with high order polynomial(s). For any data point $\mathbf{x} \in \mathbf{X}$, since $\mathbf{x} \in (\cup_{i=1}^m S_i^q) \cup (\cup_{j=1}^n S_j^l)$, \mathbf{x} satisfies a polynomial equation of the form:

$$p(\mathbf{x}) = f(\mathbf{x}) \cdot g(\mathbf{x}) = 0, \quad (12.4)$$

where $f(\mathbf{x}) = \prod_{i=1}^m \mathbf{x}^T A_{ik(i)} \mathbf{x}$ for some $k(i) \in \{1, \dots, r_i^q\}$ and $g(\mathbf{x}) = \prod_{j=1}^n \mathbf{b}_{jk(j)}^T \mathbf{x}$ for some $k(j) \in \{1, \dots, r_j^l\}$. Then $p(\mathbf{x})$ is a $(2m+n)$ -degree homogeneous polynomial in the entries of \mathbf{x} , and it is one of the polynomials of the lowest degree that can fit all the data points in \mathbf{X} [Harris, 1992, ?]. The null space of the data matrix L_{2m+n} (obtained via the Veronese map of order $2m+n$) contains s vectors c_1, \dots, c_s . We then have s polynomial(s) $p_i(\mathbf{x}) = \nu_{2m+n}^T(\mathbf{x}) c_i$ ($i = 1, \dots, s$) that fit all the data points in \mathbf{X} . In general, these polynomials may not be factorable, and instead they are linear combinations of the factorable ones in (12.4).

Given the polynomials $p_i(\mathbf{x})$ ($i = 1, \dots, s$) and the data set \mathbf{X} , we need to extract the information about the subspaces and surfaces so that we can segment the data set. Ideally, this can be solved by factoring the polynomials $p_i(\mathbf{x})$ into its irreducible factors. However, as we have discussed above, the polynomials obtained from the null space of the Veronese data matrix L may not be factorable. As in the GPCA, although it is the factors that we are after, it turns out that we do not have to perform the algebraic factorization *per se*.

We can avoid the difficulty with the factorization by utilizing the relationship between the polynomials $p_i(\mathbf{x})$ and the given data set \mathbf{X} . In particular, we need to understand certain algebraic signatures that one can derive from $p_i(\mathbf{x})$ at every data point. These signatures are associated with the first and second order derivatives of $p_i(\mathbf{x})$. We list below some of the relevant properties of the derivatives. They can all be verified by applying the fact that $p_i(\mathbf{x})$ is a linear combination of the factorable polynomials in (12.4).

Proposition 12.1 (Derivatives of the Fitting Polynomials). *Let $p_i(\mathbf{x})$ be a polynomial that fits \mathbf{X} . Then the derivatives of $p_i(\mathbf{x})$ at $\mathbf{x} \in \mathbf{X}$ are given by:*

If \mathbf{x} belongs to a quadratic surface S^q defined by $A_j, j = 1, \dots, r^q (= D - d^q)$, then

$$\nabla p_i(\mathbf{x}) = \sum_{j=1}^{r^q} 2\alpha_{ij}(\mathbf{x})A_j \mathbf{x} \in \mathbb{R}^D, \quad (12.5)$$

$$H_{p_i}(\mathbf{x}) = \sum_{j=1}^{r^q} \left[2\alpha_{ij}(\mathbf{x})A_j + A_j \mathbf{x} \nabla_{\alpha_{ij}}^T(\mathbf{x}) + \nabla_{\alpha_{ij}}(\mathbf{x})(A_j \mathbf{x})^T \right] \in \mathbb{R}^{D \times D} \quad (12.6)$$

where $\alpha_{ij}(\mathbf{x})$ are scalar functions of \mathbf{x} that contain polynomial factors from other surfaces or subspaces.

If \mathbf{x} belongs to a linear subspace S^l defined by $\mathbf{b}_j, j = 1, \dots, r^l (= D - d^l)$. Then

$$\nabla p_i(\mathbf{x}) = \sum_{j=1}^{r^l} \beta_{ij}(\mathbf{x})\mathbf{b}_j \in \mathbb{R}^D, \quad (12.7)$$

$$H_{p_i}(\mathbf{x}) = \sum_{j=1}^{r^l} \left[\mathbf{b}_j \nabla_{\beta_{ij}}^T(\mathbf{x}) + \nabla_{\beta_{ij}}(\mathbf{x})\mathbf{b}_j^T \right] \in \mathbb{R}^{D \times D}, \quad (12.8)$$

where $\beta_{ij}(\mathbf{x})$ are scalar functions of \mathbf{x} which contains polynomial factors from other subspaces and surfaces.

If \mathbf{x} is on the intersection of more than one subspace or surface, then

$$\nabla p_i(\mathbf{x}) = 0 \in \mathbb{R}^D. \quad (12.9)$$

Proposition 12.2 (Surface Normals from the Derivatives). *Let $\mathbf{x} \in \mathbf{X}$ be a general point in a d -dimensional subspace or surface S but not at any intersection. Then the matrix $\nabla_p(\mathbf{x}) = [\nabla_{p_1}(\mathbf{x}), \dots, \nabla_{p_s}(\mathbf{x})] \in \mathbb{R}^{D \times s}$ has rank $r = D - d$. Let the singular value decomposition (SVD) of $\nabla_p(\mathbf{x})$ be $\nabla_p(\mathbf{x}) = U\Sigma V^T$ with U and V being orthogonal matrices and Σ a diagonal matrix. The first r columns of U gives a set of orthonormal vectors $\mathbf{n}_1(\mathbf{x}), \dots, \mathbf{n}_r(\mathbf{x})$ to S at \mathbf{x} .*

Proof. Let $\mathbf{t}(\mathbf{x}) \in \mathbb{R}^D$ be any tangent vector to S at \mathbf{x} and let $\gamma(u) : \mathbb{R} \rightarrow \mathbb{R}^D$ be any curve in S such that $\gamma(0) = \mathbf{x}$ and $\gamma'(0) = \mathbf{t}(\mathbf{x})$. Then for any fitting polynomial $p_i(\mathbf{x})$, we have

$$p_i(\gamma(u)) = 0, \quad \forall u \in \mathbb{R}.$$

Differentiate the above equation with respect to u and evaluate the derivative at $u = 0$. We obtain

$$\nabla_{p_i}^T(\mathbf{x})\mathbf{t}(\mathbf{x}) = 0.$$

That is, the derivative of every fitting polynomial is orthogonal to all tangent vectors. The rest of the proposition then follows. \square

This proposition allows one to compute the normal vectors to the surface from the fitting polynomials. As we have already seen in the case of GPCA, the normal vectors are already sufficient to segment linear subspaces since they are invariant for each subspace. However, this is no longer true for a quadratic surface and additional information from higher-order derivatives is needed in this case.

Examining the Hessian (12.6) associated to a quadratic surface, we notice that the first term is indeed the Hessian of the factor for the surface itself, but the second and third terms depend on derivatives of factors for other subspaces and surfaces in the arrangement. This prevents us from directly using the Hessians to segment the data to different surfaces.

One solution to resolve this difficulty is to use, instead of the Hessian, the “contraction” of the Hessian by the tangent vectors to the surface.⁶

Definition 12.3 (Contraction of Hessians). *Let $T(\mathbf{x}) \doteq [\mathbf{t}_1, \dots, \mathbf{t}_d] \in \mathbb{R}^{D \times d}$ be a matrix whose columns are orthonormal tangent vectors to a subspace or a surface at a point \mathbf{x} . Then for every fitting polynomial $p_i(\mathbf{x})$, the contraction of $H_{p_i}(\mathbf{x})$ by $T(\mathbf{x})$ is defined to be the symmetric matrix*

$$C^i(\mathbf{x}) \doteq T(\mathbf{x})^T H_{p_i}(\mathbf{x}) T(\mathbf{x}) \in \mathbb{R}^{d \times d}. \quad (12.10)$$

Based on the above definition and Proposition 12.1, we have

Proposition 12.4 (Properties of Contractions). *Given an arrangement of linear subspaces and quadratic surfaces, if a point \mathbf{x} is on a quadratic surface defined by A_j ($j = 1, \dots, r$), then*

$$C^i(\mathbf{x}) = \sum_{j=1}^r 2\alpha_{ij}(\mathbf{x}) T(\mathbf{x})^T A_j T(\mathbf{x}) \in \mathbb{R}^{d \times d} \quad (12.11)$$

for some scalars $\alpha_{ij}(\mathbf{x}) \in \mathbb{R}$. If \mathbf{x} is on a linear subspace, then

$$C^i(\mathbf{x}) \equiv 0. \quad (12.12)$$

Proof. If \mathbf{x} is on the quadratic surface, $\mathbf{x}^T A_j \mathbf{x} = 0, j = 1, \dots, r$. Its directional derivative along any tangent vector \mathbf{t} at \mathbf{x} is also zero: $\mathbf{t}^T A_j \mathbf{x} = 0, j = 1, \dots, r$. When we contract $H_{p_i}(\mathbf{x})$ with the tangent vectors, all the cross terms vanish, except for the ones in (12.11). The linear case is easy since in the expression (12.8) \mathbf{b}_j are by definition the normal vectors to the subspace and their inner product with any tangent vector is zero. \square

⁶Notice that, the tangent vectors to the surface at the point are readily available as the orthogonal complement of the normal vectors.

Corollary 12.5 (Contractions on a Quadratic Hyper-Surface). *If there is a $(D-1)$ -dimensional quadratic surface in the arrangement and it is defined by a symmetric matrix A , then, for a point \mathbf{x} on it, we have*

$$C^i(\mathbf{x}) = 2\alpha_i(\mathbf{x})T(\mathbf{x})^T A T(\mathbf{x}) \in \mathbb{R}^{(D-1) \times (D-1)} \quad (12.13)$$

for some scalar $\alpha_i(\mathbf{x}) \in \mathbb{R}$.

We summarize the properties of the derivatives, Hessians, and contractions of points on an arrangement of subspaces and surfaces in Table 12.1.

Location of \mathbf{x}	$\nabla_p(\mathbf{x})$	$C(\mathbf{x})$
$S \cap S'$	0	N/A
S^l	$\text{rank}(\nabla_p(\mathbf{x})) = D - d^l$	0
S^q	$\text{rank}(\nabla_p(\mathbf{x})) = D - d^q$	$\sum 2\alpha_j(\mathbf{x})T^T A_j T$

Table 12.1. Properties of the derivatives and contractions for points at different locations in the arrangement.

12.1.3 Generalized Principal Surface Analysis (GPSA)

As we see from the analysis above, the derivatives and contractions at each data point depend on its location in the arrangement. Therefore, we can potentially use the derivatives and contractions to determine the membership of the point. We have used the derivatives for exactly the same purpose in the case of GPCA. We now examine how the contractions may help with the case of GPSA.

Linear Subspaces versus Quadratic Surfaces

According to Proposition 12.4, the contractions of points in the linear subspaces are always zero. Therefore, in principle, we can clearly separate points that belong to the linear subspaces from points that belong to the quadratic surfaces. That is, we can easily perform the following segmentation:

$$\mathbf{X} = \mathbf{X}^l \cup \mathbf{X}^q, \quad \text{s.t. } \mathbf{X}^l \subseteq \cup S^l, \quad \mathbf{X}^q \subseteq \cup S^q. \quad (12.14)$$

We can further apply GPCA to segment \mathbf{X}^l into different linear subspaces; that leaves us with only the question how to segment \mathbf{X}^q into different quadratic surfaces.

Quadratic Surfaces of Different Dimensions

If the quadratic surfaces have different dimensions, the problem is also relatively simple (at least in principle): one can segment the points into surfaces of different dimensions by examining the rank of the first order derivatives $\nabla_p(\mathbf{x})$ at each point \mathbf{x} . Therefore, the only case left is the case with an arrangement of quadratic surfaces of the same dimension.

Quadratic Surfaces of the Same Dimension

Without loss of generality, let $\mathbf{x}_1, \mathbf{x}_2 \in \mathbf{X}^q$ be two (linearly independent) points on an arrangement of d -dimensional quadratic surfaces. Let $T(\mathbf{x}_1)$ and $T(\mathbf{x}_2)$ be the two tangent spaces at \mathbf{x}_1 and \mathbf{x}_2 , respectively. Their intersection $T(\mathbf{x}_1, \mathbf{x}_2) \doteq T(\mathbf{x}_1) \cap T(\mathbf{x}_2)$ is in general a $(2d - D)$ -dimensional subspace in \mathbb{R}^D , assuming $2d > D$. Every $\mathbf{t} \in T(\mathbf{x}_1, \mathbf{x}_2)$ is a tangent to the surface at both points. We define the “mutual contractions” for $\mathbf{x}_1, \mathbf{x}_2 \in \mathbf{X}^q$ to be the contractions of Hessians at \mathbf{x}_1 and \mathbf{x}_2 with $T(\mathbf{x}_1, \mathbf{x}_2)$:

$$\begin{aligned}\bar{C}^i(\mathbf{x}_1, \mathbf{x}_2) &\doteq T(\mathbf{x}_1, \mathbf{x}_2)^T H_{p_i}(\mathbf{x}_1) T(\mathbf{x}_1, \mathbf{x}_2) \in \mathbb{R}^{(2d-D) \times (2d-D)}, \\ \bar{C}^i(\mathbf{x}_2, \mathbf{x}_1) &\doteq T(\mathbf{x}_1, \mathbf{x}_2)^T H_{p_i}(\mathbf{x}_2) T(\mathbf{x}_1, \mathbf{x}_2) \in \mathbb{R}^{(2d-D) \times (2d-D)},\end{aligned}$$

for $i = 1, \dots, s$. Notice that both $\bar{C}^i(\mathbf{x}_1, \mathbf{x}_2)$ and $\bar{C}^i(\mathbf{x}_2, \mathbf{x}_1)$ are symmetric matrices. Since the space of all $n \times n$ symmetric matrices has dimension $n(n+1)/2$, we define $M \doteq (2d - D)(2d - D + 1)/2$. Then we have the following relationships between the subspaces spanned by the two sets of mutual contractions matrices:

Theorem 12.6 (Mutual Contraction Subspace). *Suppose $M > D - d$ and $D < 2d$. If $\mathbf{x}_1, \mathbf{x}_2 \in \mathbf{X}^q$ both belong to the same quadratic surface, then we have*

$$\text{span}\{\bar{C}^1(\mathbf{x}_1, \mathbf{x}_2), \dots, \bar{C}^s(\mathbf{x}_1, \mathbf{x}_2)\} = \text{span}\{\bar{C}^1(\mathbf{x}_2, \mathbf{x}_1), \dots, \bar{C}^s(\mathbf{x}_2, \mathbf{x}_1)\}, \quad (12.15)$$

which is a proper subspace in \mathbb{R}^M . We call it the mutual contraction subspace between \mathbf{x}_1 and \mathbf{x}_2 .

Proof. Suppose the quadratic surface is defined by the set of symmetric matrices $A_j, j = 1, \dots, D - d$. Similar to the proof of Proposition 12.4, one can show that both sets of matrices span the same subspace as the following $D - d$ matrices:

$$T(\mathbf{x}_1, \mathbf{x}_2)^T A_j T(\mathbf{x}_1, \mathbf{x}_2), \quad j = 1, \dots, D - d. \quad (12.16)$$

By the assumption $M > D - d$, the subspace is proper. \square

Be aware that the mutual contraction subspaces, unlike the normal vectors for linear subspaces, are not globally invariant on the quadratic surfaces. They may change for different choices in the pair $(\mathbf{x}_1, \mathbf{x}_2)$. Nevertheless, they give very effective necessary conditions for segmenting the data points: two points belong to the same quadratic surface only if their mutual contraction subspaces are the same. We summarize our above discussions as Algorithm 12.1.

Like the GPCA algorithm, the GPSA algorithm assumes noise-free data, but it is designed in such a way that it can handle moderate noises.⁷ When the noise level is high, it might be very difficult to robustly estimate all the fitting polynomials. A simple but not necessarily the best solution is to put a heuristic threshold on

⁷For instance, the algorithm does not rely on noise-sensitive operations such as polynomial factorization.

Algorithm 12.1 (Generalized Principal Surface Analysis).

Given a large enough set of data points \mathbf{X} in \mathbb{R}^D sampled from an arrangement of linear subspaces and quadratic surfaces with the dimensions of the quadratic surfaces satisfying $2d > D$ and $M > D - d$:

- Find a set of homogeneous polynomials $p_1(\mathbf{x}), \dots, p_s(\mathbf{x})$ of the lowest degree that fit the data \mathbf{X} .
- Separate the \mathbf{X} into three groups using the criteria in Table 12.1. \mathbf{X}^{in} : points on the intersections; \mathbf{X}^l : points on linear subspaces; and \mathbf{X}^q : points on quadratic varieties.
- Segment \mathbf{X}^l into n linear subspaces via GPCA and identify the subspaces.
- Segment \mathbf{X}^q , based on the rank of $\nabla_p(\mathbf{x})$, into t groups $\mathbf{X}^{d_1}, \dots, \mathbf{X}^{d_t}$ with \mathbf{X}^{d_i} containing data points from d_i -dimensional quadratic surfaces.
- **for** $i = 1 : t$
 - Segment \mathbf{X}^{d_i} into m_i quadratic surfaces based on Theorem 12.6.
 - Identify the parameters of the m_i quadratic surfaces.
- **end**

the eigenvalues of the data matrix $\mathbf{V}_n(D)$ to determine its null space. We leave a more detailed discussion on the robustness issues to Chapter 5.

Example 12.7 We illustrate the above algorithm using a numerical example. Assume we have data points sampled from two surfaces – a paraboloid S_1 and a plane S_2 in \mathbb{R}^3 :

$$S_1: \mathbf{x}^T A \mathbf{x} = x^2 + y^2 - zw = 0, \quad A = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -0.5 \end{bmatrix}; \quad S_2: \mathbf{b}^T \mathbf{x} = y = 0, \quad \mathbf{b} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix},$$

where the homogeneous coordinates are $\mathbf{x} = [x, y, z, w]^T$ with $w = 1$. Thus a third order polynomial $p(\mathbf{x}) = x^2y + y^3 - yzw = 0$ can fit all the data points in \mathbf{X} . The gradient and Hessian at each point are:

$$\nabla_p(\mathbf{x}) = [2xy, x^2 + 3y^2 - zw, -yw, -yz]^T, \quad H_p(\mathbf{x}) = \begin{bmatrix} 2y & 2x & 0 & 0 \\ 2x & 6y & -w & -z \\ 0 & -w & 0 & -y \\ 0 & -z & -y & 0 \end{bmatrix}. \quad (12.17)$$

As shown in Table 12.2, we examine four points. Point \mathbf{x}_1 is on both surfaces, \mathbf{x}_2 is on the plane S_2 only, and $\mathbf{x}_3, \mathbf{x}_4$ are both on the quadratic surface S_1 only. The values of the gradients and contraction matrices are consistent with those in Table 12.1. Note that the first two columns of the $T(\mathbf{x})$ matrix for \mathbf{x}_3 are also the tangent vectors for S_1 at \mathbf{x}_4 . Therefore the top-left 2×2 submatrices of $C(\mathbf{x})$ for \mathbf{x}_3 and \mathbf{x}_4 are their mutual contraction matrices $\bar{C}(\mathbf{x}_3, \mathbf{x}_4)$ and $\bar{C}(\mathbf{x}_4, \mathbf{x}_3)$, respectively. The two matrices are linearly dependent, which agrees with Theorem 12.6. ■

Points	Locations	$\nabla_p(\mathbf{x})$	$H_p(\mathbf{x})$	$T(\mathbf{x})$	$C(\mathbf{x})$
$\mathbf{x}_1 = \begin{bmatrix} 1 \\ 0 \\ 1 \\ 1 \end{bmatrix}$	$S_1 \cap S_2$	$\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$	$\begin{bmatrix} 0 & 2 & 0 & 0 \\ 2 & 0 & -1 & -1 \\ 0 & -1 & 0 & 0 \\ 0 & -1 & 0 & 0 \end{bmatrix}$	N/A	N/A
$\mathbf{x}_2 = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \end{bmatrix}$	$S_2 - S_1$	$\begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$	$\begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 2 & -1 & 0 \\ 0 & -1 & 0 & -1 \\ 0 & 0 & -1 & 0 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$
$\mathbf{x}_3 = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 1 \end{bmatrix}$	$S_1 - S_2$	$\begin{bmatrix} 0 \\ 2 \\ -1 \\ -1 \end{bmatrix}$	$\begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 6 & -1 & -1 \\ 0 & -1 & 0 & -1 \\ 0 & -1 & -1 & 0 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0.5 \\ 0 & 1 & 0 \\ 0 & -1 & 1 \end{bmatrix}$	$\begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & -1 \\ 0 & -1 & 0.5 \end{bmatrix}$
$\mathbf{x}_4 = \begin{bmatrix} 0 \\ -1 \\ 1 \\ 1 \end{bmatrix}$	$S_1 - S_2$	$\begin{bmatrix} 0 \\ 2 \\ 1 \\ 1 \end{bmatrix}$	$\begin{bmatrix} -2 & 0 & 0 & 0 \\ 0 & -6 & -1 & -1 \\ 0 & -1 & 0 & 1 \\ 0 & -1 & 1 & 0 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0.5 \\ 0 & 1 & 0 \\ 0 & -1 & -1 \end{bmatrix}$	$\begin{bmatrix} -2 & 0 & 0 \\ 0 & -2 & -1 \\ 0 & -1 & -0.5 \end{bmatrix}$

Table 12.2. Result of evaluating the derivatives, Hessians, tangents, and contractions of the fitting polynomial at four points.

