DISSERTATION

LINEAR MODELS, SIGNAL DETECTION, AND THE GRASSMANN MANIFOLD

Submitted by

Anthony Norbert Schwickerath

Department of Mathematics

In partial fulfillment of the requirements

For the Degree of Doctor of Philosophy

Colorado State University

Fort Collins, Colorado

Fall 2014

Doctoral Committee:

Advisor: Michael Kirby
Co-Advisor: Chris Peterson

Louis Scharf
Richard Eykholt

ABSTRACT

LINEAR MODELS, SIGNAL DETECTION, AND THE GRASSMANN MANIFOLD

Standard approaches to linear signal detection, reconstruction, and model identification problems, such as matched subspace detectors (MF, MDD, MSD, and ACE) and anomaly detectors (RX) are derived in the ambient measurement space using statistical methods (GLRT, regression). While the motivating arguments are statistical in nature, geometric interpretations of the test statistics are sometimes developed after the fact. Given a standard linear model, many of these statistics are invariant under orthogonal transformations, have a constant false alarm rate (CFAR), and some are uniformly most powerful invariant (UMPI). These properties combined with the simplicity of the tests have led to their widespread use.

In this dissertation, we present a framework for applying real-valued functions on the Grassmann manifold in the context of these same signal processing problems. Specifically, we consider linear subspace models which, given assumptions on the broadband noise, correspond to Schubert varieties on the Grassmann manifold. Beginning with increasing (decreasing) or Schur-convex (-concave) functions of principal angles between pairs of points, of which the geodesic and chordal distances (or probability distribution functions) are examples, we derive the associated point-to-Schubert variety functions and present signal detection and reconstruction algorithms based upon this framework.

As a demonstration of the framework in action, we implement an end-to-end system utilizing our framework and algorithms. We present results of this system processing real hyperspectral images.

ACKNOWLEDGEMENTS

Any dissertation is the culmination of a decades-long journey, and this one is not exception. Unfortunately, it is not possible to acknowledge every teacher, friend, or family member who nurtured me or nudged me along, though I am grateful to you all.

The research program presented here is the product of regular meetings with Michael Kirby, Chris Peterson, and Louis Scharf. Each brought a different view of signal detection on the Grassmann manifold, whether statistical, geometric, theoretic, or data-directed. The connections made between these various views form the scaffolding upon which this work is built, and it would not have developed without their collaboration and guidance. Thank you for your time, wisdom, and patience.

Frequent opportunities to present my work to the Pattern Analysis Lab forced me to organize my ideas. Discussions with members of the lab about their own work broadened my perspectives, exposing me to ideas like the mean flag. Thank you Sofya Chepushtanova, Tegan Emerson, and Tim Marinnan; your camaraderie kept my research and myself grounded..

Financial support from the Department of Defense Science, Mathematics, & Research for Transformation (SMART) Scholarship program allowed me to focus on completing this work.

I would not have had this opportunity without the early encouragement from my family. Thank you mom (my advocate), dad (building a geodesic dome **is** an exercise in applied geometry), Julie, Angie, and Jessie.

Lastly, I would like to thank my wife, Kristi. It has been said that earning a PhD is a test of endurance and her love, encouragement, and baking kept me going throughout this journey. Words are insufficient express my thanks.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

Introduction

1.1. Signal Detection and Related Tasks

In general, **signal (or signature) detection** is the task of determining whether a signal occurs in a data set. For example, we might ask whether a broadcast radio signal exists in a specific frequency band or if the channel only consists of noise. The data can be a time series, as in this example, spatial data, or spatio-temporal data. Spatial data might be an image or measurements made by sensors placed at discrete points over a geographic area. Spatio-temporal data consists of a number of time series, each identified with a spatial location. For example, it may take the form of a sequence of images (a movie) or time series from a number of geographically distributed weather stations. Data can even come from non-physical phenomena, such as computer network traffic. Despite the different provenance of the data, the same detection algorithms may be applied to each.

In this dissertation, we will be concerned with methods which utilize an explicit data formation model. Specifically, we are interested in linear models, meaning models where the signal and noise components act independently from each other and are combined by simple addition. While this may not be a strictly accurate model in every problem domain, it is a frequently employed simplification. The widespread success of linear models suggests that this is often a useful approximation.

We differentiate between the detection task and **classification** task. In detection, we attempt to identify the presence or absense of a signal. We can think of this as a single class problem. In classification, there are multiple signal classes and we attempt to determine which one is present in the data. In our radio broadcast example, we might wish to distinguish

between talk, music, and Morse code. In some domains, we assume that the data can only contain a single signal; while in others, the data may contain a mixture of multiple classes. For example, in an image taken from an airplane, a given pixel might represent a 10m $\times$ 10m patch on the ground. A patch at the edge of a parking lot may contain a mixture of grass, tree, asphalt, and automobile. In some applications, we might wish to determine the percentage of each substance in the pixel, in others, to simply identify the class which is most prevalent.

All of the previous examples have involved a known signature that we are attempting to locate in the data. Sometimes we do not have a signature, but are simply looking for a deviation from the bulk of the data set. This task is known as **anomaly detection**. While our primary focus in this work is on signal detection, our approach may be applied to anomaly detection.

After we have detected a signal, we may wish to determine the signal which is embedded in the data. Returning to the radio example, once we know that there is a broadcast signal, we would like to reconstruct the original signal free of noise and interference, so that we can listen to it. We refer to this as **signal recovery**.

In order to perform these tasks, we first need a model. Sometimes these models are derived theoretically from first principles. In many scenarios, either due to the complex interactions or to lack of knowledge, we would be satisfied with a model which fits a training data set well. For example, if we are attempting to detect roads in an image, we might fit a model to some hand labeled examples of road pixels. We refer to the task of fitting a model to training data as **model estimation**.

Signal detection, signal recovery, and model estimation are very closely related. Solutions to these generally revolve around a single theoretical framework. Often there is an underlying statistical or geometric approach tying constellations of algorithms together.

1.2. Challenges

Currently, standard methods such as the matched subspace detector (MSD) are based upon a common linear model and are stated as generalized likelihood ratio tests (GLRT). These methods are developed in data space. They have the advantage of a directness and simplicity in both derivation and application. Some also have a constant false alarm rate (CFAR) and are uniformly most powerful invariant (UMPI). Many of these methods were originally presented by Scharf in [1] with variations and additional properties proven later, e.g., in [2–4].

There are some examples of detection or classification performed in derived spaces or manifolds, such as the Grassmann manifold [5–9]. To the best of our knowledge, these are each developed on a problem-by-problem basis. We have not found a general framework for developing a detector in a principled fashion along the lines used to motivate the standard data-space methods.

In this dissertation, we lay out a research program to develop similar general methods on the Grassmann manifold and present our initial results. This path contains some clear challenges.

- We have found no existing general signal processing framework on the Grassmann manifold.
- Mapping interesting probability distributions from a vector space to the Grassmann manifold is frequently difficult.

1.3. Overview

In Chapter 2, we provide some general background material. First we define the signal detection problem. Then we focus on the linear subspace models which appear in a number

of contexts. Based upon these, we present an overview of some classic signal detectors such as the Matched Subspace Detector (MSD) and the Adaptive Coherence/Cosine Estimator (ACE). We then present methods for constructing a linear model given labeled data. Finally, we introduce the Grassmann manifold, the setting in which we will explore signal detection and recovery throughout the remainder of this dissertation.

In Chapter 3, we develop a set of geometric tests for comparing aggregate data with more general linear models on the Grassmann manifold. First we introduce some classes of point-to-point functions on the Grassmann manifold. Then we use these functions to generate functions for comparing a point with a Schubert variety. Armed with these functions, we present signal detection and signal recovery algorithms.

Based upon the theory developed in Chapter 3, we present an example end-to-end system for detection and classification of pixels in hyperspectral imagery (HSI) in Chapter 4. This begins with model estimation using standard methods. Then detection and classification are performed using our method. This system is demonstrated on real imagery and an exploration of challenges is presented in this context.

Finally, we recap in Chapter 5. Then we discuss open questions and data sets where these techniques might yield interesting and useful results.

CHAPTER 2

Background

2.1. Introduction

Our ultimate goal is to produce a signal detection framework on the Grassmann manifold. Before we can do that, we need to discuss similar work on vector spaces, specifically linear subspace models and classic signal detectors derived from these models. These models will provide a starting point for those we will assume in Chapter 3. We will then address some methods for fitting a linear subspace model to training data. Finally, we will introduce the Grassmann manifold. Throughout this chapter, we will be developing a notation which will prove essential to discussing our results in Chapter 3.

2.2. Linear Subspace Models

As a simple example, suppose we have a temporal data set. We have n sensors and every so often (say once every second) we record the measurements of these sensors. This may be modeled as three components: signal, clutter, and noise. Signal is the part of the measurements that we are ultimately most interested in; this is the part that tells us about the true state of the world as it applies to our application. Clutter is also sometimes referred to as background or interference, depending upon the application. This is the part of the world which is signal dependent but does not apply to our question; in fact, it may actually obscure our ability to discern the signal. Finally, noise is distinct from clutter in that it is a random process, independent of the signal.

In our example, let the data at time i be organized in a vector, $\boldsymbol{x}_i \in \mathbb{R}^n$; each element in the vector is the measurement from one of the n sensors. There are many ways in which signal, clutter, and noise can interact to produce $\boldsymbol{x}_i$. One of the simplest is if they are simply

combined additively:

$$\boldsymbol{x}_i = \text{signal} + \text{clutter} + \text{noise}.$$

To be a useful model, we need to identify the sets of possible values for signal and clutter, as well as a distribution for noise, so that we can tease apart the components which comprise our measurements.

Once again, let us take a simple approach. Suppose signal and clutter each lives in a linear subspace. We can write a model like this as

$$\boldsymbol{x}_i = \mathbf{S}\boldsymbol{\psi}_i + \mathbf{C}\boldsymbol{\phi}_i + \boldsymbol{\nu}_i,$$

where the column spaces of the fixed (and usually known) $\mathbf{S} \in \mathbb{R}^{n\times r}$ and $\mathbf{C} \in \mathbb{R}^{n\times p}$ are the linear subspaces where the signal and clutter reside, respectively.[1] The vectors $\boldsymbol{\psi}_i \in \mathbb{R}^r$ and $\boldsymbol{\phi}_i \in \mathbb{R}^p$ are the basis-specific coordinates in the signal and noise subspaces. The n-dimensional random vector $\boldsymbol{\nu}_i$ is distributed as the noise is, say $N_n[\mathbf{0}, \sigma^2\mathbf{I}_n]$. We call such a model a **linear subspace model**.

Linear subspace models make up one of the most popular classes used in signal detection. While the general idea is consistent, there are variations on this theme used in the literature. The simplest form,

$$\boldsymbol{x}_i = \mathbf{S}\boldsymbol{\psi}_i + \boldsymbol{\nu}_i \tag{2.1}$$

[1]When we are referring to the column space of a matrix, we will sometimes write $\boldsymbol{\mathcal{S}} = \langle\mathbf{S}\rangle$ to explicitly indicate that linear subspace $\boldsymbol{\mathcal{S}}$ is the span of the columns of $\mathbf{S}$. By convention, we use the same letter. A boldface $\mathbf{S}$ indicates the matrix encoding a specific fixed basis. A script $\boldsymbol{\mathcal{S}}$ indicates the abstract linear subspace.

can be read two different ways. First, it can mean that there is no clutter, only signal and noise. This is the way in which we will use this form. Alternately, some papers use this notation when clutter still exists but is written as the mean of the distribution of $\boldsymbol{\nu}_i$. We do not favor that use of notation, as we prefer $\boldsymbol{\nu}_i$ to be identically distributed for all i.

In some situations, we may believe that spatial data collected in a small region or temporal data collected over a short period of time is more restricted than the previous model would show. If we believe that this is the case, we may write the model as[2]

$$\boldsymbol{x}_i = \mathbf{S}\mathbf{M}\boldsymbol{\psi}_i + \mathbf{C}\boldsymbol{\phi}_i + \boldsymbol{\nu}_i. \tag{2.2}$$

Here a fixed but unknown mixture matrix $\mathbf{M} \in \mathbb{R}^{r\times q}$ selects a subspace of $\boldsymbol{\mathcal{S}}$ for the data, as is done for the Matched Direction Detector (MDD) to be presented in Section 2.3.3. Now that the restricted signal subspace $\langle \mathbf{SM} \rangle$ is q-dimensional rather than r-dimensional, $\boldsymbol{\psi}_i \in \mathbb{R}^q$ is the signal for the i-th measurement (in the coordinate system defined by the columns of $\mathbf{SM}$), $\boldsymbol{x}_i$.

This formulation draws a clear distinction between the signal, correlated background clutter, and broadband noise which is often lacking in work that simply uses the signature-noise model of Equation (2.1), such as that surveyed in [11, 12]. The signature-background-noise model considers $\boldsymbol{\nu}_i$ to represent only broadband noise, such as sensor noise. In some formulations written as Equation (2.1), the clutter term still exists but is given as the mean of the noise distribution. In others, rather than being broadband noise, $\boldsymbol{\nu}_i$ has a rank-deficient covariance matrix and so constrains the noise to live in a subspace. In this case, the noise may be dominated by clutter, rather than sensor noise.

[2]While this draws on the explanation given by Scharf, et al. in [10], some letters chosen to denote the different components are altered to be more mnemonic or to avoid conflict with other standard notation.

2.3. Classical Subspace Detectors

Given the general statement of the model in Equation (2.2), we may still have differences in what is known. In some we know all of the model parameters: the signature and clutter subspaces, as well as the true noise distribution. In some we know these but not the exact portion of the signature subspace where the local measurements lie. In others we know the subspaces but not the noise parameters. In the final type, the RX anomaly detector, we only know the general form of the model a priori, but not the exact signature subspace or noise parameters.

Definition 2.3.1. *Assume a family of distributions with parameter space Θ. Let $\boldsymbol{X}$ be a random vector with a fixed but unknown distribution selected from this family. Let $\{\Theta_0, \Theta_1\}$ be a partition of Θ and hypotheses H_0 and H_1 be that the distribution have parameters in Θ_0 and Θ_1, respectively. We refer to H_0 as the **null hypothesis** and H_1 as the **alternate hypothesis**.*

*Given a realization $\boldsymbol{x}$ of $\boldsymbol{X}$, the likelihood function $l(\theta; \boldsymbol{x}) = f(\boldsymbol{x}; \theta)$, the probability density function at $\boldsymbol{x}$ conditioned on $\theta \in \Theta$. The **generalized likelihood ratio** is*

$$L(\boldsymbol{x}) = \frac{\sup_{\theta \in \Theta_1} l(\theta; \boldsymbol{x})}{\sup_{\theta \in \Theta_0} l(\theta; \boldsymbol{x})}$$

*A **generalized likelihood ratio test** (GLRT) assigns a hypothesis to the data using the generalized likelihood ratio. For a threshold η, $L(\boldsymbol{x}) > \eta$ results in the assignment of $\boldsymbol{x}$ to H_1, while $L(\boldsymbol{x}) < \eta$ assigns $\boldsymbol{x}$ to H_0.*

All four of the following detectors employ GLRTs. In each, there is a definition for the null hypothesis H_0, which assumes that the measurements only contain clutter and noise, and an alternate hypothesis H_1, where signal is also present. Due to the simplicity of the

derived test statistics and the general applicability of the model assumptions, these have found widespread use in a range of signal detection and estimation problem domains. For some examples in hyperspectral imagery and radar, see [13, 12].

2.3.1. Matched Subspace Detector (MSD). Up to this point we have assumed our measurements to produce real ($\mathbb{R}$) values. A number of the signal processing papers that we will refer to assume complex ($\mathbb{C}$) values. We use the complex forms here, but the results still hold when restricted to the real domain.

The Matched Subspace Detector (MSD) was originally introduced by Scharf [1] without an interference term and expanded to the form presented here in Scharf and Friedlander [10]. The data is assumed to be produced by the model

$$\boldsymbol{x}_i = \mathbf{S}\boldsymbol{\psi}_i + \mathbf{C}\boldsymbol{\phi}_i + \boldsymbol{\nu}_i.$$

Without loss of generality, we will assume that the noise has the multivariate normal distribution $\boldsymbol{\nu}_i \sim \mathbb{C}\mathrm{N}_n\left[\mathbf{0}, \sigma^2\mathbf{I}\right]$. The hypotheses used to produce the GLRT are

$$H_0 : \boldsymbol{\psi}_i = \mathbf{0}$$

$$H_1 : \boldsymbol{\psi}_i \neq \mathbf{0}.$$

H_0 represents the case where no signal is present and H_1 where signal is present. Hence, when the likelihood ratio

$$L(\boldsymbol{x}) = \frac{l(H_1; \boldsymbol{x})}{l(H_0; \boldsymbol{x})},$$

is greater than 1, it is more likely that the data contains signal than that it does not.

If we know the variance (σ^2) of the noise, the test statistic is

$$L_{\text{MSD-1}}(\boldsymbol{x}) = \frac{1}{\sigma^2}\boldsymbol{x}^{\mathsf{H}}\mathbf{P}_{\boldsymbol{\mathcal{C}}^{\perp}}\mathbf{P}_{\boldsymbol{\mathcal{G}}}\mathbf{P}_{\boldsymbol{\mathcal{C}}^{\perp}}\boldsymbol{x},$$

where $\mathbf{P}_{\boldsymbol{\mathcal{C}}^{\perp}}$ is the orthogonal projection matrix onto the orthogonal complement of the clutter subspace ($\boldsymbol{\mathcal{C}}$). $\boldsymbol{\mathcal{G}}$ is the projection of the portion of the signal subspace which is orthogonal to the clutter subspace, $\mathbf{P}_{\boldsymbol{\mathcal{C}}^{\perp}}\boldsymbol{\mathcal{S}}$. If we do not know the variance of the noise, the test statistic is

$$L_{\text{MSD-2}}(\boldsymbol{x}) = \frac{\boldsymbol{x}^{\mathsf{H}}\mathbf{P}_{\boldsymbol{\mathcal{C}}^{\perp}}\mathbf{P}_{\boldsymbol{\mathcal{G}}}\mathbf{P}_{\boldsymbol{\mathcal{C}}^{\perp}}\boldsymbol{x}}{\boldsymbol{x}^{\mathsf{H}}\mathbf{P}_{\boldsymbol{\mathcal{C}}^{\perp}}\mathbf{P}_{\boldsymbol{\mathcal{G}}^{\perp}}\mathbf{P}_{\boldsymbol{\mathcal{C}}^{\perp}}\boldsymbol{x}}.$$

To use a detection statistic $L(\boldsymbol{x})$, we must select a threshold η. When $L(\boldsymbol{x}) < \eta$, we label $\boldsymbol{x}$ as H_0. Otherwise we label $\boldsymbol{x}$ as H_1 (containing signal). $L_{\text{MSD-2}}$ has been shown to be a Constant False Alarm Rate (CFAR) detector. This means that the threshold η can be set to provide (in a statistical sense) a constant rate of claiming H_0 events as H_1 independent of the signature subspace $\boldsymbol{\mathcal{S}}$.