12.1.4 Variations to the Basic GPSA Algorithm

We here address how the basic GPSA algorithm should be improved or modified in case some of the conditions for the algorithm vary.

Segmentation Polynomials

In the special case when all the quadratic surfaces are hyper-surfaces (which are often the only quadratic surfaces considered in practice), each surface is defined by a single symmetric matrix A and there is only one fitting polynomial p for \mathbf{X}^q . According to Theorem 12.6, we have:

Corollary 12.8 (Mutual Contractions on a Quadratic Hyper-Surface). *If all the quadratic surfaces are of $D - 1$ dimension, for two points \mathbf{x}_1 and \mathbf{x}_2 to be on the same quadratic surface S^q , we must have*

$$\bar{C}(\mathbf{x}_1, \mathbf{x}_2) \sim \bar{C}(\mathbf{x}_2, \mathbf{x}_1) \in \mathbb{R}^{(D-2) \times (D-2)}, \quad (12.18)$$

where \sim means “equal up to a nonzero scalar.”

Notice that the notion of “segmentation polynomials” that we have introduced in Section ?? also applies to the case with quadratic surfaces. This can be very useful in handling noises in the data: Instead of identifying the entire set of polynomials that fit the data, we can use only one polynomial and segment the data correctly to different hypersubspaces or hypersurfaces, based on the criteria for the hypersurface case $d = D - 1$. Therefore, Corollary 12.8 in fact applies to quadratic surfaces of any dimension. The only case in which it does not offer effective constraints on the mutual contraction is when $D = 3$ since $D - 2 = 1$. We study this special case below.

Special Case: $D = 3, d = 2$

According to Corollary 12.8, the GPSA algorithm does not work for only one special case: (homogeneous) 2-dimensional quadratic surfaces in \mathbb{R}^3 . Because $D - 2 = 1$, the mutual contractions $\bar{C}(\mathbf{x}_1, \mathbf{x}_2)$ become scalars. Therefore some additional algebraic signatures are needed in order to separate points on quadratic

surfaces in \mathbb{R}^3 . Given a point \mathbf{x} on a quadratic surface defined by the symmetric matrix $A \in \mathbb{R}^{3 \times 3}$, we want to solve the 6 unknown entries of A . From Proposition 12.2 and Corollary 12.5, we have the equations:

$$\begin{aligned} 2\alpha(\mathbf{x})\mathbf{n}(\mathbf{x})^T A\mathbf{x} &= \mathbf{n}(\mathbf{x})^T \nabla_p(\mathbf{x}) \in \mathbb{R}, \\ 2\alpha(\mathbf{x})T(\mathbf{x})^T AT(\mathbf{x}) &= C(\mathbf{x}) \in \mathbb{R}^{2 \times 2}, \end{aligned}$$

where $\mathbf{x}, \mathbf{n}(\mathbf{x}), \nabla_p(\mathbf{x}), T(\mathbf{x})$ and $C(\mathbf{x})$ are all known. The second equation is symmetric and it only gives three scalar equations. Let $A^s \in \mathbb{R}^6$ be the vector of the stacked unknown entries of A . Since the scalar $\alpha(\mathbf{x}) \in \mathbb{R}$ is also unknown, we can rewrite the above two equations in A^s as the following relationship

$$M(\mathbf{x})A^s \sim V(\mathbf{x}), \quad M(\mathbf{x}) \in \mathbb{R}^{4 \times 6}, V(\mathbf{x}) \in \mathbb{R}^4, \quad (12.19)$$

where $M(\mathbf{x}), V(\mathbf{x})$ depend only on $\mathbf{x}, \mathbf{n}(\mathbf{x}), \nabla_p(\mathbf{x}), T(\mathbf{x})$ and $C(\mathbf{x})$. Let $V(\mathbf{x})^\perp \in \mathbb{R}^{4 \times 3}$ be the orthogonal complementary of $V(\mathbf{x}) \in \mathbb{R}^4$, i.e., $(V(\mathbf{x})^\perp)^T V(\mathbf{x}) = 0$. Eliminating the unknown scale, we end up with three scalar equations given by:

$$F(\mathbf{x})A^s \doteq (V(\mathbf{x})^\perp)^T M(\mathbf{x})A^s = 0 \in \mathbb{R}^3. \quad (12.20)$$

This leads to the following proposition.

Proposition 12.9 (Hyper-Surfaces in \mathbb{R}^3). *Given two points \mathbf{x}_1 and \mathbf{x}_2 from the same 2-dimensional (homogeneous) quadratic surface in \mathbb{R}^3 , let $F(\mathbf{x}_1), F(\mathbf{x}_2) \in \mathbb{R}^{3 \times 6}$ be the two matrices defined in (12.20) for \mathbf{x}_1 and \mathbf{x}_2 , respectively. Then*

$$\det \begin{bmatrix} F(\mathbf{x}_1) \\ F(\mathbf{x}_2) \end{bmatrix} = 0, \quad \text{and} \quad \begin{bmatrix} F(\mathbf{x}_1) \\ F(\mathbf{x}_2) \end{bmatrix} A^s = 0. \quad (12.21)$$

Proposition 12.9 gives an additional equation, other than the relation (12.15), that allows us to determine whether two points belong to the same quadratic surface in \mathbb{R}^3 . In addition, we can solve the matrix A using the above equation if two such points on the surface are given.

Degenerate Quadratic Surfaces

The above GPSA algorithm does not consider degenerate quadratic surfaces that lie in a proper subspace of \mathbb{R}^D . For example, a circle (1-dimensional) in the 3-dimensional space lies on a plane. For such degenerate quadratic surfaces, a simple solution is to first identify their supporting plane as linear subspaces, say by GPCA. Once such subspaces are identified, one can recursively apply GPSA to the data points that belong to each subspace and identify further the quadratic structures of those points.

Resolving Ambiguity in Mutual Contractions

Given a data point \mathbf{x}_1 in a quadratic surface S^q , we denote the set of all the data points \mathbf{x}_2 that have the same mutual contraction with \mathbf{x}_1 as $\mathbf{X}^{\mathbf{x}_1} \subseteq \mathbf{X}^q$. Then all the data points on S^q will be included in $\mathbf{X}^{\mathbf{x}_1}$. However, since Theorem 12.6 is only a necessary condition for two points to be on the same quadratic surface, it

is possible that in some ambiguous configurations $\mathbf{X}^{\mathbf{x}_1}$ also contains data points from other quadratic surfaces.⁸ Fortunately, such *ambiguity* can be easily detected and resolved. Given a point $\mathbf{x}_2 \in \mathbf{X}^{\mathbf{x}_1}$, we can calculate $\mathbf{X}^{\mathbf{x}_2}$. It can be conceived that $\mathbf{X}^{\mathbf{x}_1}$ and $\mathbf{X}^{\mathbf{x}_2}$ should be similar only if both \mathbf{x}_1 and \mathbf{x}_2 are on the same S^q . Therefore we can verify if \mathbf{x}_2 is on the same surface with \mathbf{x}_1 by checking if the following ratio:

$$w^{12} \doteq \frac{|\mathbf{X}^{\mathbf{x}_1} \cap \mathbf{X}^{\mathbf{x}_2}|}{|\mathbf{X}^{\mathbf{x}_1} \cup \mathbf{X}^{\mathbf{x}_2}|} \quad (12.22)$$

is higher than some threshold. In this way, we can single out all the points in the same surface as \mathbf{x}_1 . The remaining data points can be similarly segmented.

12.1.5 Experiments on Synthetic Data

Segmentation with Ambiguity

Figure 12.1 shows the results for segmenting 1000 data points drawn from a paraboloid and a sphere. Most of the data points are correctly segmented. The misclassified ones are those close to the intersection of the two surfaces. Given two points $\mathbf{x}_1 = [1, 0, 0, 1]^T$ on the sphere and $\mathbf{x}_2 = [0, 0, 0, 1]^T$ on the paraboloid, the contraction matrices with respect to the common tangent vectors at \mathbf{x}_1 and \mathbf{x}_2 are $C(\mathbf{x}_1) \sim C(\mathbf{x}_2) \sim [\begin{smallmatrix} 1 & 0 \\ 0 & 1 \end{smallmatrix}]$, that satisfies Theorem 12.6. This is the ambiguity we have discussed in Section 12.1.4, and it has been successfully resolved in the experiment using the criterion (12.22). The two surfaces are correctly retrieved from the segmented data points.

Segmentation with Noises

Figure 12.2 shows the results for segmenting 1000 data points drawn from two ellipsoids contaminated by Gaussian noise with a standard deviation $\sigma = 0.2$. Although the algorithm is developed assuming the noise-free case, in practice it can tolerate a moderate amount of noises in the data. It can provide a decent initialization for other iterative optimization schemes such as EM. In this experiment, we perform a simple postprocessing by reassigning the data points based on the recovered surfaces.

⁸Suppose there exists an \hat{A} such that \hat{A}^s , the vector obtained by stacking \hat{A} , is in the null space of $T^T \otimes T^T$. If $A_1 = A_2 + \hat{A}$ then $T^T A_1 T = T^T A_2 T + T^T \hat{A} T = T^T A_2 T$. Thus even though the two data samples are from two different surfaces, they will have the same contraction by T (up to a scalar)! However in the noiseless case the set of \hat{A} is a zero-measure set, and even with noise the probability remains small.

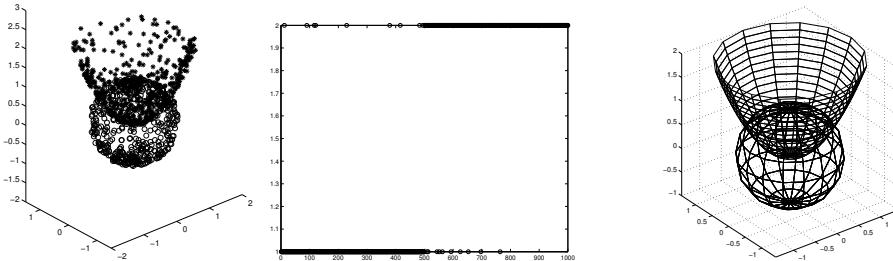


Figure 12.1. The paraboloid $x^2 + y^2 - z = 0$ and the sphere $x^2 + y^2 + z^2 = 1$. Left: The 1000 data points with the first 500 drawn from the paraboloid. Middle: The segmentation results. The x -axis is the indices of the points and the y -axis is the group number. Right: The recovered quadratic surfaces.

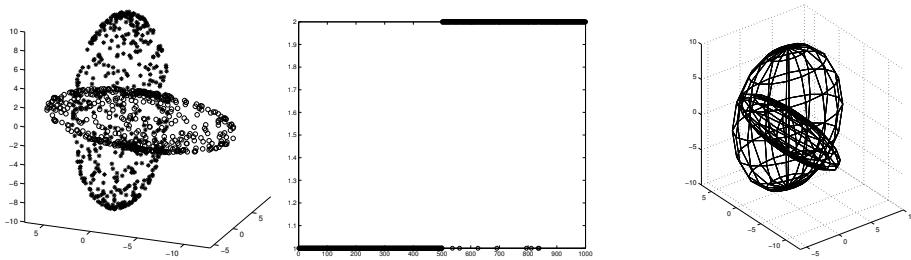


Figure 12.2. Left: The 1000 data points drawn from two ellipsoids in \mathbb{R}^3 . Middle: The segmentation results. Right: The recovered quadratic surfaces.

12.2 Other Nonlinear Extensions

Yi notes: More general manifolds... But I don't know what you guys have in mind.

12.3 Bibliographic Notes

Many different methods have been developed to model data points sampled from nonlinear manifolds. In [Scholkopf et al., 1998], a set of nonlinear mappings called *kernels* are introduced. A kernel maps the original data points into another space in which the manifold is linear. PCA is then applied to the mapped data points in the new space. ISOMAP [Tenenbaum et al., 2000] and LLE [Roweis and L.Saul, 2000] utilize the local information of the data set to infer the global properties of the manifold. To fit data points into multiple models, [Tipping and Bishop, 1999a] developed probabilistic PCA to model data points with multiple Gaussian distributions using EM. [?] presented a scalable subspace

clustering algorithm by performing clustering for all subspaces of dimensions. K-subspaces [Ho et al., 2003] is an extension of the K-means for fitting multiple linear models. [Leonardis et al., 2002] developed a subspace selection method for multiple subspaces. The non-iterative linear GPCA algorithm is first proposed in [Vidal et al., 2003b]. A recursive linear GPCA algorithm has been developed in [Huang et al., 2004].

Part IV

Appendices

+ This is page 244
Printer: Opaque this

Appendix A

Basic Facts from Algebraic Geometry

“Algebra is but written geometry; geometry is but drawn algebra.”
– Sophie Germain

As a centuries-old practice in science and engineering, people often fit polynomials to a given set of data points. In this book, we often use the zero-set of (multivariate) polynomials to model a given data set. In mathematics, polynomials and their zero sets are studied under the topic of Algebraic Geometry, with Hilbert’s Nullstellensatz establishing the basic link between Algebra (polynomials) and Geometry (the zero set of polynomials, a geometric object). In order to make this book more self-contained, in this appendix we briefly review some of the basic notions and facts that are frequently used in this book. For a more systematic introduction to this topic, the reader may refer to the classic texts of Lang [Lang, 1993] and Esenbud [?].

A.1 Polynomial Ring

Consider the D -dimensional vector space over a field R (of characteristic 0), denoted as R^D , where R normally is the field of real numbers \mathbb{R} or the field of complex numbers \mathbb{C} .

Let $R[\mathbf{x}] = R[x_1, x_2, \dots, x_D]$ be the set of all polynomials of D variables x_1, x_2, \dots, x_D . Then $R[\mathbf{x}]$ has the algebraic properties of a *commutative ring* with the two operations “summation” and “multiplication” of polynomials. The elements of R are called *scalars* which represent the constant polynomials of

degree 0. A *monomial* is a product of the variables; its degree is the number of the variables (counting repeats). A monomial of degree n takes the general form $\mathbf{x}^{\mathbf{n}} = x_1^{n_1} x_2^{n_2} \cdots x_D^{n_D}$ with $0 \leq n_j \leq n$ and $n_1 + n_2 + \cdots + n_D = n$. There are a total of

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

different degree- n monomials.

Definition A.1 (Veronese Map). *For given n and D , the Veronese map of degree n , denoted as $\nu_n : R^D \rightarrow R^{M_n(D)}$, is defined as:*

$$\nu_n : [x_1, \dots, x_D]^T \mapsto [\dots, \mathbf{x}^{\mathbf{n}}, \dots]^T, \quad (\text{A.1})$$

where $\mathbf{x}^{\mathbf{n}}$ are degree- n monomials of the form $x_1^{n_1} x_2^{n_2} \cdots x_D^{n_D}$ with $\mathbf{n} = (n_1, n_2, \dots, n_D)$ chosen in the degree-lexicographic order.

Example A.2 (The Veronese Map of Degree 2 in 3 Variables). If $\mathbf{x} = [x_1, x_2, x_3]^T \in R^3$, the Veronese map of degree 2 is given by:

$$\nu_2(\mathbf{x}) = [x_1^2, x_1 x_2, x_1 x_3, x_2^2, x_2 x_3, x_3^2]^T \in R^6.$$

■

In the context of Kernel methods (Chapter 2), the Veronese map is usually referred to as the polynomial embedding and the ambient space $R^{M_n(D)}$ is called the *feature space*.

A *term* is a scalar multiplies a monomial and every polynomial can be written as a finite sum of nonzero terms. A polynomial $p(\mathbf{x})$ is said to be *homogeneous* if the monomials in all its terms have the same degree. Sometimes, people also use the word *form* to mean a homogeneous polynomial. Thus, every homogeneous polynomial $p(\mathbf{x})$ of degree n can then be written as a superposition of the monomials:

$$p(\mathbf{x}) = \mathbf{c}_n^T \nu_n(\mathbf{x}) = \sum c_{n_1, \dots, n_D} x_1^{n_1} \cdots x_D^{n_D}, \quad (\text{A.2})$$

where $c_{n_1, \dots, n_D} \in R$ are the coefficients associated with the monomials $\mathbf{x}^{\mathbf{n}} = x_1^{n_1} \cdots x_D^{n_D}$.

In this book, we are primarily interested in the *algebra* of homogeneous polynomials with D variables.¹ Because of that, we view R^D as a projective space – the set of one-dimensional subspaces (meaning lines through the origin). Any such a one-dimensional subspace, say a line L , can be represented by a point $(a_1, a_2, \dots, a_n) \neq (0, 0, \dots, 0)$ on the line. The result is a projective $(D-1)$ -space over R which can be regarded as the D -tuples (a_1, a_2, \dots, a_n) of elements of R , modulo the equivalence relation $(a_1, a_2, \dots, a_n) \sim (ba_1, ba_2, \dots, ba_n)$ for all $b \neq 0$ in R .

¹For algebra of polynomials defined on R^D as an affine space, the reader may refer to [Lang, 1993].

If $p(x_1, x_2, \dots, x_D)$ is a homogeneous polynomial of degree n , then for $b \in R$ we have

$$p(ba_1, ba_2, \dots, ba_n) = b^n p(a_1, a_2, \dots, a_n). \quad (\text{A.3})$$

Therefore, whether $p(a_1, a_2, \dots, a_n) = 0$ or not on a line L does not depend on the representative point chosen on the line L .

We may view $R[\mathbf{x}]$ as a *graded ring* which has a direct sum

$$R[\mathbf{x}] = \bigoplus_{i=0}^{\infty} R_i = R_0 \oplus R_1 \oplus \dots \oplus R_n \oplus \dots. \quad (\text{A.4})$$

In particular, $R_0 = R$ is the set of nonzero scalars (that represent constants). It is convention to define the degree of zero 0 to be infinite. R_1 is the set of all homogeneous polynomials of degree one, i.e., the set of 1-forms,

$$R_1 \doteq \{b_1 x_1 + b_2 x_2 + \dots + b_D x_D : [b_1, b_2, \dots, b_D]^T \in R^D\}. \quad (\text{A.5})$$

Obviously, the dimension of R_1 as a vector space is also D . R_1 can also be viewed as the dual space $(R^D)^*$ of R^D . For convenience, we also define the following two sets

$$\begin{aligned} R_{(m)} &\doteq \bigoplus_{i=0}^m R_i = R_0 \oplus R_1 \oplus \dots \oplus R_m, \\ R^{(m)} &\doteq \bigoplus_{i=m}^{\infty} R_i = R_m \oplus R_{m+1} \oplus \dots, \end{aligned}$$

which are the set of polynomials of degree up to degree m and those of degree higher and equal to m .

A.2 Ideals and Algebraic Sets

Definition A.3 (Ideal). *An ideal in the (commutative) polynomial ring $R[\mathbf{x}]$ is an additive subgroup I (with respect to the summation of polynomials) such that if $p(\mathbf{x}) \in I$ and $q(\mathbf{x}) \in R[\mathbf{x}]$, then $p(\mathbf{x})q(\mathbf{x}) \in I$.*

From the definition, it is easy to verify that if I, J are two ideals of $R[\mathbf{x}]$, their intersection $K = I \cap J$ is also an ideal. However, the product

$$IJ \doteq \{f(\mathbf{x})g(\mathbf{x}), \forall f(\mathbf{x}) \in I, g(\mathbf{x}) \in J\}$$

is not necessarily an ideal. The previously defined set $R^{(m)}$ is an ideal for every m . In particular, $R^{(1)}$ is the so-called *irrelevant ideal*, sometimes written as R_+ .

An ideal is said to be *generated* by a subset $\mathcal{G} \subset I$ if every element $p(\mathbf{x}) \in I$ can be written in the form

$$p(\mathbf{x}) = \sum_{i=1}^k q_i(\mathbf{x})g_i(\mathbf{x}), \quad \text{with } q_i(\mathbf{x}) \in R[\mathbf{x}] \text{ and } g_i(\mathbf{x}) \in \mathcal{G}. \quad (\text{A.6})$$

We write (\mathcal{G}) for the ideal generated by a subset $\mathcal{G} \subset R[\mathbf{x}]$; if \mathcal{G} contains only finite number of elements $\{g_1, \dots, g_k\}$, we usually write (g_1, \dots, g_k) in place of (\mathcal{G}) . An ideal I is *principal* if it can be generated by one element (i.e., $I = p(\mathbf{x})R[\mathbf{x}]$ for some polynomial $p(\mathbf{x})$).

An ideal I of the (commutative) polynomial ring $R[\mathbf{x}]$ is *prime* if $I \not\supseteq R[\mathbf{x}]$ and if $p(\mathbf{x}), q(\mathbf{x}) \in R[\mathbf{x}]$ and $p(\mathbf{x})q(\mathbf{x}) \in I$ implies that $p(\mathbf{x}) \in I$ or $q(\mathbf{x}) \in I$. Equivalently, if I is prime and for any ideals J, K with $JK \subseteq I$ we have $J \subseteq I$ or $K \subseteq I$.

A polynomial $p(\mathbf{x})$ is said to be *prime* or irreducible if $p(\mathbf{x})$ generates a prime ideal. Equivalently, if $p(\mathbf{x})$ is irreducible if $p(\mathbf{x})$ is not a nonzero scalar and whenever $p(\mathbf{x}) = f(\mathbf{x})g(\mathbf{x})$, then one of $f(\mathbf{x})$ and $g(\mathbf{x})$ is a nonzero scalar.