Additionally, MSD has been proven to be a Uniformly Most Powerful Invariant (UMPI) detector.

DEFINITION 2.3.2. *Suppose we have a random vector X parameterized by a deterministic but unknown parameter $\theta \in \Theta$ and that Θ is partitioned into two sets, Θ_0 (which we identify with the null hypothesis H_0) and Θ_1 (identified with H_1). A detection function $d(X) \longrightarrow \{0, 1\}$ is said to be **uniformly most powerful of size** α when, for any other detection function $d'(X)$ where*

$$\sup_{\theta\in\Theta_0} E_\theta d'(X) = \alpha' \leq \alpha = \sup_{\theta\in\Theta_0} E_\theta d(X),$$

then

$$E_\theta d'(X) = 1 - \beta' \leq 1 - \beta = E_\theta d(X),$$

for all $\theta \in \Theta_1$.

Essentially what this says is that, while there may be detectors which produce fewer false positives (claim H_1 when Θ_0) in the worst case (for some $\theta \in \Theta_0$), such detectors cannot produce more correct detections (claim H_1 when Θ_1).

2.3.2. Adaptive Coherence/Cosine Estimator (ACE). Scharf and McWhorter [14] and Kraut, Scharf, and McWhorter [2] presented the Adaptive Coherence[3] Estimator(ACE) an adaptive version of the Matched Subspace Detector where the noise variance is unknown. This is based upon many of the assumptions used by MSD. The principal difference is that, rather than knowing that broadband noise is distributed as $\boldsymbol{\nu}_i \sim \mathbb{C}\mathrm{N}_n[\mathbf{0}, \sigma^2\mathbf{I}]$, we instead only assume that $\boldsymbol{\nu}_i \sim \mathbb{C}\mathrm{N}_n[\mathbf{0}, \mathbf{R}]$ and that the covariance matrix $\mathbf{R}$ is unknown. The original papers also deviate from MSD by assuming no clutter subspace. Data is assumed to be modeled by

$$\boldsymbol{x}_i = \mathbf{S}\boldsymbol{\psi}_i + \boldsymbol{\nu}_i.$$

The method used in to produce a detector to cope with this problem is to estimate the covariance with a sample covariance. Specifically, if we have M samples $\{\tilde{\boldsymbol{x}}_1, \ldots, \tilde{\boldsymbol{x}}_M\}$ known to be devoid of signal (H_0), we can produce an $n \times M$ matrix $\tilde{\mathbf{X}} = [\tilde{\boldsymbol{x}}_1, \ldots, \tilde{\boldsymbol{x}}_M]$. The sample covariance has the form $\hat{\mathbf{R}} = \tilde{\mathbf{X}}\tilde{\mathbf{X}}^{\mathsf{H}}$. Using the estimated sample covariance, we can whiten

[3]Sometimes "Cosine" is used in place of "Coherence" to emphasize a geometric rather than statistical interpretation.

the data $\hat{\boldsymbol{z}} = \hat{\mathbf{R}}^{-1/2}$. This results in the ACE detection statistic

$$\begin{aligned} L_{\mathrm{ACE}}(\boldsymbol{x}) &= \frac{\hat{\boldsymbol{z}}^{\mathsf{H}} \mathbf{P}_{\hat{\mathcal{G}}} \hat{\boldsymbol{z}}}{\hat{\boldsymbol{z}}^{\mathsf{H}} \hat{\boldsymbol{z}}} \\ &= \frac{\boldsymbol{x}^{\mathsf{H}} \hat{\mathbf{R}}^{-1} \mathbf{S} \left(\mathbf{S}^{\mathsf{H}} \hat{\mathbf{R}}^{-1} \mathbf{S} \right) \mathbf{S}^{\mathsf{H}} \hat{\mathbf{R}}^{-1} \boldsymbol{x}}{\boldsymbol{x}^{\mathsf{H}} \hat{\mathbf{R}}^{-1} \boldsymbol{x}} \end{aligned}$$

If $\dim \boldsymbol{\mathcal{S}} = 1$, we write $\boldsymbol{\mathcal{S}} = \langle \boldsymbol{s} \rangle$. Then the detection statistic can be rewritten

$$L_{\mathrm{ACE}}(\boldsymbol{x}) = \frac{|\boldsymbol{s}^{\mathsf{H}} \hat{\mathbf{R}}^{-1} \boldsymbol{x}|^2}{\boldsymbol{x}^{\mathsf{H}} \hat{\mathbf{R}}^{-1} \boldsymbol{x} \boldsymbol{s}^{\mathsf{H}} \hat{\mathbf{R}}^{-1} \boldsymbol{s}}.$$

Both of these adaptive detectors are CFAR. Later, Kraut, Scharf, and Butler [3] demonstrated that this detector is UMPI.

2.3.3. Matched Direction Detector (MDD). In some applications, we may believe that data which is nearby spatially or temporally is more similar than what the general signal subspace might suggest. The Matched Direction Detector (MDD) was originally introduced by Besson, Scharf, and Vincent [4] to address this situation. This is similar to MSD, but the signal is assumed to lie in some (unknown) 1-dimensional subspace of $\boldsymbol{\mathcal{S}}$ for all of the data. In other words, the model is

$$\boldsymbol{x}_i = \mathbf{S}\boldsymbol{\mu}\psi_i + \mathbf{C}\boldsymbol{\phi}_i + \boldsymbol{\nu}_i.$$

Here, the unknown vector $\boldsymbol{\mu} \in \mathbb{C}^r$ fixes the 1-dimensional subspace of $\boldsymbol{\mathcal{S}}$ and ψ_i is the location in this subspace. The hypotheses are the same as before. Noise is assumed to be $\mathbb{C}\mathrm{N}_n[\mathbf{0}, \sigma^2 I]$ with σ^2 known.

Because we are assuming that a collection of measurements live in a 1-dimensional linear subspace of our signal subspace, we need to answer the question about all of measurements

simultaneously. Let us take a collection of N measurements $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$ and write them as a matrix $\mathbf{X} = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N]$. The likelihood ratio test produces a detection function

$$L_{\mathrm{MDD}}(\mathbf{X}) = \frac{1}{\sigma^2}\lambda_{\max}\left(\mathbf{X}^{\mathsf{H}}\mathbf{P}_{\boldsymbol{\mathcal{G}}}\mathbf{X}\right) \underset{H_0}{\overset{H_1}{\gtrless}} \eta$$

where $\boldsymbol{\mathcal{G}} = \boldsymbol{\mathcal{C}}^{\perp} \cap \boldsymbol{\mathcal{S}}$, the part of $\boldsymbol{\mathcal{S}}$ which is orthogonal to $\boldsymbol{\mathcal{C}}$.

Besson and Scharf [15] demonstrated that the MDD is CFAR.

2.3.4. RX Anomaly Detector. Reed and Yu presented a CFAR adaptive (anomaly) detector in [16, 17]; this is now generally known in the literature as the RX (Reed-Xiaoli) detector. The data model is

$$\boldsymbol{x}_i = \boldsymbol{s}\mu_i + \boldsymbol{\nu}_i$$

where $\boldsymbol{s}$ is the common unknown signal vector and μ_i is the signal mask. The test operates on a collection of m n-dimensional data points ($m > n$) which is partitioned into a region assumed to be background ($\mu_i = 0$) and another which is the set under test ($\mu_i = 1$). A GLRT is constructed using the hypotheses

$$H_0 : \mathbf{X} = \mathbf{N}$$

$$H_1 : \mathbf{X} = \boldsymbol{s}\boldsymbol{\mu}^{\mathsf{T}} + \mathbf{N}$$

where $\mathbf{X}$ is the data matrix, $\mathbf{N} = [\boldsymbol{\nu}_1, \ldots, \boldsymbol{\nu}_m]$ is the nominal process (noise + background), and $\boldsymbol{\mu}$ is the known column vector $[\mu_1, \ldots, \mu_m]^{\mathsf{T}}$.

The test statistic derived from the GLRT is

$$L(\mathbf{X}) = \frac{(\mathbf{X}\boldsymbol{\mu})^{\mathsf{T}} \left(\mathbf{X}\mathbf{X}^{\mathsf{T}}\right)^{-1} (\mathbf{X}\boldsymbol{\mu})}{\boldsymbol{\mu}^{\mathsf{T}}\boldsymbol{\mu}} \underset{H_0}{\overset{H_1}{\gtrless}} \eta.$$

This detection function is CFAR.

2.4. Model Identification

With the exception of RX, we have focused on signal detectors which require a known signal subspace. This raises the question of how we determine the signal subspace. In some cases, such as when identifying gases in hyperspectral imagery [12], the spectral signature may be determined in a laboratory using pure samples and a spectrometer. In practice, these ideal signatures may not be representative of what is found in the field. Also, there are cases where it may be impractical or impossible to identify a signal subspace under laboratory conditions.

In cases such as these, it may be necessary to identify the signal subspace based upon field data. This is an area of active research. For three examples of determining spectral signatures (sometimes termed "endmembers") in unlabeled hyperspectral imagery, see [18–20]. In this dissertation, we focus on simple–and, in all but one case, well-known–supervised methods.

The basic form of each of these methods is as follows. First, assume that our data is modeled by

$$\boldsymbol{x}_i = \mathbf{S}\boldsymbol{\phi}_i + \boldsymbol{\nu}_i.$$

We assume that we know which measurements contain signal and which do not.[4] We then solve an optimization problem which produces an ordered basis for the space spanned by $\{\boldsymbol{x}_i\}$. The exact details of how this is done distinguishes these methods from each other. Finally, select the r basis vectors which best fit the data.

2.4.1. Principal Component Analysis (PCA). Perhaps the most well known method for identifying the subspace nearest a cloud of noisy measurements is Principal Component Analysis (PCA).

Given a zero-mean data set in $\mathbb{R}^J$, the idea behind principal components analysis is to find a linear subspace $\mathbb{R}^K$ $(K < J)$ which captures the maximum variance in the data. In other words, consider the set of N zero-mean, m-dimensional column data vectors $\mathbf{X} = \left[\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(N)}\right]$. The principal components are the basis vectors of this subspace in order of decreasing variance. The first principal component $\boldsymbol{y}^{(1)}$ solves the problem

$$\arg\max_{\boldsymbol{y}} \ \|\mathbf{X}\boldsymbol{y}\|_2^2$$

$$\text{subject to } \boldsymbol{y}^T\boldsymbol{y} = 1 \ .$$

In other words, $\boldsymbol{y}$ is the (unit) eigenvector of the covariance matrix of $\mathbf{X}$ corresponding to the largest eigenvalue. In fact, the second principal component is the (unit) eigenvector of the covariance matrix corresponding to the second largest eigenvalue. The eigenvectors of the covariance matrix $\mathbf{X}\mathbf{X}^T$ are also the left singular vectors $\mathbf{U}$ in the singular value decomposition (SVD) of $\mathbf{X}$, as noted throughout the literature, including in [21–23].

[4]In their simplest form, which we present, we only consider signal-containing measurements. By considering both data with and without signal, we would be able to produce a model $\boldsymbol{x}_i = \mathbf{S}\boldsymbol{\phi}_i + \mathbf{C}\boldsymbol{\psi}_i + \boldsymbol{\nu}_i$.

```
1: function SubspaceUsingPCA(X)
2:     (U, Σ, V) ← SVD(X)
3:     ρ ← diag(Σ)
4:     return (U, ρ)
5: end function
```

FIGURE 2.1. Algorithm for computing the best subspace basis using PCA.

Now assume that our data is modeled by the linear subspace model

$$\boldsymbol{x}_i = \mathbf{S}\boldsymbol{\phi}_i + \boldsymbol{\nu}_i$$

with $\boldsymbol{\nu}_i \sim \mathbb{C}\mathrm{N}_n\left[\mathbf{0}, \sigma^2\mathbf{I}\right]$. Then the basis for the least-squares r-dimensional subspace $\boldsymbol{\mathcal{S}}^\star$ spanning the r eigenvectors (left singular vectors) associated with the r largest eigenvalues (singular values). Pseudocode for accomplishing this is shown in Figure 2.1.

2.4.2. Maximum Noise Fraction (MNF). When applied to real data containing noise, PCA produces results that do not necessarily decrease in quality (increase in noise) with increasing dimension (r) of the subspace. To address this issue, Green, et al. [24] produced what they term the maximum noise fraction (MNF), and which is elsewhere [23] referred to as noise-adjusted PCA (NAPCA).

In the context of hyperspectral imagery, assume each pixel to be a random vector. We decompose the ith pixel to be $\boldsymbol{x}_i = \boldsymbol{s}_i + \boldsymbol{\nu}_i$, the sum of a signal component $\boldsymbol{s}_i = \left\{s_1^{(i)}, \ldots, s_n^{(i)}\right\}^\mathsf{T}$ and a noise component $\boldsymbol{\nu}_i = \left\{\nu_1^{(i)}, \ldots, \nu_n^{(i)}\right\}^\mathsf{T}$.

Define the noise fraction of the jth spectral band to be

$$\frac{\mathrm{Var}\left[\nu_j^{(i)}\right]}{\mathrm{Var}\left[x_j^{(i)}\right]},$$

```
1: function SubspaceUsingMNF(X)
2:     dX ← X − [x_{(i+1),j}]_{i,j}
3:     (U, V, Y, C, S) ← GSVD(X, dX)
4:     ρ ← ReverseColumnOrder([{c_{i,i}/s_{i,i}}_i])
5:     Z ← ReverseColumnOrder(X ∗ Y^{−T})
6:     Q ← GramSchmidtBasis(Z)                    ▷ See Figure 2.3
7:     return (Q, ρ)
8: end function
```

FIGURE 2.2. Algorithm for computing the best subspace basis using MNF.

where $\mathrm{Var}[w]$ is the variance of the random variable w. As the name "maximum noise fraction" suggests, we want the linear transformation

$$\begin{aligned}\boldsymbol{y}_i &= \mathbf{A}^T\boldsymbol{x}_i \\ &= \mathbf{A}^T\boldsymbol{s}_i + \mathbf{A}^T\boldsymbol{\nu}_i\end{aligned}$$

that solves the problem

$$\arg\max_{\mathbf{A}} \frac{\mathrm{Var}\left[\mathbf{A}^T\boldsymbol{\nu}_i\right]}{\mathrm{Var}\left[\mathbf{A}^T\boldsymbol{x}_i\right]}$$

$$\text{subject to } \mathbf{A}^T\mathbf{A} = \mathbf{I}$$

Since we do not know the covariance of the noise, $\langle\boldsymbol{\nu}_i, \boldsymbol{\nu}_i{}^{\mathsf{T}}\rangle$, Green, et al. suggest approximating it with the $\langle\boldsymbol{x}_i, \boldsymbol{x}_{i+\Delta}{}^{\mathsf{T}}\rangle$ for some offset Δ.

Originally, this was cast as the solution of two PCA problems. Roger [23] shows that solving these is equivalent to solving the generalized singular value decomposition problem. We use his method in Figure 2.2.

2.4.3. FLAG MEAN. A much more recent approach known as the flag mean is introduced by Draper, et al. in [25] and compared with other subspace means in Marrinan, et.all [26]. A

```
function GramSchmidtBasis(A)
    (n, m) ← size(A)                          ▷ A ∈ ℂ^{n×m}
    q_1 ← A/‖A‖_2                             ▷ q_i is the ith column of Q
    r ← 1
    for k = 2, ..., n do
        z_k = (I − QQ^H)A_k
        if ‖z_k‖_2 = 0 then
            r ← r + 1
            q_r ← z_k/‖z_k‖_2
        end if
    end for
    return Q
end function
```

FIGURE 2.3. Algorithm for constructing a basis for the range of a matrix using Gram-Schmidt.

subspace mean finds an average subspace of a given a collection of N subspaces $D = \{\boldsymbol{\mathcal{X}}_i\}$, where $\boldsymbol{\mathcal{X}}_i \in \boldsymbol{\mathcal{V}}$. Generally the $\boldsymbol{\mathcal{X}}_i$ are all of the same dimension.

The flag mean differs in two key respects. First, the subspaces $\boldsymbol{\mathcal{X}}_i$ do not need to have the same dimension. Second, it produces a nested sequence of linear subspaces known as a flag. Any of these subspaces can be used as a mean.

Let r be the dimension of the span of $\boldsymbol{\mathcal{X}}_1 \oplus \cdots \oplus \boldsymbol{\mathcal{X}}_N$. We wish to produce a sequence of 1-dimensional subspaces $\boldsymbol{\mathcal{U}}^{(j)}$ which solve the optimization problems

$$\boldsymbol{\mathcal{U}}^{(1)} \doteq \arg\min_{\boldsymbol{\mathcal{U}} \in \boldsymbol{\mathcal{V}}} \sum_{\boldsymbol{\mathcal{X}} \in D} d_{pF}\left(\boldsymbol{\mathcal{U}}, \boldsymbol{\mathcal{X}}\right)^2$$

and for $j = 2, \ldots, r$

$$\boldsymbol{\mathcal{U}}^{(j)} \doteq \arg\min_{\boldsymbol{\mathcal{U}} \in \boldsymbol{\mathcal{V}}} \sum_{\boldsymbol{\mathcal{X}} \in D} d_{pF}\left(\boldsymbol{\mathcal{U}}, \boldsymbol{\mathcal{X}}\right)^2$$

$$\text{subject to } \boldsymbol{\mathcal{U}}^{(j)} \perp \boldsymbol{\mathcal{U}}^{(l)}, \text{ for } l < j.$$

```
function SubspaceUsingFlagMean({X^(i)}_{i=1}^k)
    Z ← [ {GramSchmidtBasis(X^(i))}_{i=1}^k ]        ▷ See Figure 2.3
    (U, Σ, V) ← SVD(Z)
    ρ ← diag(Σ)
    return (U, ρ)
end function
```

FIGURE 2.4. Algorithm for computing the best subspace basis using the flag mean.