Definition A.4 (Homogeneous Ideal). *A homogeneous ideal of $R[\mathbf{x}]$ is an ideal that is generated by homogeneous polynomials.*

Note that the sum of two homogeneous polynomials of different degrees is no longer a homogeneous polynomial. Thus, a homogeneous ideal contains nonhomogeneous polynomials too.

Definition A.5 (Algebraic Set). *Given a set of homogeneous polynomials $J \subset R[\mathbf{x}]$, we may define a corresponding (projective) algebraic set $Z(J)$ as a subset of R^D to be*

$$Z(J) \doteq \{(a_1, a_2, \dots, a_n) \in R^D \mid f(a_1, a_2, \dots, a_n) = 0, \forall f \in J\}. \quad (\text{A.7})$$

If we view algebraic sets as the closed sets of R^D , this assigns a topology to the space R^D , which is called the *Zariski topology*.²

If $X = Z(J)$ is an algebraic set, an algebraic subset $Y \subset X$ is a set of the form $Y = Z(K)$ (where K is a set of homogeneous polynomials) that happens to be contained in X . A nonempty algebraic set is said to be *irreducible* if it is not the union of two nonempty smaller algebraic subsets. We call irreducible algebraic sets as *algebraic varieties*. For instance, any subspace of R^D is an irreducible algebraic variety because its vanishing ideal is generated by linear forms.

There is an inverse construction of algebraic sets. Given any subset $X \subseteq R^D$, we define the *ideal of X* to be the set of all polynomials that vanish on X :

$$I(X) \doteq \{f(\mathbf{x}) \in R[\mathbf{x}] \mid f(a_1, a_2, \dots, a_n) = 0, \forall (a_1, a_2, \dots, a_n) \in X\}. \quad (\text{A.8})$$

One can easily verify that $I(X)$ is an ideal. If we identify two polynomial functions if they agree at all the points of X , we get the *coordinate ring* $A(X)$ of X as the quotient $R[\mathbf{x}]/I(X)$.

Now, consider a set of homogeneous polynomials $J \subset R[\mathbf{x}]$ (which is not necessarily an ideal) and a subset $X \subset R^D$ (which is not necessarily an algebraic set).

Proposition A.6. *The following facts are true:*

²This is because the intersection of any algebraic sets is an algebraic set; and the union of finitely many algebraic sets is also an algebraic set.

1. $I(Z(J))$ is the smallest ideal that contains J ;
2. $Z(I(X))$ is the smallest algebraic set that contains X .

Proposition A.7. *If X is an algebraic set and $I = Z(X)$ is the ideal of X , then X is irreducible if and only if I is a prime ideal.*

Proof. If X is irreducible and $f(\mathbf{x})g(\mathbf{x}) \in I$, since $Z(\{I, f(\mathbf{x})\}) \cup Z(\{I, g(\mathbf{x})\}) = X$, then either $X = Z(\{I, f(\mathbf{x})\})$ or $X = Z(\{I, g(\mathbf{x})\})$. That is, either $f(\mathbf{x})$ or $g(\mathbf{x})$ vanishes on X and is in I . Conversely, suppose $X = X_1 \cup X_2$. If both X_1 and X_2 are algebraic sets strictly smaller than X , then there exist polynomials $f_1(\mathbf{x})$ and $f_2(\mathbf{x})$ that vanish on X_1 and X_2 respectively, but not on X . Since the product $f_1(\mathbf{x})f_2(\mathbf{x})$ vanishes on X , we have $f_1(\mathbf{x})f_2(\mathbf{x}) \in I$ but neither $f_1(\mathbf{x})$ or $f_2(\mathbf{x})$ is in I . So I is not prime. \square

A.3 Algebra and Geometry: Hilbert's Nullstellensatz

In this book, we often use an algebraic set to model a given set of data points and the (ideal of) polynomials that vanish on the set provide a natural parametric model for the data. One question that is of particular importance in this context is: Is there an one-to-one correspondence between ideals and algebraic sets? This is in general not true as the ideals $I = (f^2(\mathbf{x}))$ and $J = (f(\mathbf{x}))$ both vanish on the same algebraic set as the zero-set of the polynomial $f(\mathbf{x})$. Fortunately, this turns out to be essentially the only case that prevents the one-to-one correspondence between ideals and algebraic sets.

Definition A.8 (Radical Ideal). *Given a (homogeneous) ideal I of $R[\mathbf{x}]$, the (homogeneous) radical ideal of I is defined to be*

$$\text{rad}(I) \doteq \{f(\mathbf{x}) \in R[\mathbf{x}] \mid f(\mathbf{x})^m \in I \text{ for some integer } m\}. \quad (\text{A.9})$$

Please let it to the reader to verify that $\text{rad}(I)$ is indeed an ideal and furthermore, if I is homogeneous, so is $\text{rad}(I)$.

Hilbert proved in 1893 the following important theorem that establishes one of the fundamental results in algebraic geometry:

Theorem A.9 (Nullstellensatz). *Let R be an algebraically closed field (e.g., $R = \mathbb{C}$). If $I \subset R[\mathbf{x}]$ is an (homogeneous) ideal, then*

$$I(Z(I)) = \text{rad}(I). \quad (\text{A.10})$$

Thus, the correspondences $I \mapsto Z(I)$ and $X \mapsto I(X)$ induce a one-to-one correspondence between the collection of (projective) algebraic sets of R^D and (homogeneous) radical ideals of $R[\mathbf{x}]$.

One may find up to five different proofs for this theorem in [?].³ The importance of Nullstellensatz cannot be exaggerated. It is a natural extension of Gauss' fundamental theorem of algebra⁴ to polynomials with multiple variables. One of the remarkable consequences of Nullstellensatz is that it identifies a geometric object (algebraic sets) with an algebraic object (radical ideals).

In our context, we often assume our data points are drawn from an algebraic set and use the set of vanishing polynomials as a parametric model for the data. Hilbert's Nullstellensatz guarantees such a model for the data is well-defined and unique. To some extent, when we fit vanishing polynomials to the data, we are essentially inferring the underlying algebraic set. In the next section, we will discuss how to extend Hilbert's Nullstellensatz to the practical situation in which we only have finitely many sample points from an algebraic set.

A.4 Algebraic Sampling Theory

In this book, we often face a common mathematical problem: We need to identify a (projective) algebraic set $Z \subseteq R^D$ from a finite, though maybe very large, number of sample points in Z . In general, the variety Z is not necessarily irreducible⁵ and the ideal $I(Z)$ is not necessarily prime.

From an algebraic viewpoint, it is impossible to recover a continuous variety Z from a finite number of discrete sample points, regardless how many. To see this, the set of all polynomials that vanish on one (projective) point z is a submaximal ideal⁶ \mathfrak{m} in the (homogeneous) polynomial ring $R[z]$. The set of polynomials that vanish on a set of sample points $\{z_1, z_2, \dots, z_i\} \subseteq Z$ is the intersection

$$\mathfrak{a}_i \doteq \mathfrak{m}_1 \cap \mathfrak{m}_2 \cap \dots \cap \mathfrak{m}_i, \quad (\text{A.11})$$

which is a radical ideal that is typically much larger than $I(Z)$.

Thus, some additional assumptions must be imposed on the algebraic set in order to make the problem of inferring $I(Z)$ from the samples well-defined. Typically, we assume that the ideal $I(Z)$ of the algebraic set Z in question is generated by a set of (homogeneous) polynomials whose degrees are bounded by a relatively small n . That is,

$$\begin{aligned} I(Z) &\doteq (f_1, f_2, \dots, f_s) \quad \text{s.t. } \forall j \deg(f_j) \leq n, \\ Z(I) &\doteq \{z \in R^D \mid f_i(z) = 0, i = 1, 2, \dots, s\}. \end{aligned}$$

³Strictly speaking, for homogeneous ideals, for the one-to-one correspondence to be exact, one should only consider proper radical ideals.

⁴Every degree- n polynomial in one variable has exactly n roots in an algebraically closed field such as \mathbb{C} (counting repeats).

⁵For instance, it is often the case that Z is the union of many subspaces or algebraic surfaces.

⁶The ideal of a point in the affine space is a maximal ideal; and the ideal of a point in the projective space is conventionally called a submaximal ideal. They both are “maximal” in the sense that they cannot be a subideal of any other homogeneous ideal of the polynomial ring.

We are interested in how to retrieve $I(Z)$ uniquely from a set of sample points $\{\mathbf{z}_1, \mathbf{z}_2, \dots, \mathbf{z}_i\} \subseteq Z$. In general, $I(Z)$ is always a proper subideal of \mathfrak{a}_i , regardless how large i is. However, the information about $I(Z)$ can still be retrieved from \mathfrak{a}_i in the following sense.

Theorem A.10 (Sampling of a Low-Degree Algebraic Set). *Consider a (continuous) set $Z \subseteq R^D$ whose ideal $I(Z)$ is generated by polynomials in $R_{(n)}$. For almost all⁷ sequences $\{\mathbf{z}_1, \mathbf{z}_2, \dots\} \subseteq Z^\infty$, there exists a large enough N such that $I(Z) = (\mathfrak{a}_N \cap R_{(n)})$. In particular, if $I(Z) \cap R_{(n)} = \emptyset$, so will be $\mathfrak{a}_N \cap R_{(n)} = \emptyset$ for a large enough N .*

Proof. Let $\mathfrak{a}_0 = R[\mathbf{x}]$ be the graded ring of all homogeneous polynomials:

$$\mathfrak{a}_0 \doteq R[\mathbf{x}] = R_0 \oplus R_1 \oplus \cdots \oplus R_n \oplus \cdots.$$

Define $A_0 = (\mathfrak{a}_0 \cap R_{(n)})$, the ideal generated by polynomials in $\mathfrak{a}_0 = R[\mathbf{x}]$ of degree less than or equal to n . Since $1 \in R[\mathbf{x}] \cap R_{(n)}$ is the generator of this ideal, we have $A_0 = R[\mathbf{x}]$.

If $A_0 \neq \mathfrak{a}$ (i.e., $Z \neq \emptyset$), pick $N_1 = 1$ point $\mathbf{z}_0 \in Z$. Then $1(\mathbf{z}_0) \neq 0$. Let $\mathfrak{a}_{N_1} = \mathfrak{a}_1$ be the submaximal ideal that vanishes on \mathbf{z}_0 and we define $A_1 = (\mathfrak{a}_1 \cap R_{(n)})$. Notice that the degree of the generators of A_1 is 1, strictly higher than that of A_0 .

Since $R[\mathbf{x}]$ is finitely generated (i.e., is a Noetherian ring [Lang, 1993]), so is A_1 . Notice that $I(Z)$ is a subideal of A_1 . Let $\{a_1, a_2, \dots, a_r\}$ be the minimal set of generators for the quotient ideal $A_1/I(Z)$ in the coordinate ring $R[\mathbf{x}]/I(Z)$ of Z . If $A_1/I(Z) \neq 0$ (i.e., $A_1 \neq I(Z)$), define $\phi : Z^r \rightarrow R$, a map from the r -product of Z to R , by

$$\phi(\mathbf{z}_1, \dots, \mathbf{z}_r) \doteq \det \begin{bmatrix} a_1(\mathbf{z}_1) & a_2(\mathbf{z}_1) & \cdots & a_r(\mathbf{z}_1) \\ a_1(\mathbf{z}_2) & a_2(\mathbf{z}_2) & \cdots & a_r(\mathbf{z}_2) \\ \vdots & \vdots & \ddots & \vdots \\ a_1(\mathbf{z}_r) & a_2(\mathbf{z}_r) & \cdots & a_r(\mathbf{z}_r) \end{bmatrix} \in R. \quad (\text{A.12})$$

We contend that ϕ is nonzero on an open set of Z^r . For any fixed set of $r-1$ points $\mathbf{z}_2, \dots, \mathbf{z}_r \in Z$, the polynomial $\psi(\mathbf{z}) \doteq \phi(\mathbf{z}, \mathbf{z}_2, \dots, \mathbf{z}_r)$ is a superposition of the generators $a_1(\mathbf{z}), a_2(\mathbf{z}), \dots, a_r(\mathbf{z})$. If $\phi \equiv 0$ on Z^r , then $\psi = w_1 a_1 + w_2 a_2 + \cdots + w_r a_r = 0$ in $A_1/I(Z)$ for some nonzero $w_1, w_2, \dots, w_r \in R$. This contradicts the minimality of the generators.⁸ Thus, $\phi = 0$ only on a closed, hence zero-measure, set of Z^r . Now, in addition to \mathbf{z}_0 , we pick r new points $\mathbf{z}_1, \mathbf{z}_2, \dots, \mathbf{z}_r$ in general position on Z such that $\phi \neq 0$. That gives us a total of $N_2 = N_1 + r = 1 + r$ points on Z , for which we may continue to construct in a similar fashion \mathfrak{a}_{N_2} and A_2 .

Thus, inductively, let $\mathfrak{a}_{N_{i+1}}$ be the new ideal that vanishes on all the $N_{i+1} \doteq N_i + r_i$ points chosen so far and let $A_{i+1} = (\mathfrak{a}_{N_{i+1}} \cap R_{(n)})$. By the choice

⁷“Almost all” means “except for only a zero-measure set.”

⁸If w_1, w_2, \dots, w_r are all zeros, simply apply the same arguments to a subset of the generators a_1, a_2, \dots, a_r .

of the points, any linear combination of the generators a_1, \dots, a_{r_i} for $A_i/I(Z)$ cannot be in $A_{i+1}/I(Z)$ (otherwise we must have $\phi = 0$ on the chosen r_i points). Nevertheless, since $A_{i+1}/I(Z) \subset A_i/I(Z) = (a_1, \dots, a_{r_i})$, the generators of A_{i+1} must be of the form:

$$b = a_1 h_1 + a_2 h_2 + \cdots + a_{r_i} h_{r_i}, \quad (\text{A.13})$$

where $h_j \in R[x]$, $j = 1, \dots, r_i$. One of the polynomials h_j must be of $\deg(h_j) \geq 1$. Otherwise, the polynomial $\sum_{j=1}^i a_j \cdot h_j \in A_{i+1}/I(Z)$ with nonzero $h_j \in R$ vanishes on the newly chosen r_i points, which contradicts $\phi \neq 0$. That is, if $A_i/I(Z) \neq 0$ (i.e., $A_i \neq I(Z)$), the lowest degree of the generators of $A_{i+1}/I(Z)$ must be strictly higher than that of $A_i/I(Z)$.⁹

Since A_i is always generated by polynomials (of the maximal degree n) in the finite-dimensional (vector) space $R_{(n)}$, the following descending sequence

$$A_0 \supseteq A_1 \supseteq A_2 \supseteq \cdots \supseteq A_i \supseteq A_{i+1} \supseteq \cdots \quad (\text{A.14})$$

must stabilize after $i > n$ (i.e., it is an Artinian sequence [Lang, 1993]), and furthermore $A_{n+1} = A_{n+2} = I(Z)$. \square

We point out that in the above proof, no clear bound on the total number N of points needed is given because the number r_i at each inductive step depends on the particular sequence $\{z_j\}$ and the algebraic set Z and can be very hard to estimate when Z is not irreducible.¹⁰ Nevertheless, the above theorem guarantees that, even if a sequence of sample points of Z can be initially ineffective for identifying the algebraic set, the above sequence of ideals A_i will eventually stabilize to the desired ideal $I(Z)$ as long as the remainder of the sequence is sufficiently random.

Example A.11 (A Hyperplane in \mathbb{R}^3). Consider a plane $P = \{z \in \mathbb{R}^3 : f(z) = az_1 + bz_2 + cz_3 = 0\}$. Given any two points in general position in the plane P , $f(x) = ax_1 + bx_2 + cx_3$ will be the only (homogeneous) polynomial of degree 1 that fits the two points. In terms of the language introduced before, we have $I(P) = (\mathfrak{a}_2 \cap R_{(1)})$. \blacksquare

Example A.12 (Zero Polynomial). When $Z = \mathbb{R}^D$, the only polynomial vanishes on Z is the zero polynomial, i.e., $I(Z) = (0)$. Since the zero polynomial is regarded to be of infinite degree, we have $(\mathfrak{a}_N \cap R_{(n)}) = \emptyset$ for any given n and a large enough $N(n)$. \blacksquare

The above theorem can be viewed as a first step towards an algebraic analogy to the well-known Nyquist-Shannon sampling theory in signal processing.¹¹ Here a signal is replaced by an algebraic set and the frequency bandwidth is replaced by the bound on the degree of polynomials. It has been widely practiced in engineering that a curve or surface described by polynomial equations can be recovered

⁹If it happens so that $A_{i+1} = I(Z)$, the statement still holds since the generator for $A_{i+1}/I(Z)$ is 0, which has infinite degree.

¹⁰However, loose bounds can be easily obtained from the dimension of $R_{(n)}$ as a vector space. In fact, in the algorithm, we implicitly used the dimension $M_n(K)$ of R_n as a bound for N .

¹¹Which stipulates that a continuous signal with a limited frequency bandwidth Ω can be uniquely determined from a sequence of discrete samples with a sampling rate higher than 2Ω .

from a sufficient number of sample points in general configuration, a procedure often loosely referred to as “polynomial fitting.” However, the algebraic basis for this is often not clarified and the conditions for the uniqueness of the solution are usually not well characterized or specified. This problem certainly merits further investigation in the future.

A.5 Decomposition of Ideals and Algebraic Sets

Modeling a data set as an algebraic set does not stop at obtaining its vanishing ideal (and polynomials). The ultimate goal is to extract all the internal geometric or algebraic structures of the algebraic set. For instance, if an algebraic set consists of multiple subspaces – the so-called subspace arrangement, we need to know how to derive from its vanishing ideal the number of subspaces, their dimensions, and a basis of each subspace.

Thus, given an algebraic set X or equivalently its vanishing ideal $I(X)$, we like to decompose or segment it into a union of subsets each of which can no longer be further decomposed. As we have introduced earlier, an algebraic set that cannot be decomposed into smaller algebraic sets is called irreducible. As one of the fundamental finiteness theorem of algebraic geometry, we have:

Theorem A.13. *An algebraic set can have only finitely many irreducible components. That is, for some n ,*

$$X = X_1 \cup X_2 \cup \cdots \cup X_n, \quad (\text{A.15})$$

where X_1, X_2, \dots, X_n are irreducible algebraic varieties.

Proof. The proof is essentially based on the fact that the polynomial ring $R[\mathbf{x}]$ is Noetherian (i.e., finitely generated), and there are only finitely many prime ideals containing $I(X)$ that are minimal with respect to inclusion (See [?]). \square

The vanishing ideal $I(X_i)$ of each irreducible algebraic variety X_i must be a prime ideal that is minimal over the radical ideal $I(X)$ – there is no prime subideal of $I(X_i)$ that includes $I(X)$. The ideal $I(X)$ is precisely the intersection of all the minimal prime ideals:

$$I(X) = I(X_1) \cap I(X_2) \cap \cdots \cap I(X_n). \quad (\text{A.16})$$

This intersection is called a *minimal primary decomposition* of the radical ideal $I(X)$. Thus the primary decomposition of a radical ideal is closely related to the notion of “segmenting” or “decomposing” an algebraic set into multiple irreducible algebraic varieties: If we know how to decompose the ideal, we can easily find the irreducible algebraic variety corresponding to each primary component. To a large extent, Generalized Principal Component Analysis introduced in Chapter 4 studies how to decompose the vanishing ideal of a subspace arrangement and hence segment the subspace arrangement into individual subspaces (which are irreducible).

A.6 Hilbert Function, Polynomial, and Series

Finally, we introduce an important invariants of algebraic sets, given by the so-called Hilbert function. Knowing the values of Hilbert function can be very useful in the identification of subspace arrangements, especially the number of subspaces and their dimensions.

Given a (projective) algebraic set Z and its vanishing ideal $I(Z)$, We can grade the ideal by polynomial degree as

$$I(Z) = I_0(Z) \oplus I_1(Z) \oplus \cdots \oplus I_i(Z) \oplus \cdots. \quad (\text{A.17})$$

The *Hilbert function* of Z is defined to be

$$h_I(i) \doteq \dim(I_i(Z)). \quad (\text{A.18})$$

Notice that $h_I(i)$ is exactly the number of linearly independent polynomials of degree i that vanish on Z . In this book, we also refer to h_I as the Hilbert function of the algebraic set Z .¹²

The *Hilbert series*, also known as the Poincaré series, of the ideal I is defined to be the power series¹³

$$\mathcal{H}(I, t) \doteq \sum_{i=0}^{\infty} h_I(i)t^i = h_I(0) + h_I(1)t + h_I(2)t^2 + \cdots. \quad (\text{A.19})$$

Thus, given $\mathcal{H}(I, t)$, we know all the values of the Hilbert function h_I from its coefficients.

Example A.14 (Hilbert Series of the Polynomial Ring). The Hilbert series of the polynomial ring $R[\mathbf{x}] = \mathbb{R}[x_1, x_2, \dots, x_D]$ is

$$\mathcal{H}(R[\mathbf{x}], t) = \sum_{i=0}^{\infty} \dim(R_i)\tfrac{t^i}{i!} = \sum_{i=0}^{\infty} \binom{D+i-1}{i} t^i = \frac{1}{(1-t)^D}. \quad (\text{A.20})$$

One can easily verify the correctness of the formula with the special case $D = 1$. Obviously, the coefficients of the Hilbert series of any ideal (as a subset of $R[\mathbf{x}]$) are bounded by those of $\mathcal{H}(R[\mathbf{x}], t)$ and hence the Hilbert series converges. ■

Example A.15 (Hilbert Series of a Subspace). The above formula can be easily generalized to the vanishing ideal of a subspace S of dimension d in \mathbb{R}^D . Let the co-dimension of the subspace be $c = D - d$. We have

$$\mathcal{H}(I(S), t) = \left(\frac{1}{(1-t)^c} - 1 \right) \cdot \left(\frac{1}{(1-t)^{D-c}} \right) = \frac{1 - (1-t)^c}{(1-t)^D}. \quad (\text{A.21})$$

■

¹²In the literature, however, the Hilbert function of an algebraic set Z is sometimes defined to be the dimension of the homogeneous components of the coordinate ring $A(Z) \doteq R[\mathbf{x}]/I(Z)$ of Z , which is the codimension of $I(Z)$ as a subspace in R .