Here $d_{pF}(\boldsymbol{\mathcal{Y}}, \boldsymbol{\mathcal{T}})$, the projection Frobenius norm,[5] is the 2-norm of the sines of the principal angles[6] between $\boldsymbol{\mathcal{Y}}$ and $\boldsymbol{\mathcal{T}}$.

This problem can be solved using the SVD. Let the columns of the matrix $\mathbf{X}_i$ form a basis for $\boldsymbol{\mathcal{X}}_i$ and $\mathbf{X}_i{}^{\mathsf{T}}\mathbf{X}_i = \mathbf{I}$. Construct a matrix

$$\mathbf{X} = [\mathbf{X}_1 | \cdots | \mathbf{X}_N] .$$

If we take the SVD of $\mathbf{X} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^{\mathsf{T}}$, then for $j = 1, \ldots, r$, $\boldsymbol{\mathcal{U}}^{(j)}$ is the span of the j-th column of $\mathbf{U}$.

2.5. The Grassmann Manifold

Up to this point, we have focused on detection algorithms which explicitly view the data in the ambient space where the measurements were taken: $\mathbb{R}^n$ or $\mathbb{C}^n$. There has been some work which instead maps aggregate measurements to the Grassmann manifold and performs comparisons there [5–9].

[5] As we shall see later, in some contexts this is referred to as the chordal distance.
[6] See Section 2.5.2 for the definition of principal angles.

DEFINITION 2.5.1. *The* ***Grassmann manifold***[7] $Gr(m, \boldsymbol{\mathcal{V}})$ *is the manifold of points which parameterize* m*-dimensional linear subspaces of the vector space* $\boldsymbol{\mathcal{V}}$*. When* $\boldsymbol{\mathcal{V}} = \mathbb{R}^n$*, we denote this* $Gr(m, n)$ *and can view it as homeomorphic to* $O(n)/\left(O(m) \times O(n-m)\right)$ *or* $P_{m,n}$ *[27, Section 1.3.2].*

For example, suppose we wish to perform signal detection on the pixels in an image. Specifically, suppose we have far more spectral bands than the three provided by a traditional RGB color sensor. This is the case with hyperspectral imagery, where we use a spectrometer to collect data over, in some cases, hundreds of spectral bands. If the sensor samples n spectral bands, a single pixel may be thought of as a vector in $\mathbb{R}^n$. See Figure 2.5 for a schematic of this interpretation. If we look at a 2×2 pixel tile, we now have 4 such vectors. This corresponds to the first two steps shown schematically in Figure 2.6. In our schematics, the color coding should be read as identifying specific pixels, vectors, and points, not as indicating the value of the pixel.

Continuing with the next step in the schematic, we consider the linear subspace spanning these vectors. If our data contains broadband noise, it is reasonable to assume that the vectors are linearly independent, yielding a 4-dimensional subspace. This subspace is uniquely identified by a point on the Grassmann manifold $Gr(4, n)$. Of course, if our tile consisted of $k \times k$ pixels (where $k^2 < n$), our point would reside on $Gr(k^2, n)$ instead.

Generally this approach has been taken in the context of cluster-based classifiers rather than detectors based upon linear subspace models.

2.5.1. NOTATION. As already noted, there are isomorphisms between points on the Grassmann manifold $Gr(m, n)$, m-dimensional linear subspaces in an n-dimensional vector

[7]Many texts refer to a Grassmann manifold as a **Grassmannian**. We prefer the term Grassmann manifold to emphasize that it is a manifold.

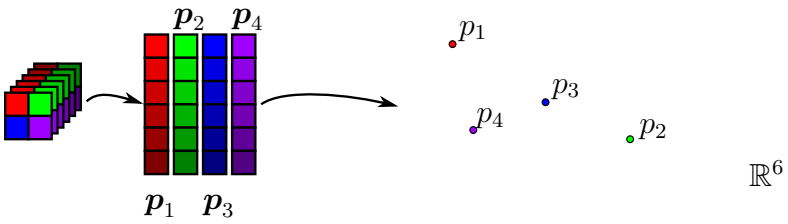


FIGURE 2.5. Mapping pixel data to Euclidean space.

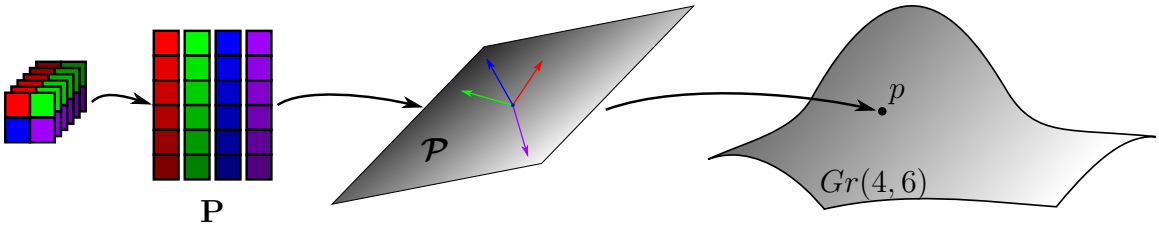


FIGURE 2.6. Mapping a tile of pixel data to a Grassmann manifold.

space $\boldsymbol{\mathcal{V}}$, and equivalence classes of full-rank $n \times m$ matrices. Because these isomorphisms are well-known, there is a temptation to treat these objects as interchangeable. In the next chapter, our discussion of subspaces, matrices, and points on the Grassmann manifold will become more involved, though. To facilitate this, we will lay out an explicit notation now. We have already touched upon some aspects of this notation.

First, we write an $n \times m$ matrix as the boldface capital $\mathbf{T}$. When viewed as a function on points in an m-dimensional vector space, we will write the range of $\mathbf{T}$ as $\mathcal{R}(\mathbf{T})$. This is the same as the span of the columns of $\mathbf{T}$, written $\langle \mathbf{T} \rangle$.

Suppose we have two matrices $\mathbf{T}^{(1)}$ and $\mathbf{T}^{(2)}$. Even though $\mathbf{T}^{(1)} \neq \mathbf{T}^{(2)}$, it is still possible that $\left\langle \mathbf{T}^{(1)} \right\rangle = \left\langle \mathbf{T}^{(2)} \right\rangle$. In other words, while there is not an isomorphism between the set of m-dimensional linear subspaces and rank m matrices, there is an isomorphism between m-dimensional linear subspaces and **equivalence classes** of full rank $n \times m$ matrices.

Specifically, these equivalence classes are defined by the relation

$$\mathbf{T}^{(1)} \sim \mathbf{T}^{(2)}$$

$$\iff \qquad \left\langle \mathbf{T}^{(1)} \right\rangle = \left\langle \mathbf{T}^{(2)} \right\rangle .$$

Given a representative $\mathbf{T}$, the isomorphism is defined by

$$[\mathbf{T}] \mapsto \langle \mathbf{T} \rangle$$

$$\boldsymbol{\mathcal{T}} \mapsto \left[\mathbf{T}^{(0)} \right]$$

where $[\mathbf{T}]$ is the class of matrices equivalent to $\mathbf{T}^{(0)}$, and $\mathbf{T}^{(0)}$ is constructed so that $\left\langle \mathbf{T}^{(0)} \right\rangle = \boldsymbol{\mathcal{T}}$.

As mentioned in Definition 2.5.1, points on $Gr(m, n)$ parameterize the set of m-dimensional linear subspaces of an n-dimensional vector space. We will write points on a Grassmann manifold in lower case, e.g., $t \in Gr(m, n)$. The isomorphism between z and $\boldsymbol{\mathcal{T}}$ is the one implicit in the definition of the Grassmann manifold.

Why make the distinction between t and $\boldsymbol{\mathcal{T}}$? First, the Grassmann manifold is a set of points, not a set of subspaces. While this distinction may not be important in some applications, it exists none the less. Second and more importantly for us, in Chapter 3 we will be comparing subspaces of differing dimension. While comparing subspaces makes perfect sense, comparing a point $t^{(1)} \in Gr(m_1, n)$ to one on $t^{(2)} \in Gr(m_2, n)$ does not.

2.5.2. Principal Angles and Vectors. How do we compare two subspaces? A standard answer to this is to compute the principal angles.

DEFINITION 2.5.2. *Given an inner product space* $\boldsymbol{\mathcal{V}}$ *and two subspaces* $\boldsymbol{\mathcal{A}}, \boldsymbol{\mathcal{B}} \in \boldsymbol{\mathcal{V}}$ *with* $r > \dim \boldsymbol{\mathcal{A}} \geq \dim \boldsymbol{\mathcal{B}}$*, the first* ***principal angle*** *is defined*

$$\theta_1(\boldsymbol{\mathcal{A}}, \boldsymbol{\mathcal{B}}) \doteq \min_{\substack{\boldsymbol{u} \in \boldsymbol{\mathcal{A}} \\ \boldsymbol{v} \in \boldsymbol{\mathcal{B}} \\ \|\boldsymbol{u}\| = \|\boldsymbol{v}\| = 1}} \arccos \boldsymbol{u}^{\mathsf{H}} \boldsymbol{v}.$$

Let $\boldsymbol{u}_1$ *and* $\boldsymbol{v}_1$ *be minimizers yielding* θ_1*. We refer to these as the* ***principal vectors*** *associated with the first principal angle.*

The principal angles $\theta_2, \ldots, \theta_r$ *are defined recursively as*

$$\theta_i(\boldsymbol{\mathcal{A}}, \boldsymbol{\mathcal{B}}) \doteq \min_{\substack{\boldsymbol{u} \in \boldsymbol{\mathcal{A}} \\ \boldsymbol{v} \in \boldsymbol{\mathcal{B}} \\ \|\boldsymbol{u}\| = \|\boldsymbol{v}\| = 1}} \arccos \boldsymbol{u}^{\mathsf{H}} \boldsymbol{v}$$

$$\text{subject to } \boldsymbol{u}^{\mathsf{H}} \boldsymbol{u}_j = 0, \boldsymbol{v}^{\mathsf{H}} \boldsymbol{v}_j = 0 \text{ for all } j = 1, \ldots, i-1.$$

As in the case with the first principal angle, $\boldsymbol{u}_i$ *and* $\boldsymbol{v}_j$*, the* i*th principal vectors, are minimizers yielding* θ_i*.*

We will write the function producing the principal angles

$$\boldsymbol{\theta}(\boldsymbol{\mathcal{A}}, \boldsymbol{\mathcal{B}}) \doteq [\theta_1(\boldsymbol{\mathcal{A}}, \boldsymbol{\mathcal{B}}), \ldots, \theta_r(\boldsymbol{\mathcal{A}}, \boldsymbol{\mathcal{B}})],$$

with $0 \leq \theta_{\min} = \theta_1 \leq \theta_2 \leq \cdots \leq \theta_r = \theta_{\max} \leq \pi/2$.

Björck and Golub [28] describe a method for computing the principal angles and principal vectors using the SVD. We use this approach in Algorithms 2.7 and 2.8.

2.6. SUMMARY

In this chapter, we have discussed linear subspace models. Based upon different assumptions about the linear model and our knowledge, we presented some classic matched and

```
function PrincipalAngles(A, B)
    (Q_A, R_A) ← QR(A)
    (Q_B, R_B) ← QR(B)
    (U, Σ, V) ← SVD(Q_A^H Q_B)
    return arccos DIAG(Σ)
end function
```

FIGURE 2.7. Algorithm for computing principal angles between the ranges of two matrices.

```
function PrincipalVectors(A, B)
    (Q_A, R_A) ← QR(A)
    (Q_B, R_B) ← QR(B)
    (U, Σ, V) ← SVD(Q_A^H Q_B)
    return (Q_A U, Q_B V)
end function
```

FIGURE 2.8. Algorithm for computing principal vectors given two matrices.

adaptive subspace detectors, as well as the RX anomaly detector. In most cases, the usefulness of linear subspace models relies upon our knowledge of the signal subspace. To identify the subspace, we presented some simple methods which have been employed to fit a subspace to data. Finally, looking forward to our contributions in Chapter 3, we discussed the Grassmann manifold and some ways in which it has been applied to classification. Next we will bridge the gap between linear subspace model-based detection and Grassmann manifold techniques.

CHAPTER 3

Geometric Tests

3.1. Introduction

Let us start with the premise that we have the ability and are motivated to map our data to a Grassmann manifold. As noted in Chapters 1 and 2, this is an approach which has been used in signal detection and classification [5–8].

Suppose we have two collections of data, e.g., two points on a Grassmann manifold. It is reasonable to study functions that compare these two points. A function might be a likelihood ratio, a probability, a distance, or any other "well-behaved" comparison. In Section 3.2, we describe and motivate what we mean by "well-behaved."

When we consider a linear model, as described in Chapter 2, we are comparing our data point to more than one other point. We are, in some sense, comparing it to all possible model parameter choices. In Section 3.3, we discuss the construction of a set of realizations for a given model. This novel approach results in a new set of theoretical results. We present the optimal function value and set of optimizers for the class of models which use only a single subspace. These lead directly to algorithms for signal detection and reconstruction.

3.2. Point-to-Point Functions

Consider a function f which, given two points on the same Grassmann manifold, produces a real value. That is

$$f : Gr(m, \boldsymbol{\mathcal{V}}) \times Gr(m, \boldsymbol{\mathcal{V}}) \longrightarrow \mathbb{R}.$$

Figure 3.1 provides a schematic representation. Two commonly used functions are the geodesic distance

$$d_{\text{geo}}(t,u) = \|\boldsymbol{\theta}(\boldsymbol{\mathcal{T}},\boldsymbol{\mathcal{U}})\|_2 \tag{3.1}$$

and the chordal distance [29, 5]

$$d_{\text{chord}}(t,u) = \|\sin\left(\boldsymbol{\theta}\left(\boldsymbol{\mathcal{T}},\boldsymbol{\mathcal{U}}\right)\right)\|_2 \tag{3.2}$$

where $\boldsymbol{\theta}\left(\boldsymbol{\mathcal{T}},\boldsymbol{\mathcal{U}}\right)$ is the vector of principal angles between the subspaces identified with points $t, u \in Gr(m, \boldsymbol{\mathcal{V}})$. A probability density function and likelihood ratio are also examples [27]. Absil, et al. [30] specifically mention the distribution of d_{geo} between two uniformly distributed random points.[1]

Notice that both the geodesic and chordal distances are functions of the principal angles between the two spaces. It has been shown that certain metrics on a Grassmann manifold can be written as functions of principal angles [31, 32]. Hence, we will consider functions f which can be stated as real-valued functions g on the vector of principal angles between the two subspaces

$$f(t,u) = g\left(\boldsymbol{\theta}\left(\boldsymbol{\mathcal{T}},\boldsymbol{\mathcal{U}}\right)\right)$$

where $t, u \in Gr(m, \boldsymbol{\mathcal{V}})$. See Figure 3.1 for a schematic view of this relationship.

In Section 3.3, we will want to find the minimum value of a point-to-point function between a fixed point and every point in a set. To facilitate this, it will be useful to consider functions f which preserve an order on the arguments. In other words, if we define our

[1] A general practical form of this distribution appears to still be an open problem.

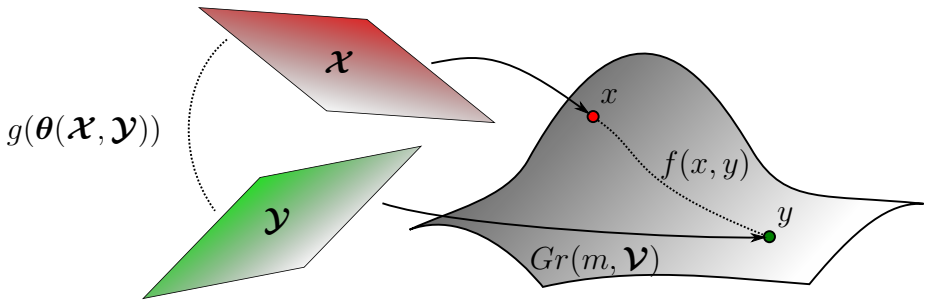


FIGURE 3.1. The distance between two points on a Grassmann manifold.

order to be some relation $\leq_{\boldsymbol{\mathcal{V}}}$ on vector space $\boldsymbol{\mathcal{V}}$, then we wish to consider functions where $\boldsymbol{u} \leq_{\boldsymbol{\mathcal{V}}} \boldsymbol{v} \implies f(\boldsymbol{u}) \leq f(\boldsymbol{v})$ for all $\boldsymbol{u}, \boldsymbol{v} \in \boldsymbol{\mathcal{V}}$.

There are three preorders that are often applied to vectors. First, we can simply consider the product order, which we will simply write $\leq$.

DEFINITION 3.2.1. *Let $\boldsymbol{u}$ and $\boldsymbol{v}$ be vectors from the same n-dimensional vector space with fixed basis. $\boldsymbol{u} \leq \boldsymbol{v}$ if and only if $u_i \leq v_i$ for all $i = 1, \ldots, n$. This relation is refered to as the* ***product order****.*

DEFINITION 3.2.2. *If the function f preserves the product order, then we say that the function is* ***monotone increasing****. If $f(\boldsymbol{u}) = f(\boldsymbol{v})$ only when $\boldsymbol{u} = \boldsymbol{v}$, then we say that the function is* ***strictly increasing****.*

We can also consider the majorization preorder.

DEFINITION 3.2.3. *Let $\boldsymbol{u}$ and $\boldsymbol{v}$ be vectors from the same n-dimensional vector space with fixed basis and $\{u_{[i]}\}$ be a permutation of the elements of $\boldsymbol{u}$ such that $u_{[1]} \geq \cdots \geq u_{[n]}$ and*

similarly for $\boldsymbol{v}$*. We say that* $\boldsymbol{v}$ ***majorizes*** $\boldsymbol{u}$ *(or* $\boldsymbol{u}$ ***is majorized by*** $\boldsymbol{v}$*) if and only if*

$$\sum_{i=1}^{r} u_{[i]} \leq \sum_{i=1}^{r} v_{[i]} \qquad \text{for } r = 1, \ldots, n-1,$$
$$\sum_{i=1}^{n} u_{[i]} = \sum_{i=1}^{n} v_{[i]}.$$

If $\boldsymbol{v}$ *majorizes* $\boldsymbol{u}$*, we write* $\boldsymbol{u} \prec \boldsymbol{v}$*.*

DEFINITION 3.2.4. *When a function* f *preserves the majorization preorder, we say that the function is* ***Schur-convex****.*

Majorization only provides information on vectors whose components produce the same sum. When considering vectors of principal angles, a function being Schur-convex only provides us with ordering when the principal angles sum to the same value. If we remove this restriction, we arrive at the following definitions.