¹³In general, the Hilbert series can be defined for any finitely-generated graded module $E = \bigoplus_{i=1}^{\infty} E_i$ using any Euler-Poincaré \mathbb{Z} -valued function $h_E(\cdot)$ as $\mathcal{H}(E, t) \doteq \sum_{i=0}^{\infty} h_E(i)t^i$ [Lang, 1993]. We here for $E = I$ choose $h_I(i) = \dim(I_i)$.

The following theorem, also due to Hilbert, reveals that the values of the Hilbert function of an ideal have some remarkable properties:

Theorem A.16 (Hilbert). *Let $I(Z)$ be the vanishing ideal of an algebraic set Z over $R[x_1, \dots, x_D]$, then the values of its Hilbert function $h_I(i)$ agree, for large i , with those of a polynomial of degree $\leq D$. This polynomial, denoted as $H_I(i)$, is called the Hilbert polynomial of $I(Z)$.*

Then in the above example, for the polynomial ring, the Hilbert function itself is obviously a polynomial in i

$$H_R(i) = h_R(i) = \binom{D+i-1}{i} = \frac{1}{(D-1)!}(D+i-1)(D+i-2)\cdots(i+1).$$

However, for a general ideal I (of an algebraic set), it is not necessarily true that all values of its Hilbert function h_I agree with those of its Hilbert polynomial H_I . They might agree only when i is large enough. Thus, for a given algebraic set (or ideal), it would be interesting to know how large i needs to be in order for the Hilbert function to coincide with a polynomial. As we will see in Appendix B, for subspace arrangements, there is a very elegant answer to this question. One can even derive closed-form formulae for the Hilbert polynomials. These results are very important and useful for Generalized Principal Component Analysis, both conceptually and computationally.

Appendix B

Algebraic Properties of Subspace Arrangements

In this book, the main problem that we study is how to segment a collection of data points drawn from a subspace arrangement $\mathcal{A} = \{S_1, S_2, \dots, S_n\}$, formally introduced in Chapter ??.¹ $Z_{\mathcal{A}} = S_1 \cup S_2 \cup \dots \cup S_n$ is the union of all the subspaces. $Z_{\mathcal{A}}$ can be naturally described as the zero set of a set of polynomials, which makes it an *algebraic set*. The solution to the above problem typically relies on inferring the subspace arrangement $Z_{\mathcal{A}}$ from the data points. Thus, knowing the algebraic properties of $Z_{\mathcal{A}}$ may significantly facilitate this task.

Although subspace arrangements seem to be a very simple class of algebraic sets, a full characterization of their algebraic properties is a surprisingly difficult, if not impossible, task. Subspace arrangements have been a centuries-old subject that still actively interweaves many mathematical fields: algebraic geometry and topology, combinatorics and complexity theory, graph and lattice theory, etc. Although the results are extremely rich and deep, in fact only a few special classes of subspace arrangements have been well characterized.

In this appendix, we examine some important concepts and properties of subspace arrangements that are closely related to the subspace-segmentation problem. The purpose of this appendix is two-fold: 1. to provide a rigorous justification for the GPCA algorithms derived in the book, especially Chapter 4; 2. to introduce important properties of subspace arrangements, which may suggest potential improvements of the algorithms. For readers who are interested only in

¹Unless stated otherwise, the subspace arrangement considered will always be a central arrangement, as in Definition 3.7.

the basic GPCA algorithms and their applications, this appendix can be skipped at first read.

B.1 Ideals of Subspace Arrangements

Vanishing Ideal of a Subspace.

A d -dimensional subspace S can be defined by $k = D - d$ linearly independent linear forms $\{l_1, \dots, l_k\}$:

$$S \doteq \{\mathbf{x} \in R^D : l_i(\mathbf{x}) = 0, i = 1, \dots, k = D - d\}, \quad (\text{B.1})$$

where l_i is of the form $l_i(\mathbf{x}) = a_{i1}x_1 + a_{i2}x_2 + \dots + a_{iD}x_D$ with $a_{ij} \in R$. Let S^* denote the space of all linear forms that vanish on S , then $\dim(S^*) \doteq k = D - d$. The subspace S is also called the zero set of S^* , i.e., points in the ambient space that vanish on all polynomials in S^* , which is denoted as $Z(S^*)$. We define

$$I(S) \doteq \{p \in R[\mathbf{x}] : p(\mathbf{x}) = 0, \forall \mathbf{x} \in S\}. \quad (\text{B.2})$$

Clearly, $I(S)$ is an ideal generated by linear forms in S^* , and it contains polynomials of all degrees that vanish on the subspace S . Every polynomial $p(\mathbf{x})$ in $I(S)$ can be written as a superposition:

$$p = l_1 h_1 + l_2 h_2 + \dots + l_k h_k \quad (\text{B.3})$$

for some polynomials $h_1, \dots, h_k \in R[\mathbf{x}]$. Furthermore, $I(S)$ is a prime ideal.²

Vanishing Ideal of a Subspace Arrangement.

Given a subspace arrangement $Z_{\mathcal{A}} = S_1 \cup S_2 \cup \dots \cup S_n$, its vanishing ideal is

$$I(Z_{\mathcal{A}}) = I(S_1) \cap \dots \cap I(S_n). \quad (\text{B.4})$$

The ideal $I(Z_{\mathcal{A}})$ can be graded by the degree of the polynomial

$$I(Z_{\mathcal{A}}) = I_m(Z_{\mathcal{A}}) \oplus I_{m+1}(Z_{\mathcal{A}}) \oplus \dots \oplus I_i(Z_{\mathcal{A}}) \oplus \dots. \quad (\text{B.5})$$

Each $I_i(Z_{\mathcal{A}})$ is a vector space that consists of forms of degree i in $I(Z_{\mathcal{A}})$, and $m \geq 1$ is the least degree of the polynomials in $I(Z_{\mathcal{A}})$. Notice that forms that vanish on $Z_{\mathcal{A}}$ may have degrees strictly less than n . One example is an arrangement of two lines and one plane in \mathbb{R}^3 . Since any two lines lie on a plane, the arrangement can be embedded into a hyperplane arrangement of two planes, and there exist forms of second degree that vanish on the union of the three subspaces. The dimension of $I_i(Z_{\mathcal{A}})$ is known as the Hilbert function $h_I(i)$ of $Z_{\mathcal{A}}$.

Example B.1 (Boolean Arrangement). The Boolean arrangement is the collection of coordinate hyperplanes $H_j \doteq \{\mathbf{x} : x_j = 0\}, 1 \leq j \leq D$. The vanishing ideal of the Boolean arrangement is generated by a single polynomial $p(\mathbf{x}) = x_1 x_2 \cdots x_D$ of degree D . ■

²It is a prime ideal because for any product $p_1 p_2 \in I(S)$, either $p_1 \in I(S)$ or $p_2 \in I(S)$.

Example B.2 (Braid Arrangement). The Braid arrangement is the collection of hyperplanes $H_{jk} \doteq \{x : x_j - x_k = 0\}$, $1 \leq j \neq k \leq D$. Similarly, the vanishing ideal the Braid arrangement is generated by a single polynomial $p(x) = \prod_{1 \leq j < k \leq D} (x_j - x_k)$. ■

Theorem B.3 (Regularity of Subspace Arrangements). *The vanishing ideal $I(Z_A)$ of a subspace arrangement $Z_A = S_1 \cup S_2 \cup \dots \cup S_n$ is n -regular. This implies that $I(Z)$ has a set of generators with degree $\leq n$.*

Proof. For the concept of n -regularity and the proof of the above statement, please refer to [?]. □

Due to the above theorem, the subspace arrangement Z_A is uniquely determined as the zero set of all polynomials of degree up to n in its vanishing ideal, i.e., as the zero set of polynomials in

$$Z_A = Z(I_{(n)}),$$

where $I_{(n)} \doteq I_0 \oplus \dots \oplus I_n$.

Product Ideal of a Subspace Arrangement

Let $J(Z_A)$ be the ideal generated by the products of linear forms

$$\{l_1 \cdot l_2 \cdots l_n, \quad \forall l_j \in S_j^*, j = 1, \dots, n\}.$$

Or equivalently, we can define $J(Z_A)$ to be the product of the n ideals $I(S_1), I(S_2), \dots, I(S_n)$:

$$J(Z_A) \doteq I(S_1) \cdot I(S_2) \cdots I(S_n).$$

Then, the *product ideal* $J(Z_A)$ is a subideal of $I(Z_A)$. Nevertheless, the two ideals share the same zero set:

$$Z_A = Z(J) = Z(I). \tag{B.6}$$

By definition I is the largest ideal that vanishes on Z_A . I is in fact the *radical ideal* of the product ideal J , i.e., $I = \text{rad}(J)$. We may also grade the ideal $J(Z_A)$ by the degree

$$J(Z_A) = J_n(Z_A) \oplus J_{n+1}(Z_A) \oplus \dots \oplus I_i(Z_A) \oplus \dots \tag{B.7}$$

Notice that, unlike I , the lowest degree of polynomials in J always starts from n , the number of subspaces. The Hilbert function of J is denoted as $h_J(i) = \dim(J_i(Z_A))$. As we will soon see, the Hilbert functions (or polynomials, or series) of the product ideal J and the vanishing ideal I have very interesting and important relationships.

B.2 Subspace Embedding and PL-Generated Ideals

Let Z_A be a central subspace arrangement $Z_A = S_1 \cup S_2 \cup \dots \cup S_n$. Let $Z_{A'} = S'_1 \cup S'_2 \cup \dots \cup S'_{n'}$ be another (central) subspace arrangement. If we have $Z_A \subseteq$

$Z_{\mathcal{A}'}$, then it is necessary that for all $S_j \subset Z_{\mathcal{A}}$ there exists $S'_{j'} \subset Z_{\mathcal{A}'}$ such that $S_j \subseteq S'_{j'}$. If so, we call

$$Z_{\mathcal{A}} \subseteq Z_{\mathcal{A}'}$$

a *subspace embedding*. Beware that it is possible $n' < n$ for a subspace embedding as more than one subspace S_j of $Z_{\mathcal{A}}$ may belong to the same subspace $S'_{j'}$ of $Z_{\mathcal{A}'}$. The subspace arrangements in Theorem 4.8 are examples of subspace embedding. If $Z_{\mathcal{A}'}$ happens to be a hyperplane arrangement, we call the embedding a *hyperplane embedding*.

Is the zero-set of each homogeneous component of $I(Z_{\mathcal{A}})$, in particular $I_m(Z_{\mathcal{A}})$, a subspace embedding of $Z_{\mathcal{A}}$? Unfortunately, this is not true as counter examples can be easily constructed.

Example B.4 (Five Lines in \mathbb{R}^3). Consider five points in \mathbb{P}^2 (or equivalently, five lines in \mathbb{R}^3) The Veronese embedding of order two of a point $\mathbf{x} = [x_1, x_2, x_3] \in \mathbb{R}^3$ is $[x_1^2, x_1x_2, x_1x_3, x_2^2, x_2x_3, x_3^2] \in \mathbb{R}^6$. For five points in general position, the matrix $\mathbf{V}_2 = [\nu_2(\mathbf{x}_1), \dots, \nu_2(\mathbf{x}_5)]$ is of rank 5. Let \mathbf{c}^T be the only vector in the left null space of \mathbf{V}_2 : $\mathbf{c}^T \mathbf{V}_2 = 0$. Then $p(\mathbf{x}) = \mathbf{c}^T \nu_2(\mathbf{x})$ is in general an irreducible quadratic polynomial. Thus, the zero-set of $I_2(Z_{\mathcal{A}}) = p(\mathbf{x})$ is not a subspace arrangement but an (irreducible) cone in \mathbb{R}^3 . ■

Nevertheless, the following statement allows us to retrieve a subspace embedding from any polynomials in the vanishing ideal $I(Z_{\mathcal{A}})$.

Theorem B.5 (Hyperplane Embedding via Differentiation). *For every polynomial p in the vanishing ideal $I(Z_{\mathcal{A}})$ of a subspace arrangement $Z_{\mathcal{A}} = S_1 \cup S_2 \cup \dots \cup S_n$ and n points $\{\mathbf{x}_j \in S_j\}_{j=1}^n$ in general position, the union of the hyperplanes $\bigcup_{j=1}^n H_j = \{\mathbf{x} : Dp(\mathbf{x}_j)^T \mathbf{x} = 0\}$ is a hyperplane embedding of the subspace arrangement.*

Proof. The proof is based on the simple fact that the derivative (gradient) $\nabla f(\mathbf{x})$ of any smooth function $f(\mathbf{x})$ is orthogonal to (the tangent space of) its level set $f(\mathbf{x}) = c$. □

In the above statement, if we replace p with a collection of polynomials in the vanishing ideal, their derivatives give a subspace embedding in a similar fashion as the hyperplane embedding. When the collection contains all the generators of the vanishing ideal, the subspace embedding becomes tight – the resulting subspace arrangement coincides with the original one. This property has been used in the development of GPCA algorithms in Chapter 4.

Another concept that is closely related to subspace embedding is a *pl-generated ideal*.

Definition B.6 (pl-Generated Ideals). *An ideal is said to be pl-generated if it is generated by products of linear forms.*

If the ideal of a subspace arrangement $Z_{\mathcal{A}}$ is pl-generated, then the zero-set of every generator gives a hyperplane embedding of $Z_{\mathcal{A}}$.

Example B.7 (Hyperplane Arrangements). If $Z_{\mathcal{A}}$ is a hyperplane arrangement, $I(Z_{\mathcal{A}})$ is always pl-generated as it is generated by a single polynomial of the form:³

$$p(\mathbf{x}) = (\mathbf{b}_1^T \mathbf{x})(\mathbf{b}_2^T \mathbf{x}) \cdots (\mathbf{b}_n^T \mathbf{x}), \quad (\text{B.8})$$

where $\mathbf{b}_i \in R^D$ are the normal vectors to the hyperplanes. ■

Obviously, the vanishing ideal $I(S)$ of a single subspace S is always pl-generated. The following example shows that this is also true for an arrangement of two subspaces.

Example B.8 (Two Subspaces [?]). Let us show that for an arrangement $Z_{\mathcal{A}}$ of two subspaces, $I(Z_{\mathcal{A}})$ is always pl-generated. Let $Z_{\mathcal{A}} = S_1 \cup S_2$ and define $U^* \doteq S_1^* \cap S_2^*$ and $V^* \doteq S_1^* \setminus U^*$, $W^* \doteq S_2^* \setminus U^*$. Let (u_1, \dots, u_k) be a basis for U^* , (v_1, \dots, v_l) a basis for V^* , and (w_1, \dots, w_m) a basis for W^* . Then obviously $I(Z_{\mathcal{A}}) = I(S_1) \cap I(S_2)$ is generated by $(u_1, \dots, u_k, v_1 w_1, v_1 w_2, \dots, v_l w_m)$. ■

Now consider an arrangement of n subspaces: $Z_{\mathcal{A}} = S_1 \cup S_2 \cup \dots \cup S_n$. By its definition, the product ideal $J(Z_{\mathcal{A}})$ is always pl-generated. Now, is the vanishing ideal $I(Z_{\mathcal{A}})$ always pl-generated? Unfortunately, this is not true. Below are some counterexamples.

Example B.9 (Lines in \mathbb{R}^3 [?]). For a central arrangement $Z_{\mathcal{A}}$ of r lines in general position in \mathbb{R}^3 , $I(Z_{\mathcal{A}})$ is not pl-generated when $r = 5$ or $r > 6$. Example B.4 gives a proof for the case with $r = 5$. ■

Example B.10 (Planes in \mathbb{R}^4 [?]). For a central arrangement $Z_{\mathcal{A}}$ of r planes in general position in \mathbb{R}^4 , $I(Z_{\mathcal{A}})$ is not pl-generated for all $r > 2$. ■

However, can each homogeneous component $I_i(Z_{\mathcal{A}})$ be “pl-generated” when i is large enough? For instance, can it be that $I_n = J_n = S_1^* \cdot S_2^* \cdots S_n^*$? This is in general not true for an arbitrary arrangement and below is a counterexample.

Example B.11 (Three Subspaces in \mathbb{R}^5 – due to R. Fossum). Consider $R[\mathbf{x}] = \mathbb{R}[x_1, \dots, x_5]$ and an arrangement $Z_{\mathcal{A}}$ of three three-dimensional subspaces in \mathbb{R}^5 whose vanishing ideals are given by, respectively:

$$I(S_1) = (x_1, x_2), \quad I(S_2) = (x_3, x_4), \quad I(S_3) = ((x_1 + x_3), (x_2 + x_4)).$$

Denote their intersection as $I = I(S_1) \cap I(S_2) \cap I(S_3)$. The intersection contains the element

$$x_1 x_4 - x_2 x_3 = (x_1 + x_3)x_4 - (x_2 + x_4)x_3 = x_1(x_2 + x_4) - x_2(x_1 + x_3).$$

Then any element $(x_1 x_4 - x_2 x_3)l(x_1, \dots, x_5)$ with l a linear form is in $I_3(Z_{\mathcal{A}})$, the homogeneous component of elements of degree three. In particular, $(x_1 x_4 - x_2 x_3)x_5$ is in $I_3(Z_{\mathcal{A}})$. However, it is easy to check that this element cannot be written in the form

$$\sum_i (a_i x_1 + b_i x_2)(c_i x_2 + d_i x_4)(e_i(x_1 + x_3) + f_i(x_2 + x_4))$$

for any $a_i, b_i, c_i, d_i, e_i, f_i \in \mathbb{R}$. Thus, $I_3(Z_{\mathcal{A}})$ is not spanned by $S_1^* \cdot S_2^* \cdot S_3^*$. ■

³In algebra, an ideal which is generated by a single generator is called a principal ideal.

However, notice that the subspaces in the above example are not in “general position” – their intersections are not of the minimum possible dimension. Could $I_n = J_n = S_1^* \cdot S_2^* \cdots S_n^*$ be instead true for n subspaces if they are in general position? The answer is yes. In fact, we can say more than that. As we will see in the next section, from the Hilbert functions of I and J , we actually have

$$I_i = J_i, \quad \forall i \geq n$$

if S_1, S_2, \dots, S_n are “transversal” (i.e., all intersections are of minimum possible dimension). In other words, J_i could differ from I_i only for $i < n$.

B.3 Hilbert Function of Subspace Arrangements

In this section, we study for subspace arrangements the Hilbert function defined in Section A.6. We first discuss a few reasons why in the context of generalized principal component analysis, it is very important to know the values of the Hilbert function for the vanishing ideal I or the product ideal J of subspace arrangements. We then examine the values of the Hilbert function for a few concrete examples. Finally, we give a complete characterization of the Hilbert function, Hilbert polynomial, and Hilbert series of a general subspace arrangement. In particular, we give a closed-form formula for the Hilbert polynomial of the vanishing ideal and product ideal of the subspace arrangement.

B.3.1 Hilbert Function and GPCA

In general, for a subspace arrangement $Z_{\mathcal{A}} = S_1 \cup S_2 \cup \cdots \cup S_n$ in general position, the values of the Hilbert function $h_I(i)$ of its vanishing ideal $I(Z_{\mathcal{A}})$ are invariant under a continuous change of the positions of the subspaces. They depend only on the dimensions of the subspaces d_1, d_2, \dots, d_n or their co-dimensions $c_i = D - d_i, i = 1, \dots, n$. Thus, Hilbert function gives a rich set of invariants of subspace arrangements. In the context of GPCA, such invariants can help to determine the type of the subspace arrangement, such as the number of subspaces and their individual dimensions from a given set of (possibly noisy) sample points.

To see this, consider a sufficiently large number of sample points in general position are drawn from the subspaces $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^N \subset Z_{\mathcal{A}}$, let the embedded data matrix (via the Veronese map of degree i) to be

$$\mathbf{V}_i \doteq [\nu_i(\mathbf{x}_1), \nu_i(\mathbf{x}_2), \dots, \nu_i(\mathbf{x}_N)]^T. \quad (\text{B.9})$$

According to the Algebraic Sampling Theorem of Appendix A, the dimension of $\text{Null}(\mathbf{V}_i)$ is exactly the number of linearly independent polynomials of degree i that vanish on $Z_{\mathcal{A}}$. That is, the following relation holds

$$\dim(\text{Null}(\mathbf{V}_i)) = h_I(i) \quad (\text{B.10})$$

or equivalently,

$$\text{rank}(\mathbf{V}_i) = \dim(R_i) - h_I(i). \quad (\text{B.11})$$

Thus, if we know the Hilbert function for different subspace arrangements in advance, we can determine from the rank of the data matrix from which subspace arrangement the sample data points are drawn. The following example illustrates the basic idea.