DEFINITION 3.2.5. *Let* $\boldsymbol{u}$ *and* $\boldsymbol{v}$ *be vectors from the same* n*-dimensional vector space with fixed basis. We say that* $\boldsymbol{v}$ ***weakly majorizes*** $\boldsymbol{u}$ *(or* $\boldsymbol{u}$ ***is weakly majorized by*** $\boldsymbol{v}$*) if and only if*

$$\sum_{i=1}^{r} u_{[i]} \leq \sum_{i=1}^{r} v_{[i]} \qquad r = 1, \ldots, n.$$

If $\boldsymbol{v}$ *weakly majorizes* $\boldsymbol{u}$*, we write* $\boldsymbol{u} \prec_w \boldsymbol{v}$*.*

A function f which preserves the weak majorization preorder is both Schur-convex and monotone increasing [33, Section A3.1.2]. Most of our results apply to monotone increasing functions on principal angles, so they also apply those known to preserve weak majorization.

3.3. Point-to-Set Functions

Up to this point, we have only considered functions for comparing two points on a Grassmann manifold. This may be viewed as analogous to the case in Chapter 2 of comparing a data point with a single realization of a model. The practical detectors discussed in Section 2.3 use the maximum likelihood estimate for each hypothesis, effectively considering the likelihood function for all realizations of the model parameters.

In this section, we propose the use of a set of points on the Grassmann manifold. Specifically, we provide a mapping of a broad family of linear subspace models to the set of loci known as Schubert varieties. Using such a set and a point-to-point function as discussed in Section 3.2, we will produce a point-to-set function which behaves as a geometric analog of the MLE. As with the GLRT, a signal detection algorithm can be developed from such a point-to-set function.

To build the necessary base of concepts, we will use simple examples to motivate general results. In some cases, we will use vague terms such as "nearby." If necessary, these will be clarified when we formally develop the associated general result.

Consider the simple linear model for the ith sample

$$\boldsymbol{x}_i = \boldsymbol{s}\psi_i + \boldsymbol{\nu}_i,$$

where $\boldsymbol{x}_i, \boldsymbol{s}, \boldsymbol{\nu}_i \in \mathbb{R}^n$ and $\psi_i \in \mathbb{R}$. Suppose we consider the span of two data samples, $\boldsymbol{x}_j$ and $\boldsymbol{x}_{j+1}$, and map that to the associated point on a Grassmann manifold. Then we will be mapping our data to $p \in Gr(2,n)$[2]. While $\boldsymbol{s}$ is known, the broadband noise terms $\boldsymbol{\nu}_j$ and $\boldsymbol{\nu}_{j+1}$ may result in our point lying anywhere on $Gr(2,n)$. Assuming a reasonably high SNR,

[2]This statement is true with probability 1. There is a probability 0 set of events where we end up with a 0- or 1-dimensional subspace.

though, our point p will be near a point $t \in Gr(2,n)$ whose associated subspace $\boldsymbol{\mathcal{T}}$ actually contains $\langle \boldsymbol{s} \rangle$ with high probability. So let us consider the set of points on a Grassmann manifold which contain $\langle \boldsymbol{s} \rangle$. In other words, we can think of these as points where the noise is impossibly[3] well-behaved–$\boldsymbol{\nu}_j$ and $\boldsymbol{\nu}_{j+1}$ lie in the same 1-dimensional linear subspace. Assuming we have signal ($\psi_j, \psi_{j+1} \neq 0$), this is the best case we can hope for and actually produce linearly independent samples.

What does the set of points on $Gr(2,n)$ produced by these well-behaved realizations of our model look like? When $n = 3$ it looks like the schematic in Figure 3.2(a). Since it is all of the 2-dimensional subspaces containing $\langle \boldsymbol{s} \rangle$, it is simply the set of configurations formed by rotating a plane about $\boldsymbol{s}$.

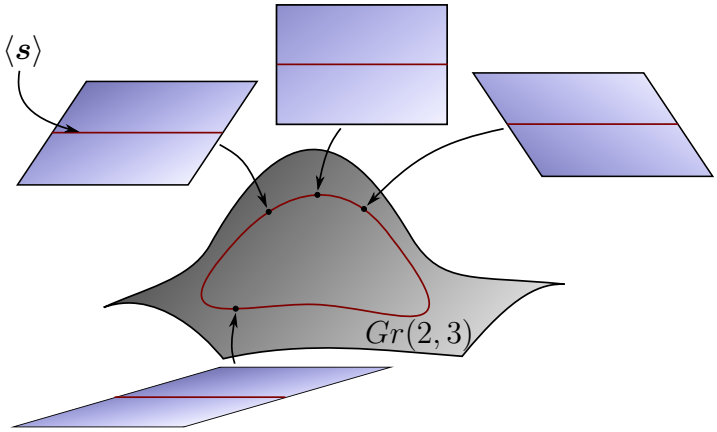
Now let us look at another linear model,

$$\boldsymbol{x}_i = \mathbf{S}\boldsymbol{\psi} + \boldsymbol{\nu}_i, \tag{3.3}$$

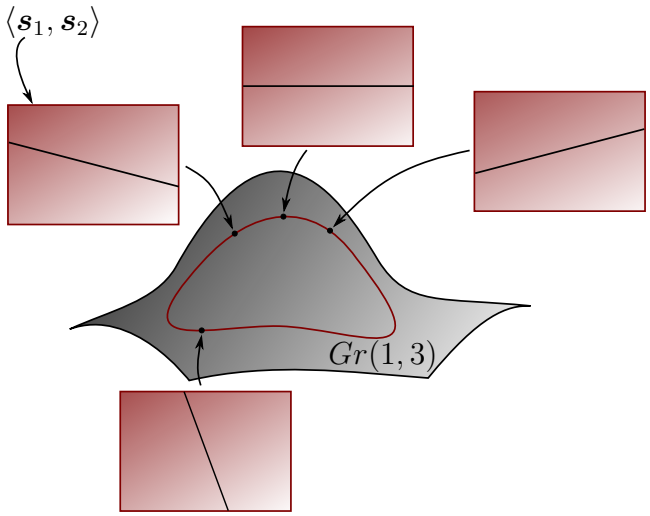
where $\boldsymbol{x}_i, \boldsymbol{\nu}_i \in \mathbb{R}^n$, $\mathbf{S} \in \mathbb{R}^{n \times 2}$, and $\boldsymbol{\psi}_i \in \mathbb{R}^2$. This time, instead of combining two samples, we will only use one and map it to $Gr(1,n)$. As before, if the SNR is reasonably high, we may now expect to be near points which correspond to a 1-dimensional linear subspace contained in $\langle \mathbf{S} \rangle$. We can think of these as points which have no noise that can be distinguished from the signal. When $n = 3$ this set looks like the schematic in Figure 3.2(b).

The sets of these "best case" points that we expect data to cluster around are examples of sets called Schubert varieties. There are several equivalent definitions of a Schubert variety. We use the following due to its clear connection with some linear models of interest.

[3]While this is a useful way of thinking about this set, such an event has probability 0 for all reasonable distributions on $\boldsymbol{\nu}_i$.



(a) $\Omega_{\langle \boldsymbol{s} \rangle,1}$ in $Gr(2,3)$.



(b) $\Omega_{\langle \boldsymbol{s}_1, \boldsymbol{s}_2 \rangle,1}$ in $Gr(1,3)$.

FIGURE 3.2. Examples of Schubert varieties. The red linear subspace is the subspace $\boldsymbol{\mathcal{S}}$ and the black linear subspace $\boldsymbol{\mathcal{T}}$ is the subspace which corresponds to the point $t \in \Omega_{\boldsymbol{\mathcal{S}},a} \subset Gr(m,n)$.

DEFINITION 3.3.1. *Given an n-dimensional vector space $\boldsymbol{\mathcal{V}}$, a flag*[4] $F = \{\mathbf{0}\} \subset \boldsymbol{\mathcal{U}}_1 \subset \cdots \subset \boldsymbol{\mathcal{U}}_k \subset \boldsymbol{\mathcal{V}}$*, and vector of associated integers $\boldsymbol{a} = [a_1, \ldots, a_k]$ with $0 < a_1 < a_2 < \cdots < a_k \leq m$, its **Schubert variety** on $Gr(m, \boldsymbol{\mathcal{V}})$ is*

$$\Omega_{F,\boldsymbol{a}} \doteq \{t \mid t \in Gr(m, \boldsymbol{\mathcal{V}}), \dim(\boldsymbol{\mathcal{T}} \cap \boldsymbol{\mathcal{U}}_i) \geq a_i, \forall i\}.$$

[4]A **flag** is an increasing sequence of linear subspaces.

Let us return to the examples shown in Figure 3.2. As noted previously, while the ambient space is the same, in (a) the realizations contain both our signature and some noise and in (b) the realizations are exactly subspaces of our signature subspace. These simple examples are restricted by the low ambient dimension (3), but with higher ambient dimension comes the opportunity for these behaviors and more to exist in a single Schubert variety. This combination of simple and clear statements with complex phenomena mark this as an interesting formalism to explore.

To the best of our knowledge, associating a Schubert variety with a linear signal model is a new idea. While there may be interesting results for the general case, we will begin with a restricted case; for the remainder of this treatment, let us restrict ourselves to relatively simple Schubert varieties which have a clear connection to existing linear subspace models. Specifically, consider the case where the flag is comprised of only a single signature subspace, $\boldsymbol{\mathcal{S}} \in \boldsymbol{\mathcal{V}}$.[5] While we will prove results for $\Omega_{\boldsymbol{\mathcal{S}},a}$, it may be useful to keep the special case where $a = 1$ in mind. The model associated with $\Omega_{\boldsymbol{\mathcal{S}},1}$ is the one used by MSD when there is no clutter term (see Section 2.3.1 and Equation (3.3)).

3.3.1. Detection Theory. To develop a signal detection algorithm, we need to find a closed form for the pointwise minimum value of g between $\Omega_{\boldsymbol{\mathcal{S}},a}$ and our aggregate data point $p \in Gr(m, \boldsymbol{\mathcal{V}})$. This is the geometric analog of the maximum likelihood estimate. We will denote this with the function $g^\star$,

$$g^\star(\Omega_{\boldsymbol{\mathcal{S}},a}, p) = \min_{t \in \Omega_{\boldsymbol{\mathcal{S}},a}} g(\boldsymbol{\theta}\left(\boldsymbol{\mathcal{T}}, \boldsymbol{\mathcal{P}}\right). \tag{3.4}$$

[5]As we do frequently throughout this dissertation, the vector space $\boldsymbol{\mathcal{V}}$ may be read as the ambient space in which me make our measurements. This will often be $\mathbb{R}^n$, $\mathbb{C}^n$, or some subset of one of these. The primary requirement is simply that $\boldsymbol{\mathcal{V}}$ be an inner product space.

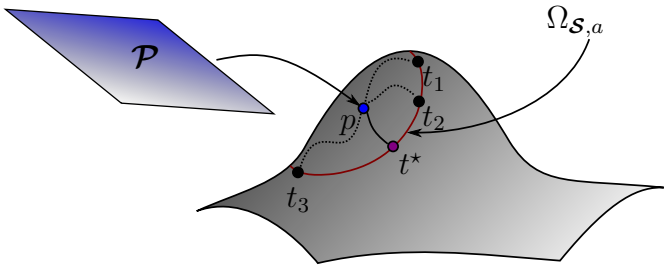


FIGURE 3.3. The distance between a point and a set (e.g., a Schubert variety) on a Grassmann manifold.

This is shown schematically in Figure 3.3. As in Figure 3.2, the red line represents the Schubert variety corresponding to our model, the black dots t_1, t_2, t_3 represent points in the Schubert variety. The purple dot $t^\star$ is the point on the Schubert variety which is (according to g) closest to the blue dot (p), the aggregate data point under consideration.

Before we find the solution for the general case, let us informally consider the $\Omega_{\boldsymbol{\mathcal{S}},1}$ case and use that as a guide for $\Omega_{\boldsymbol{\mathcal{S}},a}$. Assume that we are given a monotone increasing function g on the principal angles between two points on $Gr(m, \boldsymbol{\mathcal{V}})$. We would like to find a closed form for $g^\star(\Omega_{\boldsymbol{\mathcal{S}},1}, p)$. For example, if g is d_{geo}, then this amounts to deriving a function which directly computes the smallest geodesic distance between our data, p, and our model, $\Omega_{\boldsymbol{\mathcal{S}},1}$.

As usual, let our ambient vector space be denoted $\boldsymbol{\mathcal{V}}$. Points in $\Omega_{\boldsymbol{\mathcal{S}},1}$ are associated with subspaces of the form $\langle \boldsymbol{v}_1, \ldots, \boldsymbol{v}_{m-1}, \boldsymbol{s} \rangle$ where the vectors $\boldsymbol{s} \in \boldsymbol{\mathcal{S}} \subset \boldsymbol{\mathcal{V}}$, $\boldsymbol{v}_i \in \boldsymbol{\mathcal{V}}$, and $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_{m-1}, \boldsymbol{s}$ are linearly independent. We can select $m-1$ mutually orthogonal vectors $\boldsymbol{p}_1, \ldots, \boldsymbol{p}_{m-1} \in \boldsymbol{\mathcal{P}}$ that are also orthogonal to $\boldsymbol{s}$. This is clear if we consider constructing an orthonormal basis with Gram-Schmidt on $[\boldsymbol{s}, \mathbf{P}]$[6] where $\mathbf{P}$ is an $n \times m$ matrix and $\langle \mathbf{P} \rangle = \boldsymbol{\mathcal{P}}$.

[6]When we write a sequence of matrices and vectors in brackets (e.g., $[\boldsymbol{s}, \mathbf{P}]$), we mean the matrix formed by concatenating these column vectors and matrices to form a matrix. In this example, if $\boldsymbol{s} \in \mathbb{R}^n$ and $\mathbf{P} \in \mathbb{R}^{n \times m}$, the resulting matrix has dimensions $n \times (m+1)$.

The principal angles between $\boldsymbol{\mathcal{P}}$ and this space are

$$\boldsymbol{\theta}\left(\left\langle \boldsymbol{p}_1, \ldots, \boldsymbol{p}_{m-1}, \boldsymbol{s}\right\rangle, \boldsymbol{\mathcal{P}}\right) = \left[0, \ldots, 0, \theta\left(\langle \boldsymbol{s}\rangle, \boldsymbol{\mathcal{P}}\right)\right],$$

since the first $m-1$ principal directions will be in $\boldsymbol{\mathcal{P}}$ (and specifically, in $\left\langle \boldsymbol{p}_1, \ldots, \boldsymbol{p}_{m-1}\right\rangle$ and the associated principal angles will be 0. The final principal direction needs to be orthogonal to the first $m-1$ principal directions ($\left\langle \boldsymbol{p}_1, \ldots, \boldsymbol{p}_{m-1}\right\rangle$), so it must lie in $\langle \boldsymbol{s}\rangle$. Hence, the m-th principal angle is simply the principal angle between $\boldsymbol{\mathcal{P}}$ and $\langle \boldsymbol{s}\rangle$.

Let us select $\boldsymbol{s}^\star$ to be a principal vector in $\boldsymbol{\mathcal{S}}$ associated with $\theta_{\min}(\boldsymbol{\mathcal{S}}, \boldsymbol{\mathcal{P}})$ and $\{\boldsymbol{p^\star}_i\}_{i=1}^{m-1}$ to be the associated orthogonal basis in $\langle \boldsymbol{s}^\star\rangle^\perp \cap \boldsymbol{\mathcal{P}}$.

$$\boldsymbol{\theta}\left(\left\langle \boldsymbol{p^\star}_1, \ldots, \boldsymbol{p^\star}_{m-1}, \boldsymbol{s}^\star\right\rangle, \boldsymbol{\mathcal{P}}\right) = \left[0, \ldots, 0, \theta_{\min}\left(\boldsymbol{\mathcal{S}}, \boldsymbol{\mathcal{P}}\right)\right].$$

From the definition of the minimum principal angle, for any other $\tilde{t} \in \Omega_{\boldsymbol{\mathcal{S}},1}$,

$$\left[0, \ldots, 0, \theta_{\min}\left(\boldsymbol{\mathcal{S}}, \boldsymbol{\mathcal{P}}\right)\right] \leq \boldsymbol{\theta}\left(\tilde{\boldsymbol{\mathcal{T}}}, \boldsymbol{\mathcal{P}}\right).$$

Since g preserves the partial ordering $\leq$,

$$g\left(\left[0, \ldots, 0, \theta_{\min}\left(\boldsymbol{\mathcal{S}}, \boldsymbol{\mathcal{P}}\right)\right]\right) \leq g\left(\boldsymbol{\theta}\left(\tilde{\boldsymbol{\mathcal{T}}}, \boldsymbol{\mathcal{P}}\right)\right).$$

Hence, $\left\langle \boldsymbol{p^\star}_1, \ldots, \boldsymbol{p^\star}_{m-1}, \boldsymbol{s}^\star\right\rangle$ is a (potentially nonunique) point in $\Omega_{\boldsymbol{\mathcal{S}},1}$ which minimizes $g\left(\boldsymbol{\theta}\left(\cdot, \boldsymbol{\mathcal{P}}\right)\right)$ restricted to $\Omega_{\boldsymbol{\mathcal{S}},1}$. Therefore

$$g^\star(\Omega_{\boldsymbol{\mathcal{S}},1}, p) = g\left(\left[0, \ldots, 0, \theta_{\min}\left(\boldsymbol{\mathcal{S}}, \boldsymbol{\mathcal{P}}\right)\right]\right).$$

It seems reasonable that we can find a similar result for general a by following the template suggested by $a = 1$. The primary difficulty we face occurs where we previously stated that $[0, \ldots, 0, \theta_{\min}(\boldsymbol{\mathcal{S}}, \boldsymbol{\mathcal{P}})] \leq \theta\left(\tilde{\boldsymbol{\mathcal{T}}}, \boldsymbol{\mathcal{P}}\right)$ for any $\tilde{t} \in \Omega_{\boldsymbol{\mathcal{S}},1}$. To extend this argument, we must first prove that there actually is a unique least vector of principal angles between p and points in $\Omega_{\boldsymbol{\mathcal{S}},a}$.