Example B.12 (Three Subspaces in \mathbb{R}^3). Suppose that we only know our data are drawn from an arrangement of three subspaces in \mathbb{R}^3 . There are in total four different types of such arrangements, shown in Figure B.1. The values of their corresponding Hilbert function are listed in Table B.1. Given a sufficiently large number N of sample points from one of the

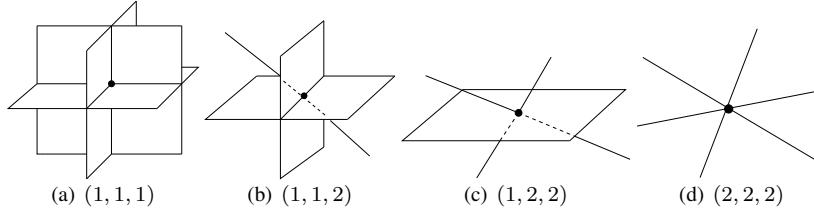


Figure B.1. Four configurations of three subspaces in \mathbb{R}^3 . The numbers are the co-dimensions (c_1, c_2, c_3) of the subspaces.

c_1	c_2	c_3	$h_{I(Z_A)}(1)$	$h_{I(Z_A)}(2)$	$h_{I(Z_A)}(3)$
1	1	1	0	0	1
1	1	2	0	0	2
1	2	2	0	1	4
2	2	2	0	3	7

Table B.1. Hilbert functions of the four arrangements (assuming the subspaces are in general position).

above subspace arrangements, the rank of the embedded data matrix $\mathbf{V}_3 \in \mathbb{R}^{N \times 10}$ can be, instead of any value between 1 and 10, only $10 - h_I(3) = 9, 8, 6, 3$, which correspond to the only four possible configurations of three subspaces in \mathbb{R}^3 : three planes, two planes and one line, one plane and two lines, or three lines, respectively, as shown in Figure B.1.

This suggests that, given the dimensions of individual subspaces, we may know the rank of the embedded data matrix. Conversely, given the rank of the embedded data matrix, we can determine to a large extent the possible dimensions of the individual subspaces. Therefore, knowing the values of the Hilbert function will help us to at least rule out in advance impossible rank values for the embedded data matrix or the impossible subspace dimensions. This is particularly useful when the data is corrupted by noise so that there is ambiguity in determining the rank of the embedded data matrix or the dimensions of the subspaces. ■

The next example illustrates how the values of Hilbert function can help determine the correct number of subspaces.

Example B.13 (Over-Fit Hyperplane Arrangements in \mathbb{R}^5). Consider a dataset sampled from a number of hyperplanes in general position in \mathbb{R}^5 . Suppose we only know that the number of the hyperplanes is at most 4, and we embed the data via the degree-4 Veronese map anyway. Table B.2 gives the possible values of the Hilbert function for an arrangement of 4, 3, 2, 1 hyperplanes in \mathbb{R}^5 , respectively. Here we use the convention that an empty set has co-dimension 5 in \mathbb{R}^5 .

c_1	c_2	c_3	c_4	$h_{I(Z_A)}(4)$	rank(V_4)
1	1	1	1	1	69
1	1	1	5	5	65
1	1	5	5	15	55
1	5	5	5	35	35

Table B.2. Values of Hilbert function of (codimension-1) hyperplane arrangements in \mathbb{R}^5 .

The first row shows that if the number of hyperplanes is exactly equal to the degree of the Veronese map, then $h_I(4) = 1$, i.e., the data matrix V_4 has a rank-1 null space. The following rows show the values of $h_I(4)$ when the number of hyperplanes is $n = 3, 2, 1$, respectively. If the rank of the matrix V_4 matches any of these values, we know exactly the number of hyperplanes in the arrangement. Figure B.2 shows a super-imposed plot of the singular values of V_4 for samples points drawn from $n = 1, 2, 3, 4$ hyperplanes in \mathbb{R}^5 , respectively.

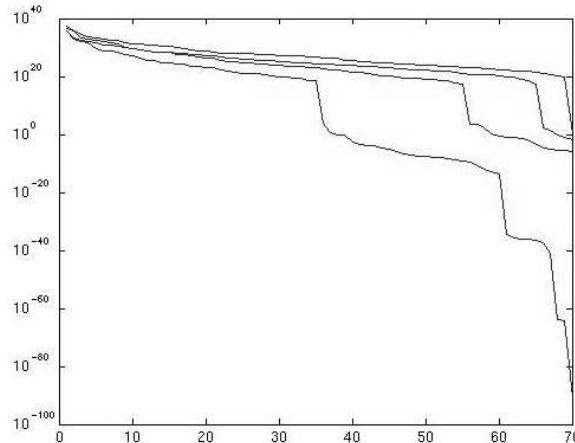


Figure B.2. A super-imposed semi-log plot of the singular values of the embedded data matrix V_4 for $n = 1, 2, 3, 4$ hyperplanes in \mathbb{R}^5 , respectively. The rank drops at 35, 55, 65, 69, which confirm the theoretical values of the Hilbert function.

Thus, in general, knowing the values of $h_I(i)$ even for $i > n$ may significantly help determine the correct number of subspaces in case the degree i of the Veronese map used for constructing the data matrix V_i is strictly higher than the number n of non-trivial subspaces in the arrangement. ■

The above examples show merely a few cases in which the values of Hilbert function may facilitate solving the GPCA problem. In Chapter [?], we will see how Hilbert function can help to significantly improve the performance of GPCA. It now remains as a question how to compute the values of Hilbert function for arbitrary subspace arrangements.

Mathematically, we are interested in finding closed-form formulae, if exist at all, for the Hilbert function (or Hilbert polynomial, or Hilbert series) of the subspace arrangements. As we will soon show, if the subspace arrangements are transversal (i.e., any intersection of subset of the subspaces has the smallest possible dimension), we are able to show that the Hilbert function (of both I and J) agrees with the Hilbert polynomial (of both I and J) with $i \geq n$; and a closed-form formula for the Hilbert polynomial is known (and will be given later). However, no general formula is known for the Hilbert function (or series) of I , especially for the values $h_I(i)$ with $i < n$. For those values, one can still compute them in advance numerically based on the identity

$$h_I(i) = \dim(\text{Null}(\mathbf{V}_i)) \quad (\text{B.12})$$

from a sufficient set of samples on the subspace arrangements. The values for each type of arrangements need to be computed only once, and the results can be stored in a table such as Table B.1 for each ambient space dimension D and number of subspaces n . We may later query these tables to retrieve information about the subspace arrangements and exploit relations among these values for different practical purposes.

However, computing the values of h_I numerically can be very expensive, especially when the dimension of the space (or the subspaces) is high. In order to densely sample the high-dimensional subspaces, the number of samples grows exponentially with the number of subspaces and their dimensions. Actually the MATLAB package that we are using runs out of the memory limit of 2GB for computing the table for the case $D = 12$ and $n = 6$.

Fortunately, for most applications in image processing, or computer vision, or systems identification, it is typically sufficient to know the values of $h_I(i)$ up to $n = 10$ and $D = 12$. For instance, for most images, the first $D = 12$ principal components already keep up to 99% of the total energy of the image, which is more than sufficient for any subsequent representation or compression purposes. Furthermore, if one chooses to use two by two blocks to represent a color image, then each block becomes one data point of dimension $2 \times 2 \times 3 = 12$. The number of segments sought for an image is typically less than ten. In system identification, the dimensions of the subspaces correspond to the orders of the systems and they are typically less than 10.

B.3.2 Special Cases of Hilbert Functions

Before we study the Hilbert function for general subspace arrangements in the next section, we here give a few special cases for which we have computed certain values of the Hilbert function.

Example B.14 (Hyperplane Arrangements). Consider $Z_{\mathcal{A}} = S_1 \cup S_2 \cup \dots \cup S_n \subset \mathbb{R}^D$ with each S_i a hyperplane. The subspaces S_i are of co-dimension 1, i.e., $c_1 = c_2 = \dots = c_n = 1$. Then we have $h_I(n) = 1$, which is consistent with the fact there is exactly one (factorable) polynomial of degree n that fits n hyperplanes. Furthermore, $h_I(i) = 0$ for all $i < n$ and

$$h_I(n+i) = \binom{D+i-1}{i}, \quad \forall i \geq 1.$$

We can generalize the case of hyperplanes to the following example. \blacksquare

Example B.15 (Subspaces Whose Duals Have No Intersection). Consider a subspace arrangement $Z_{\mathcal{A}} = S_1 \cup S_2 \cup \dots \cup S_n \subset \mathbb{R}^D$ with $S_i^* \cap S_j^* = 0$ for all $i \neq j$. In other words, if the co-dimensions of S_1, S_2, \dots, S_n are c_1, c_2, \dots, c_n , respectively, we have $c_1 + c_2 + \dots + c_n \leq D$. Notice that hyperplane arrangements are a special case here. Generalizing the result in Example A.15, one can easily show that the Hilbert series of $I(Z_{\mathcal{A}})$ (and $J(Z_{\mathcal{A}})$) is

$$\mathcal{H}(I(Z_{\mathcal{A}}), t) = \mathcal{H}(J(Z_{\mathcal{A}}), t) = f(t) \doteq \frac{\prod_{i=1}^n (1 - (1-t)^{c_i})}{(1-t)^D}. \quad (\text{B.13})$$

The values of the Hilbert function $h_I(i)$ can be easily computed from the coefficients of the function $f(t)$ associated with t^i . \blacksquare

However, if the dual subspaces S_i^* do have non-trivial intersections, the computation of Hilbert series and function becomes much more complicated. Below we give some special examples and leave the general study to the next section.

Example B.16 (Hilbert Function of Two Subspaces). We here derive a closed-form formula of $h_I(2)$ for an arrangement of $n = 2$ subspaces $Z_{\mathcal{A}} = S_1 \cup S_2$ in general position (see also Example B.8). Suppose their co-dimensions are c_1 and c_2 , respectively. In $R_1 \sim \mathbb{R}^D$, the intersection of their dual subspaces S_1^* and S_2^* has the dimension

$$c \doteq \max\{c_1 + c_2 - D, 0\}. \quad (\text{B.14})$$

Then we have

$$\begin{aligned} h_I(2) &= c \cdot (c+1)/2 + c \cdot (c_1 - c) + c \cdot (c_2 - c) + (c_1 - c) \cdot (c_2 - c) \\ &= c_1 \cdot c_2 - c \cdot (c-1)/2. \end{aligned} \quad (\text{B.15})$$

Unfortunately, even for an arrangement of three subspaces, there is no known formula for $h_I(3)$. \blacksquare

Example B.17 (Three Subspaces in \mathbb{R}^5). Consider an arrangement of three subspaces $Z_{\mathcal{A}} = S_1 \cup S_2 \cup S_3 \subset \mathbb{R}^5$ in general position. After a change of coordinates, we may assume $S_1^* = \text{span}\{x_1, x_2, x_3\}$, $S_2^* = \text{span}\{x_1, x_4, x_5\}$, and $S_3^* = \text{span}\{x_2, x_3, x_4, x_5\}$. The value of $h_I(3)$ in this case is equal to $\dim(S_1^* \cdot S_2^* \cdot S_3^*)$. Firstly, we compute $S_1^* \cdot S_2^*$ and obtain a basis for it:

$$S_1^* \cdot S_2^* = \text{span}\{x_1^2, x_1x_4, x_1x_5, x_2x_1, x_2x_4, x_2x_5, x_3x_1, x_3x_4, x_3x_5\}.$$

From this, it is then easy to compute the basis for $S_1^* \cdot S_2^* \cdot S_3^*$:

$$\begin{aligned} S_1^* \cdot S_2^* \cdot S_3^* &= \text{span}\{x_1^2x_2, x_1x_2x_4, x_1x_2x_5, x_1x_2^2, x_2^2x_4, x_2^2x_5, x_1x_2x_3, x_2x_3x_4, \\ &\quad x_2x_3x_5, x_1^2x_3, x_1x_3x_4, x_1x_3x_5, x_1x_3^2, x_3^2x_4, x_3^2x_5, x_1^2x_4, x_1x_4^2, \\ &\quad x_1x_4x_5, x_2x_4^2, x_2x_4x_5, x_3x_4^2, x_3x_4x_5, x_1^2x_5, x_1x_5^2, x_2x_5^2, x_3x_5^2\}. \end{aligned}$$

Thus, we have $h_I(3) = 26$. ■

Example B.18 (Five Subspaces in \mathbb{R}^3). Consider an arrangement of five subspaces S_1, \dots, S_5 in \mathbb{R}^3 of co-dimensions c_1, \dots, c_5 , respectively. We want to compute the value of $h_I(5)$, i.e., the dimension of homogeneous polynomials of degree five that vanish on the five subspaces $Z_A = S_1 \cup \dots \cup S_5$. For all the possible values of $1 \leq c_1 \leq \dots \leq c_5 < 3$, we have computed the values of D_5^3 and listed them in Table B.3. Notice that the values

c_1	c_2	c_3	c_4	c_5	$h_I(5)$
1	1	1	1	1	1
1	1	1	1	2	2
1	1	1	2	2	4
1	1	2	2	2	7
1	2	2	2	2	11
2	2	2	2	2	16

Table B.3. Values of the Hilbert function $h_I(5)$ for arrangements of five subspaces in \mathbb{R}^3 .

of $h_I(3)$ in the earlier Table B.1 is a subset of those of $h_I(5)$ Table B.3. In fact, many relationships like this one exist among the values of Hilbert function. If properly harnessed, they can significantly reduce the amount of work for computing Hilbert function. ■

Example B.19 (Five Subspaces in \mathbb{R}^4). Similar to the above example, we have computed the values of $h_I(5)$ for arrangements of five linear subspaces in \mathbb{R}^4 . The results are given in Table B.4. In fact, using the numerical method described earlier, we have computed using computer the values of $h_I(5)$ up to five subspaces in \mathbb{R}^{12} . ■

B.3.3 Computation of Hilbert Function

In this section, we give a general formula for the Hilbert polynomial of the subspace arrangement $Z_A = S_1 \cup S_2 \cup \dots \cup S_n$. However, due to the limit of space, we will not be able to give a detailed proof for all the results given here. Interested readers may refer to [?].

Let U be any subset of the set of indexes $\underline{n} \doteq \{1, 2, \dots, n\}$, we define the following ideals

$$I_U \doteq \bigcap_{u \in U} I(S_u), \quad J_U \doteq \prod_{u \in U} I(S_u). \quad (\text{B.16})$$

If U is empty, we use the convention $I_\emptyset = J_\emptyset = R$. We further define $V_U = \bigcap_{u \in U} S_u$, $d_U = \dim(V_U)$, and $c_U = D - d_U$.

Let us define polynomials $p_U(t)$ recursively as follows. First we define

$$p_\emptyset(t) = 1.$$

c_1	c_2	c_3	c_4	c_5	$h_I(5)$
1	1	1	1	1	1
1	1	1	1	2	2
1	1	1	1	3	3
1	1	1	2	2	4
1	1	1	2	3	6
1	1	1	3	3	8
1	1	2	2	2	8
1	1	2	2	3	11
1	1	2	3	3	14
1	1	3	3	3	17
1	2	2	2	2	15
1	2	2	2	3	19
1	2	2	3	3	23
1	2	3	3	3	27
1	3	3	3	3	31
2	2	2	2	2	26
2	2	2	2	3	31
2	2	2	3	3	36
2	2	3	3	3	41
2	3	3	3	3	46
3	3	3	3	3	51

Table B.4. Values of the Hilbert function $h_I(5)$ for arrangements of five subspaces in \mathbb{R}^4 .

For $U \neq \emptyset$ and $p_W(t)$ is already defined for all proper subsets W of U , then $p_U(t)$ is uniquely determined by the following equation

$$\sum_{W \subseteq U} (-t)^{|W|} p_W(t) \equiv 0 \pmod{(1-t)^{c_U}}, \quad \deg(p_U(t)) < c_U. \quad (\text{B.17})$$

Here $|W|$ is the number of indexes in the set W .

With the above definitions, the Hilbert series of the product ideal J is given by

$$\mathcal{H}(J, t) = \frac{p_{\underline{n}}(t)t^n}{(1-t)^D}. \quad (\text{B.18})$$

That is, the Hilbert series of the product ideal J depends only on the numbers $c_U, U \subseteq \underline{n}$. Thus, the values of the Hilbert function $h_J(i)$ are all combinatorial invariants – invariants that depend only on the values $\{c_U\}$ but not the particular position of the subspaces.

Definition B.20 (Transversal Subspaces). *The subspaces S_1, S_2, \dots, S_n are called transversal if $c_U = \min(D, \sum_{u \in U} c_u)$ for all $U \subseteq \underline{n}$. In other words, the intersection of any subset of the subspaces has the smallest possible dimension.*

Notice that the notion of “transversality” defined here is less strong than the typical notion of “general position.” For instance, according to the above definition, three coplanar lines (through the origin) in \mathbb{R}^3 are transversal. However, they are not “in general position.”

Theorem B.21. *Suppose that S_1, S_2, \dots, S_n are transversal, then $\mathcal{H}(I, t) - f(t)$ and $\mathcal{H}(J, t) - f(t)$ are polynomials in t , where $f(t) = \frac{\prod_{i=1}^n (1-(1-t)^{c_i})}{(1-t)^D}$.*

Thus, the difference between $\mathcal{H}(I, t)$ and $\mathcal{H}(J, t)$ is also a polynomial. As a corollary to the above theorem, we have

Corollary B.22. *If S_1, S_2, \dots, S_n are transversal, then $h_I(i) = H_I(i) = h_J(i) = H_J(i)$ for all $i \geq n$. That is, the Hilbert polynomials of both the vanishing ideal I and the product ideal J are the same, and the values of their Hilbert functions agree with the polynomial with $i \geq n$.*

One of the consequences of this corollary is that for transversal subspace arrangements, we must have $I_i = J_i$ for all $i \geq n$. This is a result that we have mentioned earlier in Section B.2.

Then, using the recursive formula (B.18) of the Hilbert series $\mathcal{H}(J, t)$, we can derive a closed-form formula for the values of the Hilbert function $h_I(i)$ with $i \geq n$:

Corollary B.23 (A Formula for Hilbert Function). *If S_1, S_2, \dots, S_n are transversal, then*

$$h_I(i) = h_J(i) = \sum_U (-1)^{|U|} \binom{D+i-1-c_U}{D-1-c_U}, \quad i \geq n, \quad (\text{B.19})$$

where the sum is over all index subsets U of \underline{n} for which $c_U < n$.

The following proposition shows a strong theoretical connection between Hilbert function and the GPCA problem.

Corollary B.24. *Given a transversal arrangements of n subspaces. Suppose that the co-dimensions c_1, c_2, \dots, c_n are unknown, but we know the values of the Hilbert function $h_I(i)$ for $i = n, n+1, \dots, n+D-1$, then we can uniquely recover c_1, c_2, \dots, c_n .*

As we have alluded to earlier, in the context of GPCA, these values of the Hilbert function can be estimated from the rank of the embedded data matrix \mathbf{V}_i for $i = n, n+1, \dots, n+D-1$.

B.4 Bibliographic Notes

Subspace arrangements constitute of a very special but important class of algebraic sets that have been studied in mathematics for centuries [?, ?, ?]. The importance as well as the difficulty of studying subspace arrangements can hardly

be exaggerated. Different aspects of their properties have been and are still being investigated and exploited in many mathematical fields, including algebraic geometry & topology, combinatorics and complexity theory, and graph and lattice theory, etc. See [?] for a general review. Although the results about subspace arrangements are extremely rich and deep, only a few special classes of subspace arrangements have been fully characterized. Nevertheless, thanks to the work of Derksen [?], the Hilbert function, Hilbert polynomial, and Hilbert series of the vanishing ideal (and the product ideal) of transversal subspace arrangements have been well understood recently. This appendix gives a brief summary of these theoretical developments. These results have provided a sound theoretical foundation for many of the methods developed in this book for GPCA.

Appendix C

Basic Facts from Mathematical Statistics

In the practice of science and engineering, data are often modeled as samples of a random variable (or vector) drawn from certain probabilistic distribution. Mathematical statistics then deals with the problem how to infer the underlying distribution from the given samples. To render the problem tractable, we typically assume that the unknown distribution belongs to certain parametric family (e.g., Gaussian), and the problem becomes how to estimate the parameters of the distribution from the samples.

In this appendix, we provide a brief review of some of the relevant concepts and results from mathematical statistics used in this book. The review is not meant to be complete but to make the book self-contained for readers who already have basic knowledge in probability theory and statistics. If one is looking for a more formal and complete introduction to mathematical statistics, we recommend the classic books by Bickel and Doksum [?] or by Wilks [?].

C.1 Estimation of Parametric Models

Let \mathbf{x} be a random variable or vector. For simplicity, we assume the distribution of \mathbf{x} has a density $p(\mathbf{x}, \theta)$, where the parameter (vector) $\theta = [\theta_1, \theta_2, \dots, \theta_d]^T \in \mathbb{R}^d$, once known, uniquely determines the density function $p(\cdot, \theta)$. Now suppose $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N\}$ are a set of samples of \mathbf{x} independently drawn according to the density $p(\mathbf{x}, \theta)$. That is, \mathbf{X} has the density

$$p(\mathbf{X}, \theta) = \prod_{i=1}^N p(\mathbf{x}_i, \theta). \quad (\text{C.1})$$

We call any real or vector-valued function of the samples \mathbf{X} a *statistic* and denote it by $T(\mathbf{X})$. The goal here is to properly choose the function $T(\cdot)$ so that it gives a “good” estimate for the true parameter θ .

Definition C.1 (Sufficient Statistic). A statistic $T(\mathbf{X})$ is said to be sufficient for θ if, and only if, the conditional distribution of \mathbf{X} given $T(\mathbf{X})$ does not depend on θ .

That is, $p(\mathbf{X}, \theta | T(\mathbf{X}))$ no longer depends on θ . Thus, the original samples \mathbf{X} do not contain any more information about θ than $T(\mathbf{X})$.

Theorem C.2 (Factorization Theorem). A statistic $T(\mathbf{X})$ is sufficient for θ if, and only if, there exists a function $g(t, \theta)$ and a function $h(\mathbf{X})$ such that

$$p(\mathbf{X}, \theta) = g(T(\mathbf{X}), \theta)h(\mathbf{X}). \quad (\text{C.2})$$

A popular measure of “goodness” of a statistic $T(\mathbf{X}) \in \mathbb{R}^d$ as an estimate of $\theta \in \mathbb{R}^d$ is the mean squared error between $T(\mathbf{X})$ and θ :

$$R(\theta, T) = E[\|T(\mathbf{X}) - \theta\|^2]. \quad (\text{C.3})$$

The choice of this measure is not just for convenience: When the sample size N is large, the distribution of many estimates converges to a normal distribution with θ as the mean. Then R is the variance of the estimates. In some literature, such a function is also referred to as the “risk function,” hence the capital letter “ R .”

We may rewrite the expression $R(\theta, T)$ as follows:

$$\begin{aligned} R(\theta, T) &= E[\|T(\mathbf{X}) - E[T(\mathbf{X})] + E[T(\mathbf{X})] - \theta\|^2] \\ &= E[\|T(\mathbf{X}) - E[T(\mathbf{X})]\|^2] + \|E[T(\mathbf{X})] - \theta\|^2 \\ &\doteq \text{Var}(T(\mathbf{X})) + \mathbf{b}^2(\theta, T), \end{aligned}$$

where $\mathbf{b}(\theta, T) = E[T(\mathbf{X})] - \theta$ is called the *bias* of the estimate $T(\mathbf{X})$, and $\text{Var}(T(\mathbf{X})) \in \mathbb{R}$ is the trace of the covariance matrix

$$\text{cov}(T(\mathbf{X})) \doteq E[T(\mathbf{X})T(\mathbf{X})^T] \in \mathbb{R}^{d \times d}.$$

We refer to $\text{Var}(T(\mathbf{X}))$ as the “variance” of $T(\mathbf{X})$. Thus, a good estimate is one that has both small bias and variance.

Unfortunately, there is no such thing as a universally optimal estimate that gives a smaller error R than any other estimates for all θ . For instance, if the true parameter is θ_0 , for the estimate $S(\mathbf{X}) = \theta_0$, it achieves the smallest possible error $R(\theta_0, S) = 0$. Thus, the universally optimal estimate, say T , would have to have $R(\theta_0, T) = 0$ too. As θ_0 can be arbitrary, then T has to estimate every potential parameter θ perfectly, which is impossible except for trivial cases. One can view this as a manifestation of the so-called *No Free Lunch Theorem* known in learning theory: Without any prior knowledge in θ , we can only expect a statistical estimate to be better than others most of the time, but we can never expect it to be the best *all the time*. Thus, in the future, whenever we claim some estimate is “optimal,” it will be in the restricted sense that it is optimal within a special class of estimates considered (e.g., unbiased estimates).

Define the *Fisher information matrix* to be

$$I(\theta) \doteq E\left[\left(\frac{\partial}{\partial \theta} \log p(\mathbf{X}, \theta)\right)\left(\frac{\partial}{\partial \theta} \log p(\mathbf{X}, \theta)\right)^T\right] \in \mathbb{R}^{d \times d}. \quad (\text{C.4})$$

Let $\psi(\theta) \doteq E[T(\mathbf{X})] = [\psi_1(\theta), \psi_2(\theta), \dots, \psi_d(\theta)]^T$ and define:

$$\frac{\partial \psi(\theta)}{\partial \theta} \doteq \begin{bmatrix} \frac{\partial \psi_1(\theta)}{\partial \theta_1} & \frac{\partial \psi_1(\theta)}{\partial \theta_2} & \dots & \frac{\partial \psi_1(\theta)}{\partial \theta_d} \\ \frac{\partial \psi_2(\theta)}{\partial \theta_1} & \frac{\partial \psi_2(\theta)}{\partial \theta_2} & \dots & \frac{\partial \psi_2(\theta)}{\partial \theta_d} \\ \vdots & \vdots & \dots & \vdots \\ \frac{\partial \psi_d(\theta)}{\partial \theta_1} & \frac{\partial \psi_d(\theta)}{\partial \theta_2} & \dots & \frac{\partial \psi_d(\theta)}{\partial \theta_d} \end{bmatrix} \in \mathbb{R}^{d \times d}. \quad (\text{C.5})$$

Theorem C.3 (Information Inequality). *Under reasonable conditions, we have that for all θ , $\psi(\theta)$ is differentiable and*

$$\text{cov}(T(\mathbf{X})) \geq \frac{\partial \psi(\theta)}{\partial \theta} I(\theta)^{-1} \left(\frac{\partial \psi(\theta)}{\partial \theta}\right)^T, \quad (\text{C.6})$$

where the inequality is between semi-positive definite symmetric matrices.

For unbiased estimate $\psi(\theta) = \theta$, we have $\psi'(\theta) = I$. The information inequality can be thought of as giving a lower bound for the variance of any unbiased estimate: $\text{cov}(T(\mathbf{X})) \geq I(\theta)^{-1}$, which is often referred to as the *Cramér-Rao lower bound*.

As $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N\}$ are i.i.d. samples from the distribution $p(\mathbf{x}, \theta)$, we define $I_1(\theta) \doteq E\left[\frac{\partial}{\partial \theta} \log p(\mathbf{x}_1, \theta)\left(\frac{\partial}{\partial \theta} \log p(\mathbf{x}_1, \theta)\right)^T\right] \in \mathbb{R}^{d \times d}$. Then, we have

$$I(\theta) = NI_1(\theta). \quad (\text{C.7})$$

The Cramér-Rao lower bound can be rewritten as $\text{cov}(T(\mathbf{X})) \geq \frac{1}{N} I_1(\theta)^{-1}$.

C.1.1 Uniformly Minimum Variance Unbiased Estimates

One way to resolve the above difficulty is to restrict the class of estimates. For instance, we may require the estimate $T(\mathbf{X})$ needs to be unbiased, i.e., $b(\theta, T) = 0$. Then the problem of finding the best unbiased estimate becomes

$$\min_{T(\cdot)} R(\theta, T) = \text{Var}(T(\mathbf{X})) \quad \text{s.t.} \quad E[T(\mathbf{X})] = \theta. \quad (\text{C.8})$$

The optimal T^* is then called the *uniformly minimum variance unbiased* (UMVU) estimate. Such a T^* often exists and in the absence of knowledge in θ , it seems to be the best estimate one can hope to obtain.

Definition C.4 (Complete Statistic). *A statistic T is said to be complete if the only real function $g(\cdot)$ which satisfies $E[g(T)] = 0$ for all θ is the function $g(T) \equiv 0$.*

Starting with a sufficient and complete statistic $T(\mathbf{X})$, the following theorem simplifies the computation of the UMVU estimate:

Theorem C.5 (Lehmann-Scheffé). *If $T(\mathbf{X})$ is a complete sufficient statistic and $S(\mathbf{X})$ is any unbiased estimate of θ , then $T^*(\mathbf{X}) = E[S(\mathbf{X})|T(\mathbf{X})]$ is an UMVU*

estimate of θ . If further $\text{Var}(T^*(\mathbf{X})) < \infty$ for all θ , $T^*(\mathbf{X})$ is the unique UMVU estimate.

Even so, the UMVU estimate is often too difficult to compute in practice. Furthermore, the property of unbiasedness is not invariant under functional transformation: if $T(\mathbf{X})$ is an unbiased estimate for θ , $g(T(\mathbf{X}))$ is in general not an unbiased estimate for $g(\theta)$. To have the functional invariant property, we often resort to the so-called Maximum Likelihood estimate.

C.1.2 Maximum Likelihood Estimates

As we assume the N samples \mathbf{x}_i in the given sample set \mathbf{X} are independently drawn from the same distribution, the joint distribution of \mathbf{x}_i has the density $p(\mathbf{X}, \theta) = \prod_{i=1}^N p(\mathbf{x}_i, \theta)$. Now consider $p(\mathbf{X}, \theta)$ as a function of θ with \mathbf{X} fixed. We call this function the *likelihood function*, denoted as $L(\theta, \mathbf{X}) = p(\mathbf{X}, \theta)$.

Then the *maximum likelihood (ML) estimate* of θ is given by solving the following optimization problem:

$$\hat{\theta}_N = \arg \max_{\theta} L(\theta, \mathbf{X}) = p(\mathbf{X}, \theta) = \prod_{i=1}^N p(\mathbf{x}_i, \theta). \quad (\text{C.9})$$

As $\hat{\theta}_N$ maximizes the likelihood function $L(\theta, \mathbf{X})$, a necessary condition for solving $\hat{\theta}_N$ is

$$\frac{\partial L(\theta, \mathbf{X})}{\partial \theta} \Big|_{\hat{\theta}_N} = 0. \quad (\text{C.10})$$

It is easy to show that the ML estimate is invariant under functional transformation: if $\hat{\theta}_N$ is an ML estimate of θ , then $g(\hat{\theta}_N)$ is an ML estimate of $g(\theta)$.

Since the logarithmic function is monotonic, we may choose to maximize the log likelihood function instead:

$$\hat{\theta}_N = \arg \max_{\theta} \log(L(\theta, \mathbf{X})) = \sum_{i=1}^N \log p(\mathbf{x}_i, \theta), \quad (\text{C.11})$$

which often turns out to be more convenient to use in practice. The ML estimate is a more popular choice than the UMVU estimate because its existence is easier to establish and is usually easier to compute than the UMVU estimate. Furthermore, when the sample size is large, the ML estimate is asymptotically optimal for a wide variety of parametric models. Thus, both UMVU and ML estimates give essentially the same answer in a way that we explain in more detail.

C.1.3 Estimates from a Large Number of Samples

Definition C.6 (Consistency). An estimate $\hat{\theta}_N$ of θ is said to be consistent if, and only if,

$$P[\|\hat{\theta}_N - \theta\| \geq \epsilon] \rightarrow 0 \quad (\text{C.12})$$

for all $\epsilon > 0$ as $N \rightarrow \infty$.

In other words, $\hat{\theta}_N$ is consistent if it converges in probability to θ .

Definition C.7 (Asymptotic Unbiasedness). Let $\mu_N = E[\hat{\theta}_N] \in \mathbb{R}^d$ and $\Sigma_N = \text{cov}(\hat{\theta}_N) \in \mathbb{R}^{d \times d}$. We say that $\hat{\theta}$ is asymptotically unbiased if as $N \rightarrow \infty$

$$\sqrt{N}(\mu_N - \theta) \rightarrow 0, \quad \text{and} \quad N\Sigma_N \rightarrow \Sigma > 0 \quad (\text{C.13})$$

for some positive-definite symmetric matrix $\Sigma \in \mathbb{R}^{d \times d}$.

It is easy to see that asymptotic unbiasedness is a stronger property than consistency. That is, an estimate can be consistent but asymptotically biased. In addition, for most “reasonable” estimate $\hat{\theta}_N$ (e.g., the ML estimate), due to the law of large numbers, it is often asymptotically normal distributed with mean μ_N and covariance matrix Σ_N . Therefore, the asymptotical distribution of an asymptotically unbiased estimate is uniquely characterized by the parameters θ and Σ .

Between any two asymptotically unbiased estimates, say $\hat{\theta}_N^{(1)}$ and $\hat{\theta}_N^{(2)}$, their relative *asymptotic efficiency* of $\hat{\theta}_N^{(1)}$ to $\hat{\theta}_N^{(2)}$ is defined to be the ratio

$$e(\hat{\theta}_N^{(1)}, \hat{\theta}_N^{(2)}) \doteq \frac{\det(\Sigma^{(2)})}{\det(\Sigma^{(1)})}, \quad (\text{C.14})$$

where $\Sigma^{(i)} = \lim_{N \rightarrow \infty} N\text{cov}(\hat{\theta}_N^{(i)})$, for $i = 1, 2$. The larger the efficiency ratio e , the smaller the asymptotic variance of $\hat{\theta}^{(1)}$, relative to that of $\hat{\theta}^{(2)}$. Thus, $\hat{\theta}^{(1)}$ gives more accurate or “sharper” estimate for θ although both $\hat{\theta}^{(1)}$ and $\hat{\theta}^{(2)}$ are asymptotically unbiased.

Nevertheless, according to Theorem C.3, an estimate cannot be arbitrarily more efficient than others. That is, for any asymptotically unbiased estimate $\hat{\theta}_N$, using (C.7) and (C.13), its covariance matrix is bounded asymptotically from below by the Cramér-Rao bound:

$$\lim_{N \rightarrow \infty} N\Sigma_N = \Sigma \geq I_1(\theta)^{-1}. \quad (\text{C.15})$$

Definition C.8 (Asymptotic Efficiency). An estimate $\hat{\theta}_N$ is said to be asymptotically efficient if it is asymptotically normal and it achieves equality in the Cramér-Rao bound (C.15).

Obviously, an asymptotically efficient estimate has efficiency $e \geq 1$ with respect to any other asymptotically unbiased estimates that satisfy (C.15).

Asymptotic efficiency is a desirable property for an estimate and it is sometimes referred to as asymptotic optimality. It often can be shown that UMVU estimates are asymptotically efficient. We also have that:

Proposition C.9. *In general, the maximum likelihood estimate is asymptotically efficient.*

Proof. We here outline the basic ideas for a “proof,” which can also be used to establish for other estimates their asymptotic unbiasedness or efficiency with respect to the ML estimate. Define the function

$$\psi(\mathbf{x}, \theta) \doteq \frac{\partial}{\partial \theta} \log p(\mathbf{x}, \theta) \in \mathbb{R}^d. \quad (\text{C.16})$$

Assume that the maximum likelihood estimate $\hat{\theta}_N$ exists. It satisfies the equation

$$\left. \frac{\partial L(\theta, \mathbf{X})}{\partial \theta} \right|_{\hat{\theta}_N} = \sum_{i=1}^N \psi(\mathbf{x}_i, \hat{\theta}_N) = 0. \quad (\text{C.17})$$

By the mean value theorem,

$$\sum_{i=1}^N \psi(\mathbf{x}_i, \hat{\theta}_N) - \sum_{i=1}^N \psi(\mathbf{x}_i, \theta) = \left[\sum_{i=1}^N \frac{\partial \psi(\mathbf{x}_i, \theta^*)}{\partial \theta} \right] (\hat{\theta}_N - \theta), \quad (\text{C.18})$$

where θ^* is a point between θ and $\hat{\theta}_N$. Using (C.17),

$$\sqrt{N}(\hat{\theta}_N - \theta) = \left[\frac{1}{N} \sum_{i=1}^N \frac{\partial \psi(\mathbf{x}_i, \theta^*)}{\partial \theta} \right]^{-1} \left(-N^{-\frac{1}{2}} \sum_{i=1}^N \psi(\mathbf{x}_i, \theta) \right). \quad (\text{C.19})$$

Under suitable conditions, $\hat{\theta}_N$ is consistent, and by the law of large numbers, $\frac{1}{N} \sum_{i=1}^N \frac{\partial \psi(\mathbf{x}_i, \theta^*)}{\partial \theta}$ behaves like $\frac{1}{N} \sum_{i=1}^N \frac{\partial \psi(\mathbf{x}_i, \theta)}{\partial \theta}$ which converges to

$$\begin{aligned} E\left[\frac{\partial \psi(\mathbf{x}_1, \theta)}{\partial \theta}\right] &= E\left[\frac{\partial^2}{\partial \theta^2} \log p(\mathbf{x}_1, \theta)\right] \\ &= -E\left[\frac{\partial}{\partial \theta} \log p(\mathbf{x}_1, \theta) \left(\frac{\partial}{\partial \theta} \log p(\mathbf{x}_1, \theta)\right)^T\right] = -I_1(\theta). \end{aligned}$$

It is easy to show that $\psi(\mathbf{x}, \theta)$ is zero-mean and thus, by the central limit theorem, the right-hand side of (C.19) converges to a normal distribution with zero mean and variance $I_1(\theta)^{-1}$. That is, the asymptotic variance of the ML estimate reaches the Carmér-Rao lower bound. \square

When the sample size is large and the law of large number is in effect, there is an information-theoretic justification for the ML estimate, which can be somewhat more revealing. Notice that maximizing the log likelihood function is equivalent to minimizing the following objective function:

$$\min_{\theta} H(\theta, N) \doteq \frac{1}{N} \sum_{i=1}^N -\log p(\mathbf{x}_i, \theta). \quad (\text{C.20})$$

In information theory, the quantity $-\log p(\mathbf{x}, \theta)$ is associated with the number of bits required to represent a random event \mathbf{x} that has the probability $p(\mathbf{x}, \theta)$

[Cover and Thomas, 1991]. When the sample size N is large, due to the law of large numbers, the quantity $H(\theta, N)$ converges to

$$\lim_{N \rightarrow \infty} H(\theta, N) = H(\theta) = E[-\log p(\mathbf{x}, \theta)] = \int (-\log p(\mathbf{x}, \theta)) p(\mathbf{x}, \theta_0) d\mathbf{x}, \quad (\text{C.21})$$

where $p(\mathbf{x}, \theta_0)$ is the true distribution. Notice that the above quantity is a measure similar to the notion of “entropy”: $H(\theta)$ is asymptotically the average code length of the sample set $\{\mathbf{x}_i\}$ when we assume that it is of the distribution $p(\mathbf{x}, \theta)$ while \mathbf{x} is actually drawn according to $p(\mathbf{x}, \theta_0)$. Thus, the goal of ML estimate is to find the $\hat{\theta}$ that minimizes the empirical entropy of the given sample set. This is obviously a smart thing to do as such estimate $\hat{\theta}$ gives the most compact representation of the given sample data if an optimal coding scheme is adopted [Cover and Thomas, 1991]. We refer to this as the “minimum entropy principle.”

Notice that the $\hat{\theta}$ that minimizes $\int (-\log p(\mathbf{x}, \theta)) p(\mathbf{x}, \theta_0) d\mathbf{x}$ is the same as that minimizing

$$D(p(\mathbf{x}, \theta_0) \| p(\mathbf{x}, \theta)) \doteq \int \left(\log \frac{p(\mathbf{x}, \theta_0)}{p(\mathbf{x}, \theta)} \right) p(\mathbf{x}, \theta_0) d\mathbf{x},$$

the so-called *Kullback-Leibler (KL) distance* between the two distributions $p(\mathbf{x}, \theta_0)$ and $p(\mathbf{x}, \theta)$. One may show that under general conditions, the distance is always non-negative and becomes zero if and only if $\theta = \theta_0$. Therefore, in essence, when the sample size is large, the ML objective is equivalent to minimizing the KL distance.

However, the ML estimate is known to have very bad performance in some models even with a large number of samples. This is particularly the case when the models have many redundant parameters or the distributions are degenerate. Furthermore, both UMVU and ML estimates are not the optimal estimate in a Bayesian¹ or minimax² sense. For instance, the ML estimate can be viewed as a special Bayesian estimate only when the parameter θ is of a uniform distribution.

In this book, the concepts introduced in this section help us to establish the important fact that typically, the estimate given by the algebraic GPCA algorithm is asymptotically unbiased (hence consistent). We will show that the GPCA estimate is in general not asymptotically efficient but a precise expression of its efficiency with respect to the ML estimate can be derived.

¹A bayesian estimate T^* is the solution to the following problem $\min_T \int R(\theta, T) \pi(\theta) d\theta$ for a given prior distribution $\pi(\theta)$ of θ . That is, T^* is the best estimate in terms of its average risk.

²A minimax estimate T^* is the solution to the problem $\min_T \max_\theta R(\theta, T)$. That is, T^* is the best estimate according to its worst performance. Of course, such a T^* does not have to always exist or be easier to compute than the ML estimate.

C.2 Expectation Maximization

In many practical situations, one is required to estimate a statistical model with only part of the random states being observable and the rest being “missing,” “hidden,” or “latent.” For instance, suppose that two random vectors (\mathbf{x}, \mathbf{z}) have a joint distribution (density) $p(\mathbf{x}, \mathbf{z}, \theta)$ but only samples of \mathbf{x} are observable and \mathbf{z} is not available. Our goal is, as before, to find an optimal estimate $\hat{\theta}$ for θ from the observations.

As samples of \mathbf{z} are not available, there is no way one can find the maximum likelihood estimate of θ from the *complete log likelihood function*:

$$\max_{\theta} L_c(\theta, \mathbf{X}, \mathbf{Z}) = \sum_{i=1}^N \log p(\mathbf{x}_i, \mathbf{z}_i, \theta). \quad (\text{C.22})$$

Thus, it makes sense to use only the marginal distribution of \mathbf{x} : $p(\mathbf{x}, \theta) = \int p(\mathbf{x}, \mathbf{z}, \theta) dz$ and find the maximum likelihood estimate from

$$\max_{\theta} L(\theta, \mathbf{X}) = \sum_{i=1}^N \log p(\mathbf{x}_i, \theta), \quad (\text{C.23})$$

which, in this context, is often referred to as the *incomplete log likelihood function* in the statistical literature. The problem is now reduced to a standard ML estimation problem and one can adopt any appropriate optimization method (say conjugate gradient) to find the maximum. It seems that there is no need of involving \mathbf{z} at all.