LEMMA 3.3.2. *Given a Schubert variety $\Omega_{\boldsymbol{\mathcal{S}},a} \subseteq Gr(m, \boldsymbol{\mathcal{V}})$ and a point $p \in Gr(m, \boldsymbol{\mathcal{V}})$, for every point $x \in \Omega_{\boldsymbol{\mathcal{S}},a}$*

$$[\mathbf{0}_{m-a}, \boldsymbol{\theta}_{1,\ldots,a}(\boldsymbol{\mathcal{S}}, \boldsymbol{\mathcal{P}})] \leq \boldsymbol{\theta}(\boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{P}}).$$

The proof of this lemma relies on a two-stage approach, shown schematically in Figure 3.4. First we prove in step (I) that, for certain subsets U_x of $\Omega_{\boldsymbol{\mathcal{S}},a}$, there exists a unique least vector of principal angles. This vector of principal angles is realized at point $x^\star \in U_x$. These subsets form a cover of $\Omega_{\boldsymbol{\mathcal{S}},a}$. In (II), we compare our candidate least vector of principal angles for the whole variety and realized at $y \in \Omega_{\boldsymbol{\mathcal{S}},a}$ with the unique least vector for each set in the cover we constructed in (I).

PROOF. We are given $\boldsymbol{\mathcal{S}} \subseteq \boldsymbol{\mathcal{V}}$ ($\dim \boldsymbol{\mathcal{S}} = r$) and $a \in \mathbb{Z}$ ($0 < a \leq \min\{m, r\}$). Let $\mathbf{S}$ be an $n \times r$ matrix with orthogonal unit columns and $\boldsymbol{\mathcal{S}} = \langle \mathbf{S} \rangle$. Fix a point $p \in Gr(m, \boldsymbol{\mathcal{V}})$. Define an $n \times m$ matrix $\mathbf{P}$ with orthogonal unit columns such that p refers to the subspace $\langle \mathbf{P} \rangle$.

Step (I). Let us consider an arbitrary point $x \in \Omega_{\boldsymbol{\mathcal{S}},a}$. Let $k \in \{0, \ldots, m-a\}$. We can write this as a span of the columns of an $n \times m$ matrix $[\mathbf{X}_{1,\ldots,k}, \mathbf{S}\mathbf{Q}_{m-k}]$ where $\mathbf{Q}_{m-k}$ is an $r \times (m-k)$ matrix with orthonormal columns. We can also write this same matrix as $\mathbf{M}\mathbf{Q}_m$ where $\mathbf{M} = [\mathbf{X}_{1,\ldots,k}, \mathbf{S}]$ and $\mathbf{Q}_m$ is an $r \times m$ matrix with orthonormal columns. As described

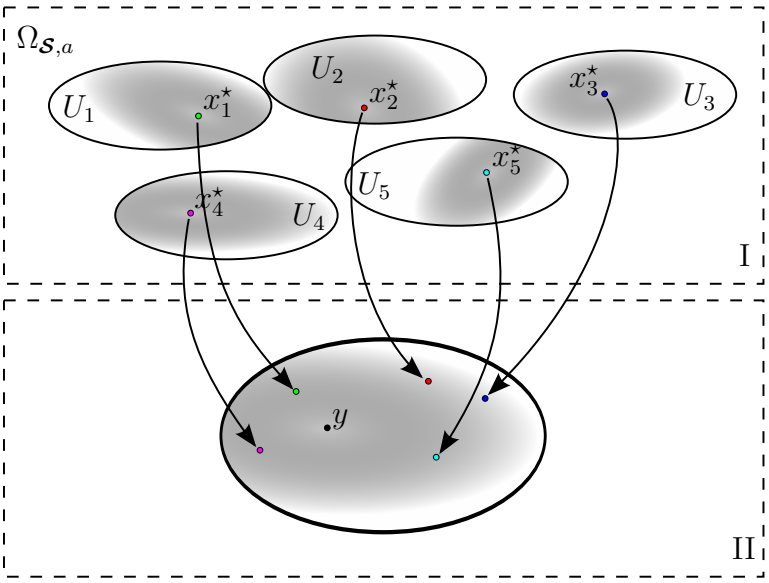


FIGURE 3.4. A schematic representation of the proof of Lemma 3.3.2.

by [28], we can calculate the principal angles

$$\begin{aligned}\theta_i(p,x) &= \arccos\sigma_i(\mathbf{P}^\mathsf{T}\mathbf{X}) \\ &= \arccos\sigma_i(\mathbf{P}^\mathsf{T}\mathbf{M}\mathbf{Q}_m),\end{aligned}$$

where $\sigma_i(\mathbf{A})$ is the i-th largest singular value of $\mathbf{A}$. By [34, Lemma 3.3.1],

$$\begin{aligned}&&\sigma_i(\mathbf{P}^\mathsf{T}\mathbf{M}\mathbf{Q}_m) &\leq \sigma_i(\mathbf{P}^\mathsf{T}\mathbf{M}) \\ \implies &&\arccos\sigma_i(\mathbf{P}^\mathsf{T}\mathbf{M}\mathbf{Q}_m) &\geq \arccos\sigma_i(\mathbf{P}^\mathsf{T}\mathbf{M}) \\ &&\theta_i(p,x) &\geq \theta_i(p,\boldsymbol{\mathcal{S}}).\end{aligned}$$

In other words, if we look at m-dimensional linear subspaces of $\boldsymbol{\mathcal{S}} + \langle \mathbf{X}_{1,\ldots,k} \rangle$, the one produced by the first m principal vectors has component-wise smaller (or equal) principal angles relative to $\boldsymbol{\mathcal{P}}$ than any other. Call this point $x^\star \in Gr(m, \boldsymbol{\mathcal{V}})$.

Step (II). Now let us consider the point $y = \langle \mathbf{P}_{a+1,\ldots,m}, \mathbf{S}_{1,\ldots,a} \rangle \in \Omega_{\boldsymbol{\mathcal{S}},a}$, where $\mathbf{P}$ and $\mathbf{S}$ are matrices whose columns are unit vectors in the directions of the principal vectors in p and $\boldsymbol{\mathcal{S}}$, respectively. Compare the principal angles produced by this to those produced for the arbitrary point $x^\star$ described above. Since y was constructed by choosing at least $m-a$ basis vectors from $\boldsymbol{\mathcal{P}}$, $\boldsymbol{\theta}_{1,\ldots,m-a}(\boldsymbol{\mathcal{Y}}, \boldsymbol{\mathcal{P}}) = \mathbf{0}_{m-a}$, so we only need to consider $\boldsymbol{\theta}_{m-a+1,\ldots,m}(\boldsymbol{\mathcal{Y}}, \boldsymbol{\mathcal{P}}) = \boldsymbol{\theta}_{1,\ldots,a}(\boldsymbol{\mathcal{S}}, \boldsymbol{\mathcal{P}})$. This is equivalent to $\arccos \boldsymbol{\sigma}_{1,\ldots,a}\left(\mathbf{P}^\mathsf{T}\mathbf{S}\right)$. Notice that the matrix $\mathbf{P}^\mathsf{T}\mathbf{S}$ is simply $\mathbf{P}^\mathsf{T}\mathbf{M}$ with the first k columns removed. Using [34, Corollary 3.1.3], for $i = 1, \ldots, a$,

$$\begin{aligned} & \sigma_{i+m-a}(\mathbf{P}^\mathsf{T}\mathbf{Y}) = \sigma_i(\mathbf{P}^\mathsf{T}\mathbf{S}) \geq \sigma_{i+k}(\mathbf{P}^\mathsf{T}\mathbf{M}) \geq \sigma_{i+m-a}(\mathbf{P}^\mathsf{T}\mathbf{M}) \\ \implies \quad & \arccos \sigma_{i+m-a}(\mathbf{P}^\mathsf{T}\mathbf{Y}) \leq \arccos \sigma_{i+m-a}(\mathbf{P}^\mathsf{T}\mathbf{M}) \\ & \theta_{i+m-a}(p, y) \leq \theta_{i+m-a}(p, x^\star). \end{aligned}$$

Since $\theta_{1,\ldots,m-a} = \mathbf{0}_{m-a}$, we also know that $\theta_{1,\ldots,m-a}(p, y) \leq \theta_{1,\ldots,m-a}(p, x^\star)$. Hence, for all $x \in \Omega_{\boldsymbol{\mathcal{S}},a}$,

$$[\mathbf{0}_{m-a}, \boldsymbol{\theta}_{1,\ldots,a}(\boldsymbol{\mathcal{S}}, \boldsymbol{\mathcal{P}})] \leq \boldsymbol{\theta}(\boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{P}}). \qquad \square$$

THEOREM 3.3.3. *Given an increasing, real-valued function g on ordered m-vectors of principal angles and the Schubert variety $\Omega_{\boldsymbol{\mathcal{S}},a}$,*

$$g^\star(\Omega_{\boldsymbol{\mathcal{S}},a}, p) = g\left([\mathbf{0}_{m-a}, \boldsymbol{\theta}_{1,\ldots,a}(\boldsymbol{\mathcal{S}}, \boldsymbol{\mathcal{P}})]\right).$$

PROOF. Define $g^\star$ to be the minimum value of g on $p \in Gr(m, \boldsymbol{\mathcal{V}})$ and any point in $\Omega_{\boldsymbol{\mathcal{S}},a}$.

$$g^\star(\Omega_{\boldsymbol{\mathcal{S}},a}, p) = \min_{t \in \Omega_{\boldsymbol{\mathcal{S}},a}} g\left(\boldsymbol{\theta}(\boldsymbol{\mathcal{T}}, \boldsymbol{\mathcal{P}})\right)$$

Points in $\Omega_{\boldsymbol{\mathcal{S}},a}$ are associated with subspaces of the form $\langle \boldsymbol{v}_1, \ldots, \boldsymbol{v}_{m-a}, \boldsymbol{s}_1, \ldots, \boldsymbol{s}_a \rangle$ where $\boldsymbol{s}_1, \ldots, \boldsymbol{s}_a \in \boldsymbol{\mathcal{S}}$ and $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_{m-a}, \boldsymbol{s}_1, \ldots, \boldsymbol{s}_a$ are linearly independent. We can select $m-a$ mutually orthogonal vectors $\boldsymbol{p}_1, \ldots, \boldsymbol{p}_{m-a} \in \boldsymbol{\mathcal{P}}$ that are also orthogonal to $\langle \boldsymbol{s}_1, \ldots, \boldsymbol{s}_a \rangle$, since[7]

$$\langle \boldsymbol{s}_1, \ldots, \boldsymbol{s}_a \rangle^{\perp} \cap \boldsymbol{\mathcal{P}} \supseteq \left(\langle \boldsymbol{s}_1, \ldots, \boldsymbol{s}_a \rangle + \boldsymbol{\mathcal{P}}^{\perp} \right)^{\perp}$$

$$\implies \qquad \dim \left(\langle \boldsymbol{s}_1, \ldots, \boldsymbol{s}_a \rangle^{\perp} \cap \boldsymbol{\mathcal{P}} \right) \geq \dim \left(\langle \boldsymbol{s}_1, \ldots, \boldsymbol{s}_a \rangle + \boldsymbol{\mathcal{P}}^{\perp} \right)^{\perp} \geq m - a.$$

The principal angles between $\boldsymbol{\mathcal{P}}$ and this space are

$$\boldsymbol{\theta} \left(\left\langle \boldsymbol{p}_1, \ldots, \boldsymbol{p}_{m-a}, \boldsymbol{s}_1, \ldots, \boldsymbol{s}_a \right\rangle, \boldsymbol{\mathcal{P}} \right) = \left[0, \ldots, 0, \theta \left(\boldsymbol{\mathcal{P}}, \langle \boldsymbol{s}_1, \ldots, \boldsymbol{s}_a \rangle \right) \right],$$

since the first $m-a$ principal vectors will be in $\boldsymbol{\mathcal{P}}$ (specifically, in $\left\langle \boldsymbol{p}_1, \ldots, \boldsymbol{p}_{m-a} \right\rangle$) and the associated principal angles will be 0. The final a principal vectors must be orthogonal to the first $m-a$ principal vectors ($\left\langle \boldsymbol{p}_1, \ldots, \boldsymbol{p}_{m-a} \right\rangle$), so they lie in $\langle \boldsymbol{s}_1, \ldots, \boldsymbol{s}_a \rangle$. Hence, the a largest principal angles are simply the principal angles between $\boldsymbol{\mathcal{P}}$ and $\langle \boldsymbol{s}_1, \ldots, \boldsymbol{s}_a \rangle$.

Let us select ${\boldsymbol{s}^{\star}}_1, \ldots, {\boldsymbol{s}^{\star}}_a$ to be the principal vectors in $\boldsymbol{\mathcal{S}}$ associated with $\boldsymbol{\theta}_{1,\ldots,a} (\boldsymbol{\mathcal{S}}, \boldsymbol{\mathcal{P}})$ and $\{{\boldsymbol{p}^{\star}}_i\}$ to be the associated orthogonal basis in $\langle {\boldsymbol{s}^{\star}}_1, \ldots, {\boldsymbol{s}^{\star}}_a \rangle^{\perp} \cap \boldsymbol{\mathcal{P}}$.

$$\boldsymbol{\theta} \left(\left\langle {\boldsymbol{p}^{\star}}_1, \ldots, {\boldsymbol{p}^{\star}}_{m-1}, \boldsymbol{s}^{\star} \right\rangle, \boldsymbol{\mathcal{P}} \right) = \left[0, \ldots, 0, \theta_{\min} (\boldsymbol{\mathcal{S}}, \boldsymbol{\mathcal{P}}) \right].$$

From Lemma 3.3.2, for any other $\tilde{t} \in \Omega_{\boldsymbol{\mathcal{S}},a}$,

$$[0, \ldots, 0, \boldsymbol{\theta}_{1,\ldots,a} (\boldsymbol{\mathcal{S}}, \boldsymbol{\mathcal{P}})] \leq \boldsymbol{\theta} \left(\tilde{\boldsymbol{\mathcal{T}}}, \boldsymbol{\mathcal{P}} \right).$$

[7] Proving that $(\boldsymbol{\mathcal{U}} + \boldsymbol{\mathcal{T}}^{\perp})^{\perp} \subseteq \boldsymbol{\mathcal{U}} \cap \boldsymbol{\mathcal{T}}$ is simply a matter of demonstrating that any arbitrary point in $(\boldsymbol{\mathcal{U}} + \boldsymbol{\mathcal{T}}^{\perp})^{\perp}$ is in both $\boldsymbol{\mathcal{U}}$ and $\boldsymbol{\mathcal{P}}$. The inequality follows from noting that

$$\dim(\langle \boldsymbol{s}_1, \ldots, \boldsymbol{s}_a \rangle + \boldsymbol{\mathcal{P}}^{\perp}) \leq a + (n - m).$$

The details do not provide any deeper understanding of the matter at hand, though.

Since g preserves the partial ordering $\leq$,

$$g\left([0,\ldots,0,\boldsymbol{\theta}_{1,\ldots,a}\left(\boldsymbol{\mathcal{S}},\boldsymbol{\mathcal{P}}\right)]\right) \leq g\left(\boldsymbol{\theta}\left(\tilde{\boldsymbol{\mathcal{T}}},\boldsymbol{\mathcal{P}}\right)\right).$$

Hence, $\left\langle \boldsymbol{p}^{\star}{}_1,\ldots,\boldsymbol{p}^{\star}{}_{m-a},\boldsymbol{s}^{\star}{}_1,\ldots,\boldsymbol{s}^{\star}{}_a\right\rangle$ is a (potentially nonunique) point in $\Omega_{\boldsymbol{\mathcal{S}},a}$ which minimizes $g(\boldsymbol{\theta}(\boldsymbol{\mathcal{P}},\cdot))$ and

$$g^{\star}(\Omega_{\boldsymbol{\mathcal{S}},a},p) = g\left([0,\ldots,0,\boldsymbol{\theta}_{1,\ldots,a}\left(\boldsymbol{\mathcal{S}},\boldsymbol{\mathcal{P}}\right)]\right). \qquad \square$$

All of the results so far had been on increasing functions g. In other words, metric- or distance-like functions. Functions related to probability, statistics, or similarity will generally be decreasing functions. We can produce analogous results for $\tilde{g}$ decreasing and $\tilde{g}^{\star}$ the maximum point-to-set value, by considering $g = -\tilde{g}$ and $\tilde{g}^{\star} = -g^{\star}$.

Returning to the clutter-free MSD model ($\Omega_{\boldsymbol{\mathcal{S}},1}$), there are a few interesting results. Combined, Corollaries 3.3.4 and 3.3.5 suggest a geometric argument behind the clutter-free MSD detector and provide an alternate interpretation of the UMP property.

COROLLARY 3.3.4. *The minimum geodesic distance between a point $p \in Gr(m,\boldsymbol{\mathcal{V}})$ and the Schubert variety $\Omega_{\boldsymbol{\mathcal{S}},1}$ is $\theta_{\min}(\boldsymbol{\mathcal{P}},\boldsymbol{\mathcal{S}})$.*

PROOF. The geodesic distance between two points $t,u \in Gr(m,\boldsymbol{\mathcal{V}})$ is defined as

$$d_{\text{geo}}(t,u) = \|\boldsymbol{\theta}\left(\boldsymbol{\mathcal{T}},\boldsymbol{\mathcal{U}}\right)\|_2.$$

This is an increasing function on ordered vectors of principal angles ($0 \leq \theta_i \leq \pi/2$ for $i = 1, \ldots, m$), hence

$$d^{\star}_{\text{geo}}(\Omega_{\boldsymbol{\mathcal{S}},1}, p) = d_{\text{geo}}\left([0, \ldots, 0, \theta_{\min}\left(\boldsymbol{\mathcal{S}}, \boldsymbol{\mathcal{P}}\right)]\right)$$

$$= \theta_{\min}\left(\boldsymbol{\mathcal{S}}, \boldsymbol{\mathcal{P}}\right). \qquad \square$$

COROLLARY 3.3.5. *The minimum geodesic distance between a point $p \in Gr(m, \boldsymbol{\mathcal{V}})$ and the Schubert variety $\Omega_{\boldsymbol{\mathcal{S}},1}$ is at least as discriminating*[8] *as that produced by any other increasing function of principal angles.*

PROOF. Let d_{geo} be the geodesic distance on $Gr(m, \boldsymbol{\mathcal{V}})$ and g_a be some arbitrary, real-valued, increasing function on ordered m-vectors of principal angles. Let $g^{\star}$ be defined as in Theorem 3.3.3 for each of these functions. Consider two points $p^{(1)}, p^{(2)} \in Gr(m, \boldsymbol{\mathcal{V}})$. If $g^{\star}_a(\Omega_{\boldsymbol{\mathcal{S}},a}, p^{(1)}) \neq g^{\star}_a(\Omega_{\boldsymbol{\mathcal{S}},a}, p^{(2)})$, then $\theta_{\min}(\boldsymbol{\mathcal{P}}^{(1)}, \boldsymbol{\mathcal{S}}) \neq \theta_{\min}(\boldsymbol{\mathcal{P}}^{(2)}, \boldsymbol{\mathcal{S}})$. Hence, $d^{\star}_{\text{geo}}(\Omega_{\boldsymbol{\mathcal{S}},a}, p^{(1)}) \neq d^{\star}_{\text{geo}}(\Omega_{\boldsymbol{\mathcal{S}},a}, p^{(2)})$. In other words, there is no real-valued, increasing function on ordered m-vectors of principal angles for which the minimum value to $\Omega_{\boldsymbol{\mathcal{S}},a}$ is more discriminating than that produced using the geodesic distance. $\square$

3.3.2. RECOVERY THEORY. Up to this point we have provided theoretical results which allow us to perform signal detection. Specifically, they allow us to convert point-to-point comparisons to point-to-linear model comparisons. Now we will move to signal recovery, determining the set of points produced by the "noise-free" model which best correspond to the data p. While we considered monotone increasing functions on the principal angles before, now we will require them to be strictly increasing. This guarantees that g restricted

[8]By "discriminating," we mean able to distinguish two points from each other, i.e., maps the two points to different values. A function is at least as discriminating as another when it can distinguish an two points that the second function maps to different values.

to $p \times \Omega_{\boldsymbol{\mathcal{S}},a}$ only achieves the minimum when $\boldsymbol{\theta}$, also restricted to $p \times \Omega_{\boldsymbol{\mathcal{S}},a}$ achieves the minimum stated in Lemma 3.3.2.

Before stating a general theorem, let us look at a few special cases. First, what happens when our data point $p \in Gr(m, \boldsymbol{\mathcal{V}})$ is actually in $\Omega_{\boldsymbol{\mathcal{S}},a}$? When an a-dimensional subspace of $\boldsymbol{\mathcal{P}}$ is an a-dimensional subspace of $\boldsymbol{\mathcal{S}}$, the data fits the model perfectly. Clearly we would expect p to be in the set of optimal solutions; and since $\boldsymbol{\theta}(\boldsymbol{\mathcal{P}}, \boldsymbol{\mathcal{P}}) = \mathbf{0}$, that is true. Are any other points in the set? Since the only point $x \in \Omega_{\boldsymbol{\mathcal{S}},a} \subset Gr(m, \boldsymbol{\mathcal{V}})$ with $\boldsymbol{\theta}(\boldsymbol{\mathcal{P}}, \boldsymbol{\mathcal{X}}) = \mathbf{0}$ is p, p is the only point in $\Omega_{\boldsymbol{\mathcal{S}},a}$ which minimizes g. This is the Schubert variety $\Omega_{\boldsymbol{\mathcal{P}},m}$.

Now suppose that p is not in $\Omega_{\boldsymbol{\mathcal{S}},a}$. As stated in Theorem 3.3.3, the minimum is

$$g^{\star}(\Omega_{\boldsymbol{\mathcal{S}},a}, p) = g\left([\mathbf{0}_{m-a}, \boldsymbol{\theta}_{1,\ldots,a}(\boldsymbol{\mathcal{S}}, \boldsymbol{\mathcal{P}})]\right).$$

Suppose $\theta_a(\boldsymbol{\mathcal{P}}, \boldsymbol{\mathcal{S}}) < \theta_{a+1}(\boldsymbol{\mathcal{P}}, \boldsymbol{\mathcal{S}})$. If $\boldsymbol{\mathcal{S}}_{1,\ldots,a}$ is the a-dimensional subspace spanning the principal directions of $\boldsymbol{\mathcal{S}}$ associated with $\boldsymbol{\theta}_{1,\ldots,a}(\boldsymbol{\mathcal{S}}, \boldsymbol{\mathcal{P}})$, then the best solutions must contain this subspace. If it contained a different a-dimensional subspace of $\boldsymbol{\mathcal{S}}$, it would, as a consequence of Lemma 3.3.2, have to contribute larger principal angles. For clarity, let us define $\boldsymbol{\mathcal{Q}}_{\boldsymbol{\mathcal{S}}\text{-fixed}} \doteq \boldsymbol{\mathcal{S}}_{1,\ldots,a}$.

We still need an additional $(m-a)$-dimensional subspace to form a point in $\Omega_{\boldsymbol{\mathcal{S}},a} \subset Gr(m, SpaceV)$. We require this subspace to only contribute principal angles of 0, so it must be a subspace of p. Additionally, since we need it to not interfere with the principal angles we carefully selected from $\boldsymbol{\mathcal{S}}$, it needs to be orthogonal to $\boldsymbol{\mathcal{Q}}_{\boldsymbol{\mathcal{S}}\text{-fixed}}$. We can draw any $(m-a)$-dimensional subspace from

$$\boldsymbol{\mathcal{Q}}_{\boldsymbol{\mathcal{P}}} \doteq \boldsymbol{\mathcal{P}} \cap \boldsymbol{\mathcal{Q}}^{\perp}_{\boldsymbol{\mathcal{S}}\text{-fixed}}.$$

We can think of the set of solutions as the Schubert variety formed by the flag $F = \boldsymbol{\mathcal{Q}}_{\boldsymbol{\mathcal{S}}\text{-fixed}} \subset \boldsymbol{\mathcal{Q}}_{\boldsymbol{\mathcal{S}}\text{-fixed}} + \boldsymbol{\mathcal{Q}}_{\boldsymbol{\mathcal{P}}}$ and minimum intersections $a < m$: $\Omega_{F,a<m}$. While we don't strictly need to make $\boldsymbol{\mathcal{Q}}_{\boldsymbol{\mathcal{P}}}$ orthogonal to $\boldsymbol{\mathcal{Q}}_{\boldsymbol{\mathcal{S}}\text{-fixed}}$, we will choose to emphasizes the orthogonal decomposition to underscore the logic behind the construction; the computation of principal angles centers around a special pair of orthogonal decompositions.

When does $b = \dim \boldsymbol{\mathcal{Q}}_{\boldsymbol{\mathcal{P}}} = (m-a)$? Exactly when $\theta_a(\boldsymbol{\mathcal{P}}, \boldsymbol{\mathcal{S}}) < \pi/2$.[9] In this case, clearly the set of minimizers is once again a single point. However, suppose $b > (m-a)$. Then you can think of the set of solutions as isomorphic to $Gr(m-a, b)$ and $\dim Gr(m-a,b) = (m-a)(b-(m-a)) > 0$.

So far this would seem to indicate that all sets of minimizers are Schubert varieties. What happens when, looking at $\boldsymbol{\theta}(\boldsymbol{\mathcal{S}}, \boldsymbol{\mathcal{P}})$, we find

$$\theta_{a-1} < \theta_a = \theta_{a+1} < \theta_{a+2}?$$

As before, we are forced to start with a fixed subspace $\boldsymbol{\mathcal{Q}}_{\boldsymbol{\mathcal{S}}\text{-fixed}} \doteq \boldsymbol{\mathcal{S}}_{1,\ldots,(a-1)}$. When it comes to selecting the next orthogonal direction, though, we have a choice. If we add any 1-dimensional linear subspace of $\boldsymbol{\mathcal{Q}}_{\boldsymbol{\mathcal{S}}\text{-choice}} \doteq \boldsymbol{\mathcal{S}}_{a,\ldots,(a+1)}$, we will contribute the same principal angle as we would with any other.

Once we have made our choice of 1-dimensional $\boldsymbol{\mathcal{S}}_a \subset \boldsymbol{\mathcal{Q}}_{\boldsymbol{\mathcal{S}}\text{-choice}}$, we know that the $(m-a)$-dimensional subspace of $\boldsymbol{\mathcal{P}}$ must come from $\boldsymbol{\mathcal{Q}}_{\boldsymbol{\mathcal{P}}} \doteq \left(\boldsymbol{\mathcal{P}} \cap \boldsymbol{\mathcal{S}}^{\perp}_{1,\ldots,a}\right)$; if our choice of $\boldsymbol{\mathcal{S}}_a$ could alter this, it would imply that we could have selected $\boldsymbol{\mathcal{S}}_a$ such that $\theta_a = 0$ and our chain of inequalities rules this out. Therefore, $\boldsymbol{\mathcal{Q}}_{\boldsymbol{\mathcal{P}}} = \boldsymbol{\mathcal{P}} \cap (\boldsymbol{\mathcal{Q}}_{\boldsymbol{\mathcal{S}}\text{-fixed}} + \boldsymbol{\mathcal{Q}}_{\boldsymbol{\mathcal{S}}\text{-choice}})^{\perp}$.

[9] For the case under consideration, this is true so long as $\dim \boldsymbol{\mathcal{S}} > a$.

As before, let $b = \dim \boldsymbol{\mathcal{Q}}_{\boldsymbol{\mathcal{P}}}$. So long as $b = (m-a)$, we can construct this set of minimizers as the Schubert variety. This time it is defined by the flag

$$F = \boldsymbol{\mathcal{Q}}_{\boldsymbol{\mathcal{S}}\text{-fixed}} \subset (\boldsymbol{\mathcal{Q}}_{\boldsymbol{\mathcal{S}}\text{-fixed}} + \boldsymbol{\mathcal{Q}}_{\boldsymbol{\mathcal{P}}}) \subset (\boldsymbol{\mathcal{Q}}_{\boldsymbol{\mathcal{S}}\text{-fixed}} + \boldsymbol{\mathcal{Q}}_{\boldsymbol{\mathcal{S}}\text{-choice}} + \boldsymbol{\mathcal{Q}}_{\boldsymbol{\mathcal{P}}})$$

and minimum intersections $(a-1) < (m-1) < m$. The set of points in $\Omega_{F,(a-1)<(m-1)<m}$ corresponds with the set of subspaces

$$\{\boldsymbol{\mathcal{Q}}_{\boldsymbol{\mathcal{S}}\text{-fixed}} + \boldsymbol{\mathcal{Q}}_{\boldsymbol{\mathcal{P}}} + \boldsymbol{\mathcal{T}} \mid t \in Gr(1, \boldsymbol{\mathcal{Q}}_{\boldsymbol{\mathcal{S}}\text{-choice}})\}.$$

Since $\dim \boldsymbol{\mathcal{Q}}_{\boldsymbol{\mathcal{S}}\text{-choice}} = 2$, $\Omega_{F,(a-1)<(m-1)<m}$ is isomorphic to $Gr(1,2)$.

However, if $b > (m-a)$, then we have a problem. We need to select exactly $(m-a)$ dimensions from a b-dimensional space and follow it up by selecting exactly 1 from the 2-dimensional one. If we try to construct a single Schubert variety, we will find ourselves unable to rule out cases where we either select extra dimensions from $\boldsymbol{\mathcal{Q}}_{\boldsymbol{\mathcal{S}}\text{-choice}}$ or extra ones from $\boldsymbol{\mathcal{Q}}_{\boldsymbol{\mathcal{P}}}$. Points in the previous solution select at least $(m-a)$ dimensions from $\boldsymbol{\mathcal{Q}}_{\boldsymbol{\mathcal{P}}}$. Points in the Schubert variety defined by flag

$$E = \boldsymbol{\mathcal{Q}}_{\boldsymbol{\mathcal{S}}\text{-fixed}} \subset (\boldsymbol{\mathcal{Q}}_{\boldsymbol{\mathcal{S}}\text{-fixed}} + \boldsymbol{\mathcal{Q}}_{\boldsymbol{\mathcal{S}}\text{-choice}}) \subset (\boldsymbol{\mathcal{Q}}_{\boldsymbol{\mathcal{S}}\text{-fixed}} + \boldsymbol{\mathcal{Q}}_{\boldsymbol{\mathcal{S}}\text{-choice}} + \boldsymbol{\mathcal{Q}}_{\boldsymbol{\mathcal{P}}})$$

and minimum intersections $(a-1) < a < m$ provides points with at least a dimensions from $\boldsymbol{\mathcal{Q}}_{\boldsymbol{\mathcal{S}}\text{-fixed}} + \boldsymbol{\mathcal{Q}}_{\boldsymbol{\mathcal{S}}\text{-choice}}$. The intersection of the two

$$\Omega_{F,(a-1)<a<m} \cap \Omega_{E,(a-1)<(m-1)<m}$$

enforces equality of intersection dimensions, rather than $\leq$, as described above. We can also think of this as the set

$$\{\boldsymbol{\mathcal{Q}}_{\boldsymbol{\mathcal{S}}\text{-fixed}} + \boldsymbol{\mathcal{T}}_{\boldsymbol{\mathcal{S}}} + \boldsymbol{\mathcal{T}}_{p} \mid t_{\boldsymbol{\mathcal{S}}} \in Gr(1, \boldsymbol{\mathcal{Q}}_{\boldsymbol{\mathcal{S}}\text{-choice}}), t_p \in Gr(m-a, \boldsymbol{\mathcal{Q}}_{\boldsymbol{\mathcal{P}}})\}$$

embedded in $Gr(m, \boldsymbol{\mathcal{V}})$.

Now that we have an idea of the variety of solution sets, we are ready to prove a general statement.

THEOREM 3.3.6. *Given a Schubert variety $\Omega_{\boldsymbol{\mathcal{S}},a}$ and a strictly increasing function on principal angles g, and a point $p \in Gr(m, \boldsymbol{\mathcal{V}})$, the set of points $B \subset \Omega_{\boldsymbol{\mathcal{S}},a}$ which minimize g is a subvariety. Specifically, B is either a Schubert variety itself or the intersection of two Schubert varieties.*

PROOF. Consider a point $t \in \Omega_{\boldsymbol{\mathcal{S}},a}$ such that $g(t,p) = g^{\star}(\Omega_{\boldsymbol{\mathcal{S}},a}, p)$. Since g is strictly increasing, if we have two such points $t^{(1)}$ and $t^{(2)}$, then $\boldsymbol{\theta}\left(\boldsymbol{\mathcal{T}}^{(1)}, \boldsymbol{\mathcal{P}}\right) = \boldsymbol{\theta}\left(\boldsymbol{\mathcal{T}}^{(2)}, \boldsymbol{\mathcal{P}}\right)$. From Lemma 3.3.2, this unique least element exists and is $[\mathbf{0}_{m-a}, \boldsymbol{\theta}_{1,\ldots,a}(\boldsymbol{\mathcal{S}}, \boldsymbol{\mathcal{P}})]$.

As previously discussed, if $p \in \Omega_{\boldsymbol{\mathcal{S}},a}$, then $B = \{p\} = \Omega_{p,m}$, which is a subvariety of $\Omega_{\boldsymbol{\mathcal{S}},a}$.

For the remainder of this proof, assume that $p \notin \Omega_{S,a}$, hence $\theta_a(\boldsymbol{\mathcal{P}}, \boldsymbol{\mathcal{S}}) > 0$. Suppose we have principal angles $\boldsymbol{\theta}(\boldsymbol{\mathcal{S}}, \boldsymbol{\mathcal{P}})$ where

$$0 \leq \theta_1 \leq \cdots \leq \theta_l < \theta_{(l+1)} = \cdots = \theta_a = \cdots = \theta_k < \theta_{(k+1)}.$$

We take this to include the case where $\theta_a = \theta_m$, i.e., $k = m$. As in the earlier discussion, let $\boldsymbol{\mathcal{S}}_{i,\ldots,j}$ be the subspace of $\boldsymbol{\mathcal{S}}$ identified with the principal angles $\boldsymbol{\theta}_{i,\ldots,j}$. Define the fixed part of $\boldsymbol{\mathcal{S}}$ to be $\boldsymbol{\mathcal{Q}}_{\boldsymbol{\mathcal{S}}\text{-fixed}} \doteq \boldsymbol{\mathcal{S}}_{1,\ldots,l}$ and the part of $\boldsymbol{\mathcal{S}}$ we get to choose from $\boldsymbol{\mathcal{Q}}_{\boldsymbol{\mathcal{S}}\text{-choice}} \doteq \boldsymbol{\mathcal{S}}_{(l+1),\ldots,k}$. Then each point in our set of minimizers consists of $\boldsymbol{\mathcal{Q}}_{\boldsymbol{\mathcal{S}}\text{-fixed}}$ extended by an $(a-l)$-dimensional

subspace of $\boldsymbol{\mathcal{Q}}_{\boldsymbol{\mathcal{S}}\text{-choice}}$ and an $(m-a)$-dimensional subspace of p which is orthogonal to the first two.

If $a = k$, then $\dim \boldsymbol{\mathcal{Q}}_{\boldsymbol{\mathcal{S}}\text{-choice}} = (a-l)$ and offers no actual choice. Here the fixed portion is actually $\boldsymbol{\mathcal{Q}}_{\boldsymbol{\mathcal{S}}} \doteq \boldsymbol{\mathcal{Q}}_{\boldsymbol{\mathcal{S}}\text{-fixed}} + \boldsymbol{\mathcal{Q}}_{\boldsymbol{\mathcal{S}}\text{-choice}}$. The part from p is $\boldsymbol{\mathcal{Q}}_{\boldsymbol{\mathcal{P}}} \doteq \boldsymbol{\mathcal{P}} \cap \boldsymbol{\mathcal{Q}}_{\boldsymbol{\mathcal{S}}}^{\perp}$. In this case, the set of solutions is the Schubert variety defined by the flag $F = \boldsymbol{\mathcal{Q}}_{\boldsymbol{\mathcal{S}}} \subset \boldsymbol{\mathcal{Q}}_{\boldsymbol{\mathcal{S}}} + \boldsymbol{\mathcal{Q}}_{\boldsymbol{\mathcal{P}}}$ and the minimum intersection is $a < m$. This is isomorphic to $Gr((m-a), \boldsymbol{\mathcal{Q}}_{\boldsymbol{\mathcal{P}}})$.