An alternative approach to maximize $L(\theta, \mathbf{X})$ is to use the available data of \mathbf{x} to estimate the values $\hat{\mathbf{z}}$ of the latent variables, and then search for the ML estimate $\hat{\theta}$ from the complete log likelihood $L_c(\theta, \mathbf{X}, \hat{\mathbf{Z}})$. There are several reasons why this often turns out to be a better idea. First, for some models $p(\mathbf{x}, \mathbf{z}, \theta)$, marginalizing \mathbf{z} out can be difficult to do or that could destroy good structures in the models. The alternative approach may better harness these structures. Second, directly maximizing $L(\theta, \mathbf{X})$ may turn out to be a very difficult optimization problem (e.g., high-dimension, many local minima), the introduction of intermediate latent variables \mathbf{z} often make the optimization easier (as we will see later). Third, in some applications, it is desired to obtain an estimate of the unobservables \mathbf{z} from the observables \mathbf{x} . The alternative approach can simultaneously estimate both θ and \mathbf{z} . Be aware that regardless of the introduction of the latent variables \mathbf{z} or not, the objective has always been to maximize the objective function $\max_{\theta} L(\theta, \mathbf{X})$.

Using the following identities

$$\forall \mathbf{z} \quad p(\mathbf{x}, \theta) = \frac{p(\mathbf{x}, \mathbf{z}, \theta)}{p(\mathbf{z}|\mathbf{x}, \theta)} \quad \text{and} \quad \int p(\mathbf{z}|\mathbf{x}, \theta) dz = 1, \quad (\text{C.24})$$

we have

$$\begin{aligned} L(\theta, \mathbf{X}) &= \sum_{i=1}^N \log p(\mathbf{x}_i, \theta) = \sum_{i=1}^N \int p(\mathbf{z}|\mathbf{x}_i, \theta) \log \frac{p(\mathbf{x}_i, \mathbf{z}, \theta)}{p(\mathbf{z}|\mathbf{x}_i, \theta)} d\mathbf{z} \\ &= \sum_{i=1}^N \int [p(\mathbf{z}|\mathbf{x}_i, \theta) \log p(\mathbf{x}_i, \mathbf{z}, \theta) - p(\mathbf{z}|\mathbf{x}_i, \theta) \log p(\mathbf{z}|\mathbf{x}_i, \theta)] d\mathbf{z}. \quad (\text{C.25}) \end{aligned}$$

Although the last expression seems more complicated than the original log likelihood $L(\theta, \mathbf{X})$, it reveals that the likelihood is a function of the *a posterior* probability $w_i(\mathbf{z}) \doteq p(\mathbf{z}|\mathbf{x}_i, \theta)$. The *a posterior* distribution of \mathbf{z} gives us the best estimate of \mathbf{z} given \mathbf{x}_i and θ . In turn, we can update the parameter θ based on the estimate of \mathbf{z} . This leads to the well-known Expectation and Maximization (EM) algorithm for optimizing the log likelihood $L(\theta, \mathbf{X})$:

Expectation Step: For fixed θ^k and every $i = 1, \dots, N$,

$$w_i^{k+1}(\mathbf{z}) = \arg \max_{w_i} [w_i(\mathbf{z}) \log p(\mathbf{x}_i, \mathbf{z}, \theta^k) - w_i(\mathbf{z}) \log w_i(\mathbf{z})].$$

Maximization Step: For fixed w_i^{k+1} ,

$$\theta^{k+1} = \arg \max_{\theta} \sum_{i=1}^N \int w_i^{k+1}(\mathbf{z}) \log p(\mathbf{x}_i, \mathbf{z}, \theta) d\mathbf{z}.$$

The Maximization step does not involve the second term in (C.25) because it is constant with w_i fixed. The Expectation step is decomposed to every i because the *a posterior* $w_i(\mathbf{z})$ depends only on \mathbf{x}_i . It is important to know that each step of the EM algorithm is in general a much simpler optimization problem than directly maximizing the log likelihood $L(\theta, \mathbf{X})$ as the sum $\sum_{i=1}^N \log p(\mathbf{x}_i, \theta)$. For many popular models (e.g., mixtures of Gaussians), one might even be able to find closed-form formulae for both steps (see Chapter 3).

Notice that the EM algorithm is an *iterative* algorithm. Like gradient ascent, it is essentially a hill-climbing algorithm that each iteration increases the value of the log likelihood.

Proposition C.10. *The Expectation Maximization process converges to one of the stationary points (extrema) of the (log) likelihood function $L(\theta, \mathbf{X})$.*

Proof. Notice that the *a posterior* w_i defined above depend on both \mathbf{z} and the parameter θ . By substituting $\mathbf{w} = \{w_i\}$ into the incomplete log-likelihood, we can view $L(\theta, \mathbf{X})$ as

$$L(\theta, \mathbf{X}) \doteq g(\mathbf{w}, \theta) \quad (\text{C.26})$$

for some function $g(\cdot)$. Instead of directly maximizing the $L(\theta, \mathbf{X})$ with respect to θ , the EM algorithm maximizes $g(\mathbf{w}, \theta)$ in a ‘hill-climbing’ style by iterating between the following two steps:

E Step: partially maximizing $g(\mathbf{w}, \theta)$ with respect to \mathbf{w} with the second variable θ fixed;

M Step: partially maximizing $g(\mathbf{w}, \theta)$ with respect to the second variable θ with \mathbf{w} fixed.

Notice that at each step the value of $g(\mathbf{w}, \theta)$ does not decrease, so does $L(\theta, \mathbf{X})$. When both steps become stationary and no longer increase the value, the process reaches a (local) extremum θ^* of the function $L(\theta, \mathbf{X})$. To see this, examine the equation

$$\frac{dL(\theta, \mathbf{X})}{d\theta} = \frac{\partial g(\mathbf{w}, \theta)}{\partial \mathbf{w}} \frac{\partial \mathbf{w}}{\partial \theta} + \frac{\partial g(\mathbf{w}, \theta)}{\partial \theta}. \quad (\text{C.27})$$

Since at θ^* , each step is stationary, we have $\frac{\partial g(\mathbf{w}, \theta)}{\partial \mathbf{w}} = 0$ and $\frac{\partial g(\mathbf{w}, \theta)}{\partial \theta} = 0$. Therefore, $\frac{dL(\theta, \mathbf{X})}{d\theta} \Big|_{\theta^*} = 0$. \square

However, for the algorithm to converge to the maximum likelihood estimate (usually the global maximum) of $L(\theta, \mathbf{X})$, a good initialization is crucial.

C.3 Mixture Models

The EM algorithm is often used for estimating a mixture model. By that, we mean the data \mathbf{x} is sampled from a distribution which is a superposition of multiple distributions:

$$p(\mathbf{x}, \theta) = \pi_1 p_1(\mathbf{x}, \theta) + \pi_2 p_2(\mathbf{x}, \theta) + \cdots + \pi_n p_n(\mathbf{x}, \theta). \quad (\text{C.28})$$

Such a distribution can be easily interpreted as the marginal distribution of a model with a latent random variable \mathbf{z} that takes discrete values in $\{1, 2, \dots, n\}$:

$$\begin{aligned} p(\mathbf{x}, \theta) &= \sum_{\mathbf{z}} p(\mathbf{x}, \mathbf{z}, \theta) = \sum_{\mathbf{z}} p(\mathbf{x}|\mathbf{z}, \theta)p(\mathbf{z}, \theta) \\ &= p(\mathbf{x}|\mathbf{z}=1, \theta)p(\mathbf{z}=1, \theta) + \cdots + p(\mathbf{x}|\mathbf{z}=n, \theta)p(\mathbf{z}=n, \theta) \end{aligned}$$

with $p(\mathbf{z}=j, \theta) = \pi_j > 0, j = 1, 2, \dots, n$. Obviously, one can use the EM algorithm to estimate the mixture model, with the mixing weights π_j as part of the unknown model parameters.

Obviously, for a mixture model, we need to estimate both the distribution parameters θ and the unknown mixing weights π_j . This increases the dimension of the optimization problem that needs to be solved. In practice, we often seek for alternative estimates of the mixture model which do not depend on the mixing weights. Such estimates may no longer be optimal but can be much easier to compute than the ML estimate.

C.3.1 Minimax Estimates

In many practical problems in which a mixture model is used, in addition to estimating the model parameter, one is also interested in the “membership” of every

sample \mathbf{x}_i , i.e., the component distribution from which \mathbf{x}_i is most likely drawn. In this case, we like to optimize the following objective:

$$\min_{\theta} \sum_{i=1}^N \left(\min_j -\log p_j(\mathbf{x}_i, \theta) \right). \quad (\text{C.29})$$

One may interpret the above objective as the following: for each sample, we find the component distribution for which \mathbf{x}_i achieves the highest likelihood; once we have decided to “assign” \mathbf{x}_i to the distribution $p_j(\mathbf{x}, \theta)$, it takes $-\log p_j(\mathbf{x}_i, \theta)$ bits to encode \mathbf{x}_i . Thus, the objective function is to minimize the total coding length given the membership of all the samples.

A straightforward way to solve the above optimization problem is to iterate between the following two steps:

Step 1: For fixed θ^k and every $i = 1, \dots, N$,

$$\sigma^{k+1}(i) = \arg \max_j \log p_j(\mathbf{x}_i, \theta). \quad (\text{C.30})$$

Step 2: With all $\sigma^{k+1}(i)$ known,

$$\theta^{k+1} = \arg \min_{\theta} \sum_{i=1}^N \left(-\log p_{\sigma^{k+1}(i)}(\mathbf{x}_i, \theta) \right). \quad (\text{C.31})$$

Notice that the two steps resemble the two steps of the EM algorithm introduced earlier. The difference is that here each sample \mathbf{x}_i is assigned to only one of the n groups while in the EM algorithm the hidden variable \mathbf{z}_i gives a probabilistic assignment of \mathbf{x}_i to the n groups. In fact, the well-known *K-means* algorithm for clustering is essentially based upon the above iteration.

C.3.2 Minimum Entropy-Product Estimates

Instead of directly searching for the maximum likelihood estimate, one can alternatively obtain an estimate of θ by minimizing the following objective function:

$$\tilde{\theta}_N = \arg \min_{\theta} \sum_{i=1}^N (-\log p_1(\mathbf{x}_i, \theta)) (-\log p_2(\mathbf{x}_i, \theta)) \cdots (-\log p_n(\mathbf{x}_i, \theta)), \quad (\text{C.32})$$

where a product of the component distributions has replaced the superposition in the mixture model $p(\mathbf{x}, \theta)$. To distinguish from the maximum likelihood estimate, we call the new estimate as the *minimum entropy-product* (MEP) estimate. One obvious advantage of the product over the conventional likelihood is that it does not require to know the mixing weights $\pi_j, j = 1, \dots, n$, which is often not available *a priori* in many applications of mixture models. In some important cases (e.g., subspace arrangements), the above objective may lead to much simpler and even closed-form solutions. The downside is that from the objective (C.32), one can no longer directly estimate the mixing weights π_j as part of the parameter θ .

In the light of the minimum entropy principle, the minimum entropy-product estimate replaces the empirical entropy with a slightly different measure of the sample set:

$$H_v(\theta, N) \doteq \frac{1}{N} \sum_{i=1}^N (-\log p_1(\mathbf{x}_i, \theta)) (-\log p_2(\mathbf{x}_i, \theta)) \cdots (-\log p_n(\mathbf{x}_i, \theta)).$$

Given a sample \mathbf{x} , if it is drawn from the first component distribution $p_1(\mathbf{x}, \theta)$, then we need a number of $-\log p_1(\mathbf{x}, \theta)$ bits to represent such a random event. Geometrically, the product $\prod_{j=1}^n (-\log p_j(\mathbf{x}, \theta))$ corresponds to a “volume” of the encoding bits in an n -dimensional space – each component distribution corresponds to one independent dimension in this space. Thus, minimizing $H_v(\theta, N)$ is equivalent to finding the estimate $\tilde{\theta}$ that gives the smallest average volume of codes needed to represent the given sample data.

There is also a technical reason why we choose the product of entropies instead of some other simple choices, say the sum of entropies: $\sum_{j=1}^n (-\log p_j(\mathbf{x}, \theta))$. From a (model) representation viewpoint, for most distributions of interest, the product gives a *genuine* representation of the parameter θ . That is, for different θ_1 and θ_2 , the products $\prod_{j=1}^n (-\log p_j(\mathbf{x}, \theta_1))$ and $\prod_{j=1}^n (-\log p_j(\mathbf{x}, \theta_2))$ correspond to different functions in \mathbf{x} . However, the sum often cannot give a genuine representation. The reader can easily verify this with $p_j(\mathbf{x}, \theta)$ being Gaussians.

Asymptotic Bias of the MEP Estimate

By the law of large numbers, we know $H_v(\theta, N)$ converges to

$$H_v(\theta) = \int \left(\prod_{j=1}^n (-\log p_j(\mathbf{x}, \theta)) \right) p(\mathbf{x}, \theta_0) d\mathbf{x}. \quad (\text{C.33})$$

Now let us define the function

$$\phi(\mathbf{x}, \theta) \doteq \frac{\partial}{\partial \theta} \prod_{j=1}^n (-\log p_j(\mathbf{x}, \theta)) = \sum_{j=1}^n -\frac{p'_j(\mathbf{x}, \theta)}{p_j(\mathbf{x}, \theta)} \prod_{k \neq j} (-\log p_k(\mathbf{x}, \theta)).$$

The derivative of $H_v(\theta)$ is

$$\frac{\partial H_v(\theta)}{\partial \theta} = E[\phi(\mathbf{x}, \theta)] = \int \phi(\mathbf{x}, \theta) p(\mathbf{x}, \theta_0) d\mathbf{x}. \quad (\text{C.34})$$

Thus, in general, the minimum entropy-product estimate $\tilde{\theta}_N$ converges to a stationary point $\tilde{\theta}$ of $H_v(\theta)$. That is, we have

$$\frac{\partial H_v(\theta)}{\partial \theta} \Big|_{\tilde{\theta}} = \int \phi(\mathbf{x}, \tilde{\theta}) p(\mathbf{x}, \theta_0) d\mathbf{x} = 0, \quad (\text{C.35})$$

where θ^* is a point between θ_0 and $\tilde{\theta}$.

In general, the true θ_0 does not necessarily satisfy the above equation. Thus $\tilde{\theta}$ is a biased estimate of θ_0 . To evaluate how large the bias is, using the mean value

theorem, we have

$$\frac{\partial H_v(\theta_0)}{\partial \theta} = \frac{\partial^2 H_v(\theta^*)}{\partial \theta^2} (\theta_0 - \tilde{\theta}). \quad (\text{C.36})$$

The bias is given by

$$\theta_0 - \tilde{\theta} = \left[\frac{\partial^2 H_v(\theta^*)}{\partial \theta^2} \right]^{-1} \frac{\partial H_v(\theta_0)}{\partial \theta}. \quad (\text{C.37})$$

Efficiency of the MEP Estimate

Following a similar derivation to that in the proof of Proposition C.9, we can obtain

$$\sqrt{N}(\tilde{\theta}_N - \theta_0) = \left[\frac{1}{N} \sum_{i=1}^N \frac{\partial \phi(\mathbf{x}_i, \theta_N^*)}{\partial \theta} \right]^{-1} \left(-N^{-\frac{1}{2}} \sum_{i=1}^N \phi(\mathbf{x}_i, \theta_0) \right), \quad (\text{C.38})$$

where θ_N^* is a point between θ_0 and $\tilde{\theta}_N$.

For many mixture models of interest (e.g., mixtures of Gaussians), the matrices $\text{cov}(\phi(\mathbf{x}, \theta))$ and $E\left[\frac{\partial \phi(\mathbf{x}, \theta)}{\partial \theta}\right]$ both exists. Then by both the law of large numbers and the central limit theorem, $\sqrt{N}(\tilde{\theta}_N - \theta_0)$ converges to a normal distribution with the covariance matrix:

$$\Sigma = E\left[\frac{\partial \phi(\mathbf{x}, \theta^*)}{\partial \theta}\right]^{-1} \text{cov}(\phi(\mathbf{x}, \theta_0)) E\left[\frac{\partial \phi(\mathbf{x}, \theta^*)}{\partial \theta}\right]^{-1}, \quad (\text{C.39})$$

which should satisfy the Cramér-Rao bound:

$$\Sigma \geq E\left[\frac{\partial \phi(\mathbf{x}, \theta_0)}{\partial \theta}\right]^{-1} I_1(\theta_0)^{-1} E\left[\frac{\partial \phi(\mathbf{x}, \theta_0)}{\partial \theta}\right]^{-1}.$$

In general, the equality holds only when $n = 1$. In general, the estimate $\tilde{\theta}_N$ given by maximizing (C.32) is not necessarily asymptotically efficient as the ML estimate $\hat{\theta}_N$.

The formulae for computing the bias and the covariance of the MEP estimate are very complicated. In practice, we can adopt random sampling techniques such as Markov Chain Monte Carlo (MCMC) method to estimate their values. With such values, we can then evaluate the accuracy and efficiency of the MEP estimate.

C.4 Model Selection Criteria

Many important model-selection criteria have been developed in the algorithmic complexity community and the statistics community for general classes of models. These criteria include minimum message length (MML) [Wallace and Boulton, 1968, Wallace and Dowe, 1999], minimum description length (MDL) [Rissanen, 1978, Hansen and Yu, 2001], Bayesian information criterion (BIC), Akaike information criterion (AIC) [Akaike, 1977], and Geometric AIC (G-AIC) [Kanatani, 2003].

All these criteria are very similar in nature: They all aim to strike a *balance* between the model complexity (typically measured by the dimension of the parameter space) and the data fidelity to the chosen model (typically measured as the sum of squares of the residuals).

Let us denote the projection of each data point $\mathbf{x}_i \in \mathbf{X}$ to the closest subspace as $\hat{\mathbf{x}}_i$ and let $\hat{\mathbf{X}} = \{\hat{\mathbf{x}}_i\}$. Then, the total error residual can be measured by:

$$\|\mathbf{X} - \hat{\mathbf{X}}\|^2 = \sum_{i=1}^N \|\mathbf{x}_i - \hat{\mathbf{x}}_i\|^2. \quad (\text{C.40})$$

Using the Grassmannian coordinates, the dimension of the parameter space for a d -dimensional subspace in \mathbb{R}^D should be $Dd - d^2$,³ which can be used to measure the model complexity.

Thus, with a model parameter space of dimension $Dd - d^2$ and Gaussian noise of variance σ^2 , the MDL criterion (equivalent to BIC in this case) [Rissanen, 1978, Rissanen, 1999, Hansen and Yu, 2001] minimizes

$$\text{MDL} = \text{BIC} \doteq \frac{1}{2\sigma^2} \|\mathbf{X} - \hat{\mathbf{X}}\|^2 + \frac{(Dd - d^2)}{2} \log N. \quad (\text{C.41})$$

Similarly, Mallows' criterion [Mallows, 1973] minimizes the quantity

$$C_p \doteq \frac{1}{2\sigma^2} \|\mathbf{X} - \hat{\mathbf{X}}\|^2 + (Dd - d^2) - N, \quad (\text{C.42})$$

while Akaike's information criterion (AIC) [Akaike, 1977] minimizes

$$\text{AIC} \doteq \frac{1}{2\sigma^2} \|\mathbf{X} - \hat{\mathbf{X}}\|^2 + (Dd - d^2), \quad (\text{C.43})$$

which is essentially the same as Mallows' C_p when the data size N is fixed for all models in comparison. More recently, Kanatani's proposed the geometric AIC [Kanatani, 2003] which minimizes

$$\text{G-AIC} \doteq \frac{1}{2\sigma^2} \|\mathbf{X} - \hat{\mathbf{X}}\|^2 + (Dd - d^2 + Nd), \quad (\text{C.44})$$

where the extra term Nd accounts for the complexity in representing the data with respect to the chosen model.

We refer to all the above criteria loosely as *information-theoretic* model selection criteria, in the sense that most of these objectives can be interpreted as variations to minimizing the optimal code length for both the model and the data with given distributions and coding schemes [Hansen and Yu, 2001].⁴

³ $Dd - d^2$ is the dimension of the Grassmannian manifold of d -dimensional subspaces in \mathbb{R}^D . To specify a subspace, one can use the so-called Grassmannian coordinates which need exactly $Dd - d^2$ entries: starting with a $D \times d$ matrix whose columns form a basis for the subspace, perform column-reduction so that the first $d \times d$ block is the identity matrix. Then, one only needs to give the rest $(D - d) \times d$ entries to specify the subspace.

⁴Even if one chooses to compare models by their algorithmic complexity, such as the minimum message length (MML) criterion [Wallace and Boulton, 1968] (an extension of the Kolmogorov com-

C.5 Robust Statistical Methods

C.5.1 Sample Influence Function

C.5.2 Multivariate Trimming

C.5.3 Random Sample Consensus

plexity to model selection), a strong connection with the above information-theoretic criteria, such as MDL, can be readily established via Shannon's optimal coding theory [Wallace and Dowe, 1999].