If $a < k$, we will have more than a single choice for $\boldsymbol{\mathcal{S}}_{1,\ldots,a}$. Let us consider the orthogonal decomposition $\boldsymbol{\mathcal{S}}_{1,\ldots,a} = \boldsymbol{\mathcal{S}}_{1,\ldots,l} \oplus \boldsymbol{\mathcal{S}}_{(l+1),\ldots,a}$. As already noted, $\boldsymbol{\mathcal{Q}}_{\boldsymbol{\mathcal{S}}\text{-fixed}}$ is the fixed portion $\boldsymbol{\mathcal{S}}_{1,\ldots,l}$, as the principal angle problem produces only a single subspace with principal angles $\boldsymbol{\theta}_{1,\ldots,l}(\boldsymbol{\mathcal{S}}, \boldsymbol{\mathcal{P}})$.

Choice comes in when selecting an $(a-l)$-dimensional subspace of $\boldsymbol{\mathcal{Q}}_{\boldsymbol{\mathcal{S}}\text{-choice}}$. Note that for all $\boldsymbol{u} \in \boldsymbol{\mathcal{Q}}_{\boldsymbol{\mathcal{S}}\text{-choice}}$, $\theta(\boldsymbol{\mathcal{P}}, \boldsymbol{u}) = \theta_a(\boldsymbol{\mathcal{P}}, \boldsymbol{\mathcal{S}})$. Therefore, for all $u \in Gr(a-l, \boldsymbol{\mathcal{Q}}_{\boldsymbol{\mathcal{S}}\text{-choice}})$, $\boldsymbol{\theta}(\boldsymbol{\mathcal{U}}, \boldsymbol{\mathcal{P}}) = \theta_a(\boldsymbol{\mathcal{S}}, \boldsymbol{\mathcal{P}}) \cdot [1, \ldots, 1]$.

Now we have shown that the subspaces associated with each minimizer contains an a-dimensional subspace

$$\boldsymbol{\mathcal{U}} \in \left\{ \boldsymbol{\mathcal{Q}}_{\boldsymbol{\mathcal{S}}\text{-fixed}} + \boldsymbol{\mathcal{T}}_{\boldsymbol{\mathcal{S}}} \mid t_{\boldsymbol{\mathcal{S}}} \in Gr(a-l, \boldsymbol{\mathcal{Q}}_{\boldsymbol{\mathcal{S}}\text{-choice}}) \right\}.$$

and that for each element (subspace) in the set, we can find a minimizer containing it. All that remains is to discover what $(m-a)$-dimensional subspaces of $\boldsymbol{\mathcal{P}}$ can be used to complete the construction for a given point.

As already noted, $\boldsymbol{\mathcal{P}}_{1,\ldots,(m-a)}$ is an $(m-a)$-dimensional subspace of $\boldsymbol{\mathcal{P}} \cap \boldsymbol{\mathcal{S}}_{1,\ldots,a}^{\perp}$. In this form, we need to know which subspace we chose for $\boldsymbol{\mathcal{S}}_{1,\ldots,a}$ before we know what options we have for $\boldsymbol{\mathcal{P}}_{1,\ldots,(m-a)}$.

Note that, because of orthogonality of the principal directions,

$$\boldsymbol{\mathcal{S}}^{\perp}_{1,\ldots,l} = \boldsymbol{\mathcal{S}}^{\perp}_{1,\ldots,k} + \boldsymbol{\mathcal{S}}_{(l+1),\ldots,k}$$

$$\boldsymbol{\mathcal{P}} \cap \boldsymbol{\mathcal{S}}^{\perp}_{1,\ldots,l} = \boldsymbol{\mathcal{P}} \cap \boldsymbol{\mathcal{S}}^{\perp}_{1,\ldots,k} + \underbrace{\boldsymbol{\mathcal{P}} \cap \boldsymbol{\mathcal{S}}_{(l+1),\ldots,k}}_{=\langle \mathbf{0} \rangle}$$

$$= \boldsymbol{\mathcal{P}} \cap \boldsymbol{\mathcal{S}}^{\perp}_{1,\ldots,k}.$$

So for any choice of $\boldsymbol{\mathcal{S}}_{1,\ldots,a}$, we can select our $\boldsymbol{\mathcal{P}}_{1,\ldots,(m-a)}$ to be any $(m-a)$-dimensional subspace of $\boldsymbol{\mathcal{Q}}_{\boldsymbol{\mathcal{P}}} = \boldsymbol{\mathcal{P}} \cap \boldsymbol{\mathcal{S}}^{\perp}_{1,\ldots,k}$.

Hence, the set of solutions is

$$\{\boldsymbol{\mathcal{Q}}_{\boldsymbol{\mathcal{S}}\text{-fixed}} + \boldsymbol{\mathcal{T}}_{\boldsymbol{\mathcal{S}}} + \boldsymbol{\mathcal{T}}_{p} \mid t_{\boldsymbol{\mathcal{S}}} \in Gr(a-l, \boldsymbol{\mathcal{Q}}_{\boldsymbol{\mathcal{S}}\text{-choice}}), t_p \in Gr(m-a, \boldsymbol{\mathcal{Q}}_{\boldsymbol{\mathcal{P}}})\} \tag{3.5}$$

We can write this as the intersection of the Schubert varieties. The Schubert variety drawing at least a dimensions from $\boldsymbol{\mathcal{Q}}_{\boldsymbol{\mathcal{S}}\text{-fixed}} + \boldsymbol{\mathcal{Q}}_{\boldsymbol{\mathcal{S}}\text{-choice}}$ is defined by the flag

$$F = \boldsymbol{\mathcal{Q}}_{\boldsymbol{\mathcal{S}}\text{-fixed}} \subset (\boldsymbol{\mathcal{Q}}_{\boldsymbol{\mathcal{S}}\text{-fixed}} + \boldsymbol{\mathcal{Q}}_{\boldsymbol{\mathcal{S}}\text{-choice}}) \subset (\boldsymbol{\mathcal{Q}}_{\boldsymbol{\mathcal{S}}\text{-fixed}} + \boldsymbol{\mathcal{Q}}_{\boldsymbol{\mathcal{S}}\text{-choice}} + \boldsymbol{\mathcal{Q}}_{\boldsymbol{\mathcal{P}}})$$

with minimum intersections $l < a < m$. The Schubert variety where points take at least $(m-a)$ dimensions from $\boldsymbol{\mathcal{Q}}_{\boldsymbol{\mathcal{P}}}$ is defined by the flag

$$E = \boldsymbol{\mathcal{Q}}_{\boldsymbol{\mathcal{S}}\text{-fixed}} \subset (\boldsymbol{\mathcal{Q}}_{\boldsymbol{\mathcal{S}}\text{-fixed}} + \boldsymbol{\mathcal{Q}}_{\boldsymbol{\mathcal{P}}}) \subset (\boldsymbol{\mathcal{Q}}_{\boldsymbol{\mathcal{S}}\text{-fixed}} + \boldsymbol{\mathcal{Q}}_{\boldsymbol{\mathcal{S}}\text{-choice}} + \boldsymbol{\mathcal{Q}}_{\boldsymbol{\mathcal{P}}})$$

and minimum intersections $l < (l-m-a) < m)$. Minimizers are exactly those which satisfy both of these conditions.

$$B = \Omega_{F,l<a<m} \cap \Omega_{E,l<(l+m-a)<m}$$

The intersection of two varieties is itself a variety, hence B is a subvariety. Therefore, the set of minimizers is a subvariety. In all cases, we constructed this set either with a single Schubert variety or with the intersection of two Schubert varieties. □

COROLLARY 3.3.7. *The set of minimizers is a single point when either*

(1) $\theta_a(\boldsymbol{\mathcal{P}}, \boldsymbol{\mathcal{S}}) < \theta_{a+1}(\boldsymbol{\mathcal{P}}, \boldsymbol{\mathcal{S}})$ *and* $\theta_a(\boldsymbol{\mathcal{P}}, \boldsymbol{\mathcal{S}}) < \pi/2$ *or*

(2) $p \in \Omega_{\boldsymbol{\mathcal{S}},a}$.

PROOF. This comes directly from the construction in the proof of Theorem 3.3.6. The first is the general case where there is only a single choice for the subspace in $\boldsymbol{\mathcal{S}}$ and a single one in p. The second is the special case where all choices from $\boldsymbol{\mathcal{S}}$ and p produce the same point. □

COROLLARY 3.3.8. *The set of minimizers is a Schubert variety when*

(1) the set of minimizers is a single point,

(2) $\theta_a(\boldsymbol{\mathcal{P}}, \boldsymbol{\mathcal{S}}) < \pi/2$, *or*

(3) $\theta_a(\boldsymbol{\mathcal{P}}, \boldsymbol{\mathcal{S}}) < \theta_{a+1}(\boldsymbol{\mathcal{P}}, \boldsymbol{\mathcal{S}})$.

PROOF. A single point is always a Schubert variety; aside from $p \in \Omega_{\boldsymbol{\mathcal{S}},a}$, this is a special case of the remaining two cases. The other two cases are exactly those in the proof of Theorem 3.3.6 where at least one of $\boldsymbol{\mathcal{S}}$ or p produces a single choice. □

COROLLARY 3.3.9. *The set of points in* $Gr(m, \boldsymbol{\mathcal{V}})$ *where there is a unique reconstruction has Haar measure 1.*

PROOF. Let us fix $\Omega_{\boldsymbol{\mathcal{S}},a}$. If $\Omega_{\boldsymbol{\mathcal{S}},a} = Gr(m, \boldsymbol{\mathcal{V}})$, this is trivially true from Corollary 3.3.7, as every $p \in Gr(m, \boldsymbol{\mathcal{V}})$ is in $\Omega_{\boldsymbol{\mathcal{S}},a}$.

```
1: function DetectSignal(g, (S, a), P)
2:     θ ← PrincipalAngles(S, P)                    ▷ See Figure 2.7
3:     return g([0_(m−a), θ_{1,...,a}])
4: end function
```

FIGURE 3.5. Algorithm for evaluating the detection function for a linear model and data point.

Then let us consider the case where $\Omega_{\boldsymbol{\mathcal{S}},a} \subsetneq Gr(m, \boldsymbol{\mathcal{V}})$. Clearly $\theta_a(\boldsymbol{\mathcal{P}}, \boldsymbol{\mathcal{S}}) < \pi/2$ almost everywhere, as perturbing p_a within $\boldsymbol{\mathcal{S}}^{\perp}_{1,\ldots,(a-1)}$ will result in $\theta_a(\boldsymbol{\mathcal{P}}, \boldsymbol{\mathcal{S}}) < \pi/2$ almost everywhere. The same argument can be made to show that $\theta_a(\boldsymbol{\mathcal{P}}, \boldsymbol{\mathcal{S}}) < \theta_{a+1}(\boldsymbol{\mathcal{P}}, \boldsymbol{\mathcal{S}})$ almost everywhere. Hence, the conjunction of these two cases is also almost everywhere. By Corollary 3.3.7, all of the points where this is true have unique minimizers in $\Omega_{\boldsymbol{\mathcal{S}},a}$.

These two cases cover every possible $\Omega_{\boldsymbol{\mathcal{S}},a} \subseteq Gr(m, \boldsymbol{\mathcal{V}})$, therefore the set of points in $Gr(m, V)$ with unique minimizers has a Haar measure of 1. □

3.3.3. SIGNAL DETECTION. Given a known model for signal formation, we can use the above geometric relationships to determine whether a given aggregate data point is close to signature containing data or not.

Since the null hypothesis with only broadband noise produces the Schubert variety which is all of $Gr(m, \boldsymbol{\mathcal{V}})$, this would be equivalent to the uniform distribution[10] on the Grassmann manifold. In the case where there is a non-trivial background clutter subspace, the Schubert variety is non-trivial as well, and the null hypothesis is no longer uniform on a Grassmann manifold.

Given a function g, Figure 3.5 provides a simple method for calculating $g^{\star}(\mathbf{S}, \mathbf{P})$.

[10]On the Grassmann manifold $Gr(m, \boldsymbol{\mathcal{V}})$, the uniform distribution is the Haar measure. In this setting, the Haar measure is the invariant measure under the orthogonal (in the case of $\boldsymbol{\mathcal{V}} = \mathbb{R}^n$) or unitary ($\boldsymbol{\mathcal{V}} = \mathbb{C}^n$) group action on $\boldsymbol{\mathcal{V}}$. It is interesting to note that the span of m i.i.d. spherically distributed vectors in $\boldsymbol{\mathcal{V}}$ maps to a uniformly distributed point on $Gr(m, \boldsymbol{\mathcal{V}})$. For a nice treatment of both of these ideas, see [27, Sections 1.4 and 2.2].

```
1: function SignalRecovery((S, a), P)
2:     θ ← PrincipalAngles(S, P)                          ▷ See Figure 2.7
3:     (Ŝ, ·) ← PrincipalVectors(S, P)                    ▷ See Figure 2.8
4:     l ← arg max_i θ_i < θ_a
5:     k ← arg min_i θ_a < θ_(i+1)
6:     X̂ ← GramSchmidtBasis([Ŝ_{1,...,k}, P])             ▷ See Figure 2.3
7:     if θ_a = 0 then
8:         l ← k                                          ▷ p ∈ Ω_{S,a}
9:     else if a = k then
10:        l ← a                                          ▷ unique best subspace in S
11:    end if
12:    return (Ŝ_{1,...,l}, Ŝ_{(l+1),...,k}, X̂_{(k+1),...})   ▷ (Q_S-fixed, Q_S-choice, Q_P)
13: end function
```

FIGURE 3.6. Algorithm to recover the signal given a model and data point.

3.3.4. SIGNAL RECOVERY. The goal here is to determine the signature component in the data. In the language that we have been using, we wish to find a point on the Schubert variety which is has a sufficiently small value of g. In the traditional signal processing literature, we really want to know the portion of this recovered subspace which intersects with F (e.g., $\boldsymbol{\mathcal{S}}$) or the parameterization of this reconstruction in the coordinates of $\boldsymbol{\mathcal{S}}$.

Given a model $(\mathbf{S}, a)$ and data $\mathbf{P}$, Figure 3.6 produces an orthogonal deconstruction of the portion of $\mathbf{S}$ which must be in the optimal reconstruction, the portion of $\mathbf{S}$ from which we must select a subspace, and the portion of $\mathbf{P}$ which best fills the remainder.

3.4. SUMMARY

In this chapter, we developed a framework for linear models on a Grassmann manifold. We began by defining the sorts of point-to-point functions that we would be concerning ourselves with. Then we presented a mapping from a class of linear models to sets of points called Schubert varieties. From here, we proved theorems for defining point-to-Schubert variety functions derived from the point-to-point functions. These functions can be used for

signal detection. Finally, we presented a reconstruction theorem which describes the set of "best" points in the Schubert variety given a data point. The cases where a unique "best" point exists were enumerated.

Algorithms for both detection and reconstruction based upon these theorems were provided. In the next chapter, we will explore the application of these algorithms to both synthetic and real data.

CHAPTER 4

Empirical Results

4.1. Introduction

In the previous chapter, we developed a theoretical framework for comparing linear models with experimental data. Specifically, we derived a class of functions on the Grassmann manifold comparing data points with Schubert varieties. From these results, we developed simple algorithms which exploit this framework.

In this chapter, we develop a basic end-to-end detection system based on the detection algorithm. This is applied to labeled hyperspectral imagery. We conclude with a discussion of the behavior of the system and challenges yet to be addressed.

4.2. System Architecture

The system provides an end-to-end detector. See Figure 4.1 for a block diagram of the system. When labeled data is given, the process begins first splits the data into training and testing sets. Then the training tiles for each class, κ, are fed to the model identification module, which produces a linear model for each class. For each of these models, the detection algorithm (see Figure 3.5) is run to produce a score. For classification, we choose the model with the lowest score.

First, we compile a list of tiles with uniform label; that is, every pixel in a tile has the same label. A tile is an spatially contiguous collection of pixels. In our system, we use a $k \times k$ square of pixels for $m = k^2$ samples of each spectral band. Each tile is aggregated as a point on $Gr(k^2, n)$, where n is the number of spectral bands.

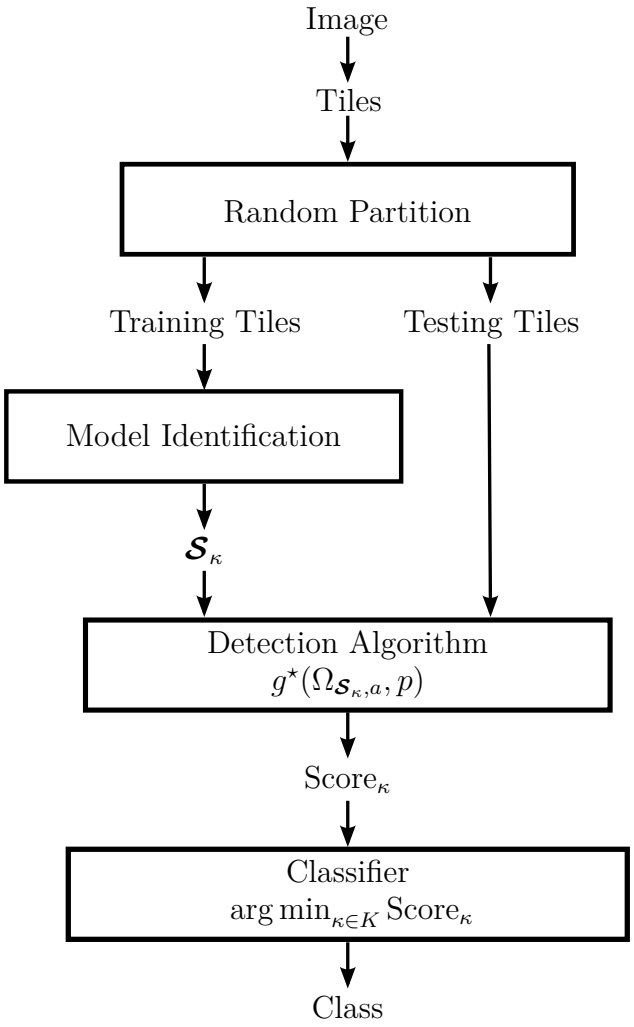


FIGURE 4.1. Block diagram of the classification system. The image, minimum intersection, a, and the set of classes under consideration, K, are given.