References

- [Akaike, 1977] Akaike, H. (1977). A new look at the statistical model selection. *IEEE Transactions on Automatic Control*, 16(6):716–723.
- [Alessandri and Coletta, 2001] Alessandri, A. and Coletta, P. (2001). Design of Luenberger observers for a class of hybrid linear systems. In *Hybrid Systems: Computation and Control*, pages 7–18. Springer Verlag.
- [Anderson and Johnson, 1982] Anderson, B. and Johnson, C. (1982). Exponential convergence of adaptive identification and control algorithms. *Automatica*, 18(1):1–13.
- [Avidan and Shashua, 2000] Avidan, S. and Shashua, A. (2000). Trajectory triangulation: 3D reconstruction of moving points from a monocular image sequence. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 22(4):348–357.
- [Ayer and Sawhney, 1995] Ayer, S. and Sawhney, H. (1995). 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.
- [Ayer et al., 1994] Ayer, S., Schroeter, P., and Bigün, J. (1994). Segmentation of moving objects by robust motion parameter estimation over multiple frames. In *European Conference on Computer Vision*, pages 316–327.
- [Babaali and Egerstedt, 2004] Babaali, M. and Egerstedt, M. (2004). Observability of switched linear systems. In *Hybrid Systems: Computation and Control*, LNCS. Springer Verlag.
- [Balluchi et al., 2002] Balluchi, A., Benvenuti, L., Benedetto, M. D., and Sangiovanni-Vincentelli, A. (2002). Design of observers for hybrid systems. In *Hybrid Systems: Computation and Control*, volume 2289 of *LNCS*, pages 76–89. Springer Verlag.
- [Bemporad et al., 2000] Bemporad, A., Ferrari, G., and Morari, M. (2000). Observability and controllability of piece- wise affine and hybrid systems. *IEEE Trans. on Aut. Cont.*, 45(10):1864–76.

- [Bemporad et al., 2003] Bemporad, A., Garulli, A., Paoletti, S., and Vicino, A. (2003). A greedy approach to identification of piecewise affine models. In *Hybrid Systems: Computation and Control*, LNCS, pages 97–112. Springer Verlag.
- [Bemporad et al., 2001] Bemporad, A., Roll, J., and Ljung, L. (2001). Identification of hybrid systems via mixed-integer programming. In *IEEE Conf. on Decision & Control*, pages 786–792.
- [Billio et al., 1999] Billio, M., Monfort, A., and Robert, C. (1999). Bayesian estimation of switching ARMA models. *Journal of Econometrics*, 93(2):229–255.
- [Black and Anandan, 1991] Black, M. and Anandan, P. (1991). Robust dynamic motion estimation over time. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 296–302.
- [Black and Anandan, 1996] Black, M. and Anandan, P. (1996). The robust estimation of multiple motions: Parametric and piecewise-smooth flow fields. *Computer Vision and Image Understanding*, 63(1):75–104.
- [Blake et al., 1999] Blake, A., North, B., and Isard, M. (1999). Learning multi-class dynamics. *Advances in Neural Information Processing Systems*, 11:389–395. MIT Press.
- [Bochnak et al., 1998] Bochnak, J., Coste, M., and Roy, M. F. (1998). *Real Algebraic Geometry*. Springer.
- [Boult and Brown, 1991] Boult, T. and Brown, L. (1991). Factorization-based segmentation of motions. In *Proc. of the IEEE Workshop on Motion Understanding*, pages 179–186.
- [Broomhead and Kirby, 2000] Broomhead, D. S. and Kirby, M. (2000). A new approach to dimensionality reduction theory and algorithms. *SIAM Journal of Applied Mathematics*, 60(6):2114–2142.
- [Collins et al., 2001] Collins, M., Dasgupta, S., and Schapire, R. (2001). A generalization of principal component analysis to the exponential family. In *Neural Information Processing Systems*, volume 14.
- [Collins and Schuppen, 2004] Collins, P. and Schuppen, J. V. (2004). Observability of piecewise-affine hybrid systems. In *Hybrid Systems: Computation and Control*, LNCS. Springer Verlag.
- [Costeira and Kanade, 1998] Costeira, J. and Kanade, T. (1998). A multibody factorization method for independently moving objects. *International Journal of Computer Vision*, 29(3):159–179.
- [Cover and Thomas, 1991] Cover, T. and Thomas, J. (1991). *Information Theory*. John Wiley & Sons, Inc.
- [Darrel and Pentland, 1991] Darrel, T. and Pentland, A. (1991). Robust estimation of a multi-layered motion representation. In *IEEE Workshop on Visual Motion*, pages 173–178.
- [Dempster et al., 1977a] Dempster, A., Laird, N., and Rubin, D. (1977a). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, 39(B):1–38.
- [Dempster et al., 1977b] Dempster, A., Laird, N., and Rubin, D. (1977b). Maximum likelihood from incomplete data via the em algorithm (with discussion). *Journal of the Royal Statistical Society B*, 39:1–38.

- [Doucet et al., 2000] Doucet, A., Logothetis, A., and Krishnamurthy, V. (2000). Stochastic sampling algorithms for state estimation of jump Markov linear systems. *IEEE Transactions on Automatic Control*, 45(1):188–202.
- [Duda et al., 2000] Duda, R., Hart, P., and Stork, D. (2000). *Pattern Classification*. Wiley, New York, 2nd edition.
- [Eckart and Young, 1936] Eckart, C. and Young, G. (1936). The approximation of one matrix by another of lower rank. *Psychometrika*, 1:211–218.
- [Eisenbud, 1996] Eisenbud, D. (1996). *Commutative Algebra: with a view towards algebraic geometry*. GTM. Springer.
- [Ezzine and Haddad, 1989] Ezzine, J. and Haddad, A. H. (1989). Controllability and observability of hybrid systems. *International Journal of Control*, 49(6):2045–2055.
- [Feng and Perona, 1998] Feng, X. and Perona, P. (1998). Scene segmentation from 3D motion. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 225–231.
- [Ferrari-Trecate et al., 2002] Ferrari-Trecate, G., Mignone, D., and Morari, M. (2002). Moving horizon estimation for hybrid systems. *IEEE Transactions on Automatic Control*, 47(10):1663–1676.
- [Ferrari-Trecate et al., 2003] Ferrari-Trecate, G., Muselli, M., Liberati, D., and Morari, M. (2003). A clustering technique for the identification of piecewise affine systems. *Automatica*, 39(2):205–217.
- [Fischler and Bolles, 1981] Fischler, M. A. and Bolles, R. C. (1981). RANSAC random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 26:381–395.
- [Fitzgibbon and Zisserman, 2000] Fitzgibbon, A. and Zisserman, A. (2000). Multibody structure and motion: 3D reconstruction of independently moving objects. In *European Conference on Computer Vision*, pages 891–906.
- [Forgy, 1965] Forgy, E. (1965). Cluster analysis of multivariate data: efficiency vs. interpretability of classifications (abstract). *Biometrics*, 21:768–769.
- [Gabriel, 1978] Gabriel, K. R. (1978). Least squares approximation of matrices by additive and multiplicative models. *J. R. Statist. Soc. B*, 40:186–196.
- [Ghahramani and Hinton, 1998] Ghahramani, Z. and Hinton, G. E. (1998). Variational learning for switching state-space models. *Neural Computation*, 12(4):963–996.
- [Han and Kanade, 2000] Han, M. and Kanade, T. (2000). Reconstruction of a scene with multiple linearly moving objects. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 542–549.
- [Han and Kanade, 2001] Han, M. and Kanade, T. (2001). Multiple motion scene reconstruction from uncalibrated views. In *IEEE International Conference on Computer Vision*, volume 1, pages 163–170.
- [Hansen and Yu, 2001] Hansen, M. and Yu, B. (2001). Model selection and the principle of minimum description length. *Journal of American Statistical Association*, 96:746–774.
- [Harris, 1992] Harris, J. (1992). *Algebraic Geometry: A First Course*. Springer-Verlag.
- [Hartley and Vidal, 2004] Hartley, R. and Vidal, R. (2004). The multibody trifocal tensor: Motion segmentation from 3 perspective views. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume I, pages 769–775.

- [Hartley and Zisserman, 2000] Hartley, R. and Zisserman, A. (2000). *Multiple View Geometry in Computer Vision*. Cambridge.
- [Hastie, 1984] Hastie, T. (1984). Principal curves and surfaces. *Technical Report, Stanford University*.
- [Hastie and Stuetzle, 1989] Hastie, T. and Stuetzle, W. (1989). Principal curves. *Journal of the American Statistical Association*, 84(406):502–516.
- [Heyden and Åström, 1997] Heyden, A. and Åström, K. (1997). Algebraic properties of multilinear constraints. *Mathematical Methods in Applied Sciences*, 20(13):1135–62.
- [Hirsch, 1976] Hirsch, M. (1976). *Differential Topology*. Springer.
- [Ho et al., 2003] Ho, J., Yang, M.-H., Lim, J., Lee, K.-C., and Kriegman, D. (2003). Clustering appearances of objects under varying illumination conditions. In *IEEE Conference on Computer Vision and Pattern Recognition*.
- [Hotelling, 1933] Hotelling, H. (1933). Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology*, 24:417–441.
- [Householder and Young, 1938] Householder, A. S. and Young, G. (1938). Matrix approximation and latent roots. *America Math. Mon.*, 45:165–171.
- [Huang et al., 2004] Huang, K., Ma, Y., and Vidal, R. (2004). Minimum effective dimension for mixtures of subspaces: A robust GPCA algorithm and its applications. In *IEEE Conference on Computer Vision and Pattern Recognition*.
- [Hubert et al., 2000] Hubert, L., Meulman, J., and Heiser, W. (2000). Two purposes for matrix factorization: A historical appraisal. *SIAM Review*, 42(1):68–82.
- [Hwang et al., 2003] Hwang, I., Balakrishnan, H., and Tomlin, C. (2003). Observability criteria and estimator design for stochastic linear hybrid systems. In *European Control Conference*.
- [Irani, 1999] Irani, M. (1999). Multi-frame optical flow estimation using subspace constraints. In *IEEE International Conference on Computer Vision*, pages 626–633.
- [Irani and Anandan, 1999] Irani, M. and Anandan, P. (1999). About direct methods. In *Workshop on Vision Algorithms*, pages 267–277.
- [Irani et al., 1992] Irani, M., Roussel, B., and Peleg, S. (1992). Detecting and tracking multiple moving objects using temporal integration. In *European Conference on Computer Vision*, pages 282–287.
- [Jancey, 1966] Jancey, R. (1966). Multidimensional group analysis. *Austral. J. Botany*, 14:127–130.
- [Jepson and Black, 1993] Jepson, A. and Black, M. (1993). Mixture models for optical flow computation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 760–761.
- [Jolliffe, 1986] Jolliffe, I. (1986). *Principal Component Analysis*. Springer-Verlag.
- [Jolliffe, 2002] Jolliffe, I. (2002). *Principal Component Analysis*. Springer-Verlag, 2nd edition.
- [Jolliffe, 1986] Jolliffe, I. (1986). *Principal Component Analysis*. Springer-Verlag, New York.
- [Juloski et al., 2004] Juloski, A., Heemels, W., and Ferrari-Trecate, G. (2004). Data-based hybrid modelling of the component placement process in pick-and-place machines. In *Control Engineering Practice*. Elsevier.

- [Kanatani, 2001] Kanatani, K. (2001). Motion segmentation by subspace separation and model selection. In *IEEE International Conference on Computer Vision*, volume 2, pages 586–591.
- [Kanatani, 2002] Kanatani, K. (2002). Evaluation and selection of models for motion segmentation. In *Asian Conference on Computer Vision*, pages 7–12.
- [Kanatani, 2003] Kanatani, K. (2003). How are statistical methods for geometric inference justified? In *Workshop on Statistical and Computational Theories of Vision, IEEE International Conference on Computer Vision*.
- [Kanatani and Matsunaga, 2002a] Kanatani, K. and Matsunaga, C. (2002a). Estimating the number of independent motions for multibody motion segmentation. In *Asian Conf. on Computer Vision*.
- [Kanatani and Matsunaga, 2002b] Kanatani, K. and Matsunaga, C. (2002b). Estimating the number of independent motions for multibody motion segmentation. In *European Conference on Computer Vision*, pages 25–31.
- [Kanatani and Sugaya, 2003] Kanatani, K. and Sugaya, Y. (2003). Multi-stage optimization for multi-body motion segmentation. In *Australia-Japan Advanced Workshop on Computer Vision*, pages 335–349.
- [Ke and Kanade, 2002] Ke, Q. and Kanade, T. (2002). A robust subspace approach to layer extraction. In *IEEE Workshop on Motion and Video Computing*, pages 37–43.
- [Lang, 1993] Lang, S. (1993). *Algebra*. Addison-Wesley Publishing Company, 3rd edition.
- [Leonardis et al., 2002] Leonardis, A., Bischof, H., and Maver, J. (2002). Multiple eigenspaces. *Pattern Recognition*, 35(11):2613–2627.
- [Lloyd, 1957] Lloyd, S. (1957). Least squares quantization in PCM. *Technical Report, Bell Laboratories, Published in 1982 in IEEE Trans. Inf. Theory* 28: 128-137.
- [Longuet-Higgins, 1981] Longuet-Higgins, H. C. (1981). A computer algorithm for reconstructing a scene from two projections. *Nature*, 293:133–135.
- [Ma et al., 2004] Ma, Y., Huang, K., Vidal, R., Košecká, J., and Sastry, S. (2004). Rank conditions on the multiple view matrix. *International Journal of Computer Vision*, 59(2):115–137.
- [Ma et al., 2003] Ma, Y., Soatto, S., Kosecka, J., and Sastry, S. (2003). *An Invitation to 3D Vision: From Images to Geometric Models*. Springer Verlag.
- [MacQueen, 1967] MacQueen, J. (1967). Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, pages 281–297.
- [Mallows, 1973] Mallows, C. (1973). Some comments on C_p . *Technometrics*, 15:661–675.
- [Martin et al., 2001] Martin, D., Fowlkes, C., Tal, D., and Malik, J. (2001). 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.
- [Murphy, 1998] Murphy, K. (1998). Switching Kalman filters. Technical report, U. C. Berkeley.

- [Neal and Hinton, 1998] Neal, R. and Hinton, G. (1998). A view of the EM algorithm that justifies incremental, sparse, and other variants. *Learning in Graphical Models, M. Jordan (ed.)*, Kluwer Academic Publishers, Boston, pages 355–368.
- [Niessen and A.Juloski, 2004] Niessen, H. and A.Juloski (2004). Comparison of three procedures for identification of hybrid systems. In *Conference on Control Applications*.
- [Pavlovic et al., 1999] Pavlovic, V., Rehg, J. M., Cham, T. J., and Murphy, K. P. (1999). A dynamic Bayesian network approach to figure tracking using learned dynamic models. In *Proc. of the Intl. Conf. on Comp. Vision*, pages 94–101.
- [Pearson, 1901] Pearson, K. (1901). On lines and planes of closest fit to systems of points in space. *The London, Edinburgh and Dublin Philosophical Magazine and Journal of Science*, 2:559–572.
- [Rissanen, 1978] Rissanen, J. (1978). Modeling by shortest data description. *Automatica*, 14:465–471.
- [Rissanen, 1999] Rissanen, J. (1999). Hypothesis selection and testing by the MDL principle. *The Computer Journal*, 42(4):260–269.
- [Roweis and L.Saul, 2000] Roweis, S. and L.Saul (2000). Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326.
- [Santis et al., 2003] Santis, E., Benedetto, M. D., and Giordano, P. (2003). On observability and detectability of continuous-time linear switching systems. In *IEEE Conf. on Decision & Control*.
- [Scholkopf et al., 1998] Scholkopf, B., Smola, A., and Muller, K.-R. (1998). Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10:1299–1319.
- [Shakernia et al., 2003] Shakernia, O., Vidal, R., and Sastry, S. (2003). Multi-body motion estimation and segmentation from multiple central panoramic views. In *IEEE International Conference on Robotics and Automation*.
- [Shashua and Levin, 2001] Shashua, A. and Levin, A. (2001). 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.
- [Shi and Malik, 1998] Shi, J. and Malik, J. (1998). Motion segmentation and tracking using normalized cuts. In *IEEE International Conference on Computer Vision*, pages 1154–1160.
- [Shizawa and Mase, 1991] Shizawa, M. and Mase, K. (1991). A unified computational theory for motion transparency and motion boundaries based on eigenenergy analysis. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 289–295.
- [Spoerri and Ullman, 1987] Spoerri, A. and Ullman, S. (1987). The early detection of motion boundaries. In *IEEE International Conference on Computer Vision*, pages 209–218.
- [Sturm, 2002] Sturm, P. (2002). Structure and motion for dynamic scenes - the case of points moving in planes. In *European Conference on Computer Vision*, pages 867–882.
- [Sun et al., 2002] Sun, A., Ge, S. S., and Lee, T. H. (2002). Controllability and reachability criteria for switched linear systems. *Automatica*, 38:775–786.

- [Szigeti, 1992] Szigeti, F. (1992). A differential algebraic condition for controllability and observability of time varying linear systems. In *IEEE Conf. on Decision & Control*, pages 3088–3090.
- [Tenenbaum et al., 2000] Tenenbaum, J. B., de Silva, V., and Langford, J. C. (2000). A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323.
- [Tipping and Bishop, 1999a] Tipping, M. and Bishop, C. (1999a). Mixtures of probabilistic principal component analyzers. *Neural Computation*, 11(2).
- [Tipping and Bishop, 1999b] Tipping, M. and Bishop, C. (1999b). Probabilistic principal component analysis. *Journal of the Royal Statistical Society*, 61(3):611–622.
- [Tomasi and Kanade, 1992] Tomasi, C. and Kanade, T. (1992). Shape and motion from image streams under orthography. *International Journal of Computer Vision*, 9(2):137–154.
- [Torr et al., 2001] Torr, P., Szeliski, R., and Anandan, P. (2001). An integrated Bayesian approach to layer extraction from image sequences. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 23(3):297–303.
- [Torr, 1998] Torr, P. H. S. (1998). Geometric motion segmentation and model selection. *Phil. Trans. Royal Society of London*, 356(1740):1321–1340.
- [Tugnait, 1982] Tugnait, J. K. (1982). Detection and estimation for abruptly changing systems. *Automatica*, 18(5):607–615.
- [Vapnik, 1995] Vapnik, V. (1995). *The Nature of Statistical Learning Theory*. Springer, N.Y.
- [Vecchio and Murray, 2004] Vecchio, D. D. and Murray, R. (2004). Observers for a class of hybrid systems on a lattice. In *Hybrid Systems: Computation and Control*. Springer Verlag.
- [Vidal et al., 2002a] Vidal, R., Chiuso, A., and Soatto, S. (2002a). Observability and identifiability of jump linear systems. In *IEEE Conf. on Decision & Control*, pages 3614–3619.
- [Vidal et al., 2003a] Vidal, R., Chiuso, A., Soatto, S., and Sastry, S. (2003a). Observability of linear hybrid systems. In *Hybrid Systems: Computation and Control*, pages 526–539. Springer Verlag.
- [Vidal and Hartley, 2004] Vidal, R. and Hartley, R. (2004). Motion segmentation with missing data by PowerFactorization and Generalized PCA. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume II, pages 310–316.
- [Vidal and Ma, 2004] Vidal, R. and Ma, Y. (2004). A unified algebraic approach to 2-D and 3-D motion segmentation. In *European Conference on Computer Vision*, pages 1–15.
- [Vidal et al., 2004] Vidal, R., Ma, Y., and Piazzoli, J. (2004). A new GPCA algorithm for clustering subspaces by fitting, differentiating and dividing polynomials. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume I.
- [Vidal et al., 2003b] Vidal, R., Ma, Y., and Sastry, S. (2003b). Generalized Principal Component Analysis (GPCA). In *IEEE Conference on Computer Vision and Pattern Recognition*, volume I, pages 621–628.
- [Vidal et al., 2005] Vidal, R., Ma, Y., Soatto, S., and Sastry, S. (2005). Two-view multibody structure from motion. *International Journal of Computer Vision*.

- [Vidal and Sastry, 2002] Vidal, R. and Sastry, S. (2002). Segmentation of dynamic scenes from image intensities. In *IEEE Workshop on Motion and Video Computing*, pages 44–49.
- [Vidal and Sastry, 2003] Vidal, R. and Sastry, S. (2003). Optimal segmentation of dynamic scenes from two perspective views. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 281–286.
- [Vidal et al., 2002b] Vidal, R., Soatto, S., Ma, Y., and Sastry, S. (2002b). Segmentation of dynamic scenes from the multibody fundamental matrix. In *ECCV Workshop on Visual Modeling of Dynamic Scenes*.
- [Wallace and Boulton, 1968] Wallace, C. and Boulton, D. (1968). An information measure for classification. *The Computer Journal*, 11:185–194.
- [Wallace and Dowe, 1999] Wallace, C. and Dowe, D. (1999). Minimum message length and Kolmogorov complexity. *The Computer Journal*, 42(4):270–283.
- [Wang and Adelson, 1993] Wang, J. and Adelson, E. (1993). Layered representation for motion analysis. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 361–366.
- [Weiss, 1996] Weiss, Y. (1996). A unified mixture framework for motion segmentation: incorporating spatial coherence and estimating the number of models. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 321–326.
- [Weiss, 1997] Weiss, Y. (1997). Smoothness in layers: Motion segmentation using non-parametric mixture estimation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 520–526.
- [Wolf and Shashua, 2001a] Wolf, L. and Shashua, A. (2001a). Affine 3-D reconstruction from two projective images of independently translating planes. In *IEEE International Conference on Computer Vision*, pages 238–244.
- [Wolf and Shashua, 2001b] Wolf, L. and Shashua, A. (2001b). Two-body segmentation from two perspective views. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 263–270.
- [Wu et al., 2001] Wu, Y., Zhang, Z., Huang, T., and Lin, J. (2001). Multibody grouping via orthogonal subspace decomposition. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 252–257.
- [Zelnik-Manor and Irani, 2003] Zelnik-Manor, L. and Irani, M. (2003). Degeneracies, dependencies and their implications in multi-body and multi-sequence factorization. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 287–293.