For each image, we perform 30 trials. For each trial, we randomly divide the tiles of each class into training data and testing data. We fit a linear model to the training data using one of the methods discussed in Section 4.2.1.

Then for each of the test tiles and each model, we compute a score using the detection algorithm. We identify the tile as belonging to the class whose detector provided the lowest score. We construct a confusion matrix which counts the number of tiles of each class identified as a given class. We compute the accuracy over all of the trials as the percent of correct classifications.

4.2.1. Model Identification. The success of any model-based detection algorithm hinges on identification of a sufficiently accurate model. The algorithm we propose is no exception. The literature is full of suggestions for determining a linear model from a set of data. For examples, see [24, 35, 19, 36, 26]. For these initial tests, we started at the simple end with PCA, which seems to be the standard. Then we tried minimum noise fraction (MNF), which was developed for visualizing hyperspectral imagery [24]. Finally we tried the relatively new flag of best fit, which conceptually seems to be a good match for our Grassmann-based classifier [26].

The classic methods PCA and MNF are generally employed to reduce the dimensionality of the data rather than to produce a linear model. In adhering to the hyperspectral image formation model, we feed the raw image data, not mean-subtracted data, into the these algorithms. See Algorithms 2.1, 2.2, and 2.4 for these subspace-fitting algorithms. Our Matlab implementations are included in Listing A.8.

With many of these model identification algorithms, we are given an ordered list of basis vectors. We then need to choose which basis vectors correspond to the subspace. In the broadest terms, this amounts to selecting the "best" subspace of the $m!$ possibilities in the power set. We make use of $\boldsymbol{\rho}$, the "energy" associated with the sequence of basis vectors. For simplicity, we assume that the best subspace is the span of the first k basis vectors, where k is the index of the knee of graph of $\{\rho_i\}_i$. We find the knee using the Kneedle method described by Satopaa, et al. [37]. See Listing A.8 for our Matlab implementation of this procedure.

4.2.2. Shortcomings. These experiments represent a first step in exploring the behavior of the Schubert variety-based method. The methodology described here has some shortcomings which future studies will need to correct.

In the current experiments, we are only considering tiles which have a single label. In reality, tiles will tend to have a mixture of labels, either at the pixel level or the subpixel level. In the future, it would be useful to observe responses on mixed tiles, as this is the situation in actual applications.

As will be discussed in more detail later, we are using relatively simple subspace fitting methods for model identification. While these provide a basis for building a test system, results based upon them should be viewed as a starting point rather than a statement of the intrinsic power of our approach.

4.3. Datasets

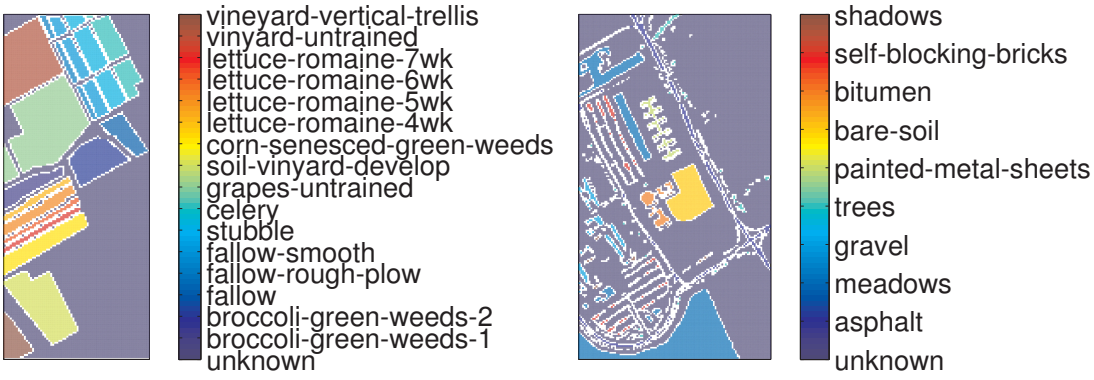
We used three hyperspectral imagery (HSI) datasets which appear in the HSI classification literature: Indian Pines, Salinas Valley, and Pavia University. The classes appearing in these are listed in Tables 4.1, 4.2, and 4.3. All three of these datasets are readily available and include a ground truth per pixel labeling for much of the image. Figure 4.2 shows the ground truth labelling of the uniform 3×3 tiles.

These datasets were each recorded using aerial hyperspectral sensors. Indian Pines is a popular dataset which was collected with the AVIRIS sensor in 1992 in Indiana, USA. The image is 145×145 pixels in 224 spectral bands. While some studies remove the water absorption bands prior to classification, we include all spectral bands.

Salinas Valley was also collected with the AVIRIS sensor in California, USA. The image is 512×217 pixels in 224 spectral bands.

Finally, the Pavia University dataset was acquired using the ROSIS sensor in Italy. The image is 610×340 pixels in 103 spectral bands. Unlike the other two datasets, which are of agricultural areas, Pavia University is an urban setting.

(a) Indian Pines



(b) Salinas Valley

(c) Pavia University

FIGURE 4.2. Ground truth label images for uniform tiles.

4.4. RESULTS

For each dataset, we chose a few sets of classes. In some cases, we intended them to be difficult by choosing a number of similar classes. For example, for the Salinas dataset, we attempt to distinguish between classes 11–14, which are all romaine lettuce at different stages of development. In other cases, we chose classes which we believed would be easier to differentiate, such as asphalt and meadow in the Pavia University dataset. We attempted to distinguish between all of the given classes from each other in each dataset.

We chose to report the accuracy, the proportion of classifications which are correct for the task (see Listing A.6 for our Matlab code). Tables 4.4, 4.5, and 4.6 report the accuracy achieved over all trials for each model identification method (PCA, MNF, and flag mean),

TABLE 4.1. Classes for Indian Pines data.

Label	Class	# Tiles
1	alfalfa	3
2	corn-notill	113
3	corn-mintill	75
4	corn	15
5	grass-pasture	33
6	grass-trees	57
7	grass-pasture-mowed	2
8	hay-windrowed	41
9	oats	0
10	soybean-notill	77
11	soybean-mintill	207
12	soybean-clean	49
13	wheat	14
14	woods	124
15	buildings-grass-trees-drives	31
16	stone-steel-towers	6

TABLE 4.2. Classes for Salinas data.

Label	Class	# Tiles
1	broccoli-green-weeds-1	192
2	broccoli-green-weeds-2	379
3	fallow	188
4	fallow-rough-plow	100
5	fallow-smooth	239
6	stubble	368
7	celery	334
8	grapes-untrained	1174
9	soil-vinyard-develop	635
10	corn-senesced-green-weeds	320
11	lettuce-romaine-4wk	83
12	lettuce-romaine-5wk	173
13	lettuce-romaine-6wk	62
14	lettuce-romaine-7wk	78
15	vinyard-untrained	772
16	vinyard-vertical-trellis	184

value of a, and task. This demonstrates the mean behavior, but does not indicate the best or worst case.

In nearly every case, two things are clear. First, our implementation of MNF produces noticeably lower quality classifications than either PCA or flag mean in nearly every case. It

TABLE 4.3. Classes for Pavia University data.

Label	Class	# Tiles
1	asphalt	279
2	meadows	1812
3	gravel	109
4	trees	61
5	painted-metal-sheets	65
6	bare-soil	513
7	bitumen	91
8	self-blocking-bricks	86
9	shadows	15

TABLE 4.4. Per category accuracy for Indian Pines tasks.

Classes	Accuracy					
	PCA		MNF		Flag Mean	
$a =$	1	2	1	2	1	2
all	0.5468	0.0000	0.1515	0.0000	0.4057	0.0000
2, 5	0.9797	0.7899	0.8872	0.7899	0.9684	0.7899
10, 11	0.6767	0.2645	0.4836	0.2645	0.6653	0.2645

did well in distinguishing classes 2 (meadows) and 4 (trees) in Pavia University, as well as in tasks which we identified as easy: differentiating classes 2 (corn-notill) and 5 (grass-pasture) in Indian Pines and classes 1 (asphalt) and 2 (meadows) in Pavia University. While we did not perform a statistical test, flag mean appears to produce classifiers which generally comparable to those generated using PCA.

Second, in nearly every case investigated, $a = 1$ produced a better classifier than $a = 2$. We only used these two values, because we only ever reliably produced signature subspaces of dimension 2. The single case where $a = 2$ excelled across the board was in distinguishing classes 2 (meadows) and 4 (trees) in the Pavia University dataset, although $a = 1$ results for this task were not far off. This may simply indicate that the signature subspaces overlap considerably and could be pruned and $a = 1$ used. Alternatively, it may suggest that a given class varies sufficiently over a small area that it is best described locally by a 2-dimensional subspace.

TABLE 4.5. Per category accuracy for Salinas tasks.

Classes	Accuracy					
	PCA		MNF		Flag Mean	
$a =$	1	2	1	2	1	2
all	0.8513	0.0360	0.1095	0.0360	0.8005	0.0360
1, 2	0.9859	0.3339	0.5570	0.3339	0.9268	0.3339
3–6	0.9472	0.2093	0.2846	0.2093	0.9769	0.2093
11–14	0.9882	0.2079	0.3224	0.2079	0.9593	0.2079
15, 16	0.9985	0.8101	0.5839	0.8101	0.9961	0.8101

TABLE 4.6. Per category accuracy for Pavia University tasks.

Classes	Accuracy					
	PCA		MNF		Flag Mean	
$a =$	1	2	1	2	1	2
all	0.6722	0.0918	0.4438	0.0918	0.3913	0.0918
1, 3, 6, 7	0.8468	0.2818	0.4185	0.2818	0.6278	0.2818
2, 4	0.9503	0.9694	0.8185	0.9694	0.7503	0.9694
1, 2	0.9806	0.1320	0.9568	0.1320	0.9369	0.1320

This study provides primarily annecdotal results. It was designed to provide direction for a more detailed investigation. While the average accuracy on various classification tasks does provide some guidance in constructing subspace models, it does not provide us with details of the behavior of each trial. Figure 4.3 offers an example of this detail.

We chose to show results for Indian Pines, as this is probably the most common dataset used in the HSI classification literature. Also, it is the smallest of the datasets considered, so comparison of individual trials can be achieved by inspection. Each row contains examples on a particular classification task: all classes, corn-notill vs. grass-pasture, and soybean-notill vs. soybean-mintill. Each column displays the classification produced for a given set of detectors, dubbed a "trial." We attempted to select trials which illustrate the range of behavior observed.

The first column of Figure 4.3 shows each task performed with the set of detectors produced during random trial 12. Figure 4.3(d) demonstrates an ability of this set of classifiers

to distinguish corn-notill from grass-pasture with only a single error. Attempting to distinguish these two classes within the full gamut of classes, as shown in Figure 4.3(a) results in significant misclassifications in most regions.

In contrast, the more difficult task of discriminating soybean-notill and soybean-mintill generally results in a number of misclassifications. Figure 4.3(g)-(i) show results using three different sets of detectors which range from favoring correct classification of soybean-notill at the expense of misclassification of soybean-mintill in (g) to the opposite in (h).

Without further investigation, the source of difficulty in discriminating some classes is not clear. Possible sources include the nature of the classes, the size and distribution of the training set, and the methods used for constructing a suite of models. The literature does note that some classes appear difficult to distinguish [9], so our difficulty may be inherent in the data set. This does not mean that our system is performing as well as it might.

We chose a small training set size, since some classes have few uniform tiles, realizing that this may penalize classes where we have a surplus of examples. The literature contains examples of infrequently occuring classes being discarded [38], so we could take this approach and then use larger training sets. We could also use a percentage of the data for training, rather than a fixed number of samples per class.

4.5. Challenges

4.5.1. Model Identification. Likely the greatest challenge is model selection. For this demonstration, we use relatively simple subspace fitting methods. We select the basis vectors based solely on the "energy" reported by the fitting method. For example, we use the singular values when employing PCA. Somewhat surprisingly, the simplest method (PCA) consistently outperforms the other two.

(a) all, Trial 12 (b) all, Trial 18 (c) all, Trial 20

(d) $\{2, 5\}$, Trial 12 (e) $\{2, 5\}$, Trial 18 (f) $\{2, 5\}$, Trial 20

(g) $\{10, 11\}$, Trial 12 (h) $\{10, 11\}$, Trial 18 (i) $\{10, 11\}$, Trial 20

FIGURE 4.3. Example Indian Pines classifications using PCA-generated subspace models ($a = 1$). Tiles are color coded by the best class (detector) using the same scale as in Figure 4.2(a).

In addition to the fitting method, selecting the basis vectors which provide the most discrimination also poses a challenge. The simple approach we employ appears to work well in some settings. As already noted, there are more sophisticated methods which take

into account information content. While outside of the scope of this dissertation, a more intelligent approach to subspace selection seems to be necessary, if we wish to produce a competitive classification system.

4.5.2. Parameter Selection. A related issue is that of selecting an appropriate value for a. In our examples, the estimated signal subspaces are quite small ($\dim \boldsymbol{\mathcal{S}} = 2$). This restricts the search to only two possible values, if we assume all models have the same value. In general, there is nothing that says that each signal model must have the same value for a. In other words, even in this simple case, if there are k classes, there are 2^k possible sets of classifiers.

It is an open question whether the minimum intersection dimension a can be selected for each detector independent of the full set of classifiers. We conjecture that this is the case, which would reduce the problem to that of selecting the most appropriate detector for each class rather than that of selecting the best ensemble of detectors. This belief follows, if our family of models fit the underlying nature of the world. More investigation is necessary, however, as our current study only provides annecdotal evidence

4.5.3. Dataset Selection. The datasets used here all indicate the simplest model ($\Omega_{\boldsymbol{\mathcal{S}},1}$) is the appropriate one and hence fail to take advantage of the power of our approach. We believe that there are datasets which will require $a > 1$, but locating one remains a challenge. We suggest that a collection in a variety of lighting and atmospheric conditions may demonstrate this. Locating such a dataset with ground truth will be difficult.

4.6. Summary

In this chapter, we demonstrated a reference implementation of our detection algorithm. We exercised the software on three hyperspectral imagery datasets commonly reported in the

literature. While not based upon a rigorous statistical analysis, it appears that the correct model for these datasets are of the form $\Omega_{\boldsymbol{\mathcal{S}},1}$.

CHAPTER 5

Conclusion

5.1. Contributions

This dissertation focused on the development of a principled framework for signal processing on the Grassmann manifold. Specifically, we made the following contributions.

- We introduced the association of a linear subspace model with a Schubert variety.
- We proved a novel and surprising ordering result on principal angles in Lemma 3.3.2.
- For Schubert varieties of the form $\Omega_{\boldsymbol{\mathcal{S}},a}$ increasing (decreasing) functions on principal angles, we proved both signal detection and signal recovery theorems (Theorems 3.3.3 and 3.3.6).
- We proved results using our framework which agree with existing signal detection theory in Euclidean space.
- We provided simple algorithms for applying the signal detection and recovery theorems and a Matlab implementation based upon these principles.

5.2. Future Directions

Our work was a fundamentally new approach. As a consequence, there are opportunities to apply the techniques developed, as well as to expand the techniques to more general cases.

The class of models which we addressed in this work provides an extension to traditional clutter-free models. There are two clear directions in which to grow. The first is to develop a Schubert variety-based approach that includes traditional models containing clutter. The second is to develop detection and recovery theorems which are applicable to general Schubert varieties, rather than only those of the form $\Omega_{\boldsymbol{\mathcal{S}},a}$.

So far, the problem domain in which we applied our technique did not benefit from the increased space of models; the best ones were of the form $\Omega_{\boldsymbol{\mathcal{S}},1}$, which is equivalent to standard clutter-free matched subspace detectors. We believe that there are applications where a model with $a > 1$ is necessary. The challenge is to identify such an application.

While our framework allows from detectors to be constructed from a broad class of functions, our practical implementation has been limited to simple, geometrically motivated functions such as the geodesic and chordal distances. The next step in this progression is to begin with a probability density function. The difficulty of developing these distributions on the Grassmann manifold originally lead us to develop our geometric framework. Returning to these questions may result in both specific results, as well as general techniques.

We view this dissertation as only the beginning. With clear opportunities to both expand the scope of the framework and to apply it to practical questions, there is no end of work in sight.

BIBLIOGRAPHY

[1] L. L. Scharf, *Statistical Signal Processing: Detection, Estimation, and Time Series Analysis*. Prentice Hall, 1991.

[2] S. Kraut, L. Scharf, and L. McWhorter, "Adaptive subspace detectors," *IEEE Transactions on Signal Processing*, vol. 49, no. 1, pp. 1–16, 2001.

[3] S. Kraut, L. Scharf, and R. Butler, "The adaptive coherence estimator: a uniformly most-powerful-invariant adaptive detection statistic," *IEEE Transactions on Signal Processing*, vol. 53, pp. 427–438, Feb. 2005.

[4] O. Besson, L. L. Scharf, and F. Vincent, "Matched direction detectors and estimators for array processing with subspace steering vector uncertainties," *IEEE Transactions on Signal Processing*, vol. 53, pp. 4453–4463, Dec. 2005.

[5] J.-M. Chang, *Classification on the Grassmannians: Theory and Applications*. PhD dissertation, Colorado State University, 2008.

[6] P. Turaga, A. Veeraraghavan, and R. Chellappa, "Statistical analysis on stiefel and Grassmann manifolds with applications in computer vision," in *Computer Vision and Pattern Recognition*, pp. 1–8, IEEE, June 2008.

[7] T. Wang and P. Shi, "Kernel grassmannian distances and discriminant analysis for face recognition from image sets," *Pattern Recognition Letters*, vol. 30, pp. 1161–1165, Oct. 2009.

[8] J.-m. Chang, C. Peterson, and M. Kirby, "Feature patch illumination spaces and karcher compression for face recognition via Grassmannians," *Advances in Pure Mathematics*, vol. 2, no. 4, pp. 226–242, 2012.

[9] S. Chepushtanova, C. Gittins, and M. Kirby, "Band selection in hyperspectral imagery using sparse support vector machines," in *SPIE DSS*, (Baltimore, Maryland), SPIE, 2014.

[10] L. Scharf and B. Friedlander, "Matched subspace detectors," *IEEE Transactions on Signal Processing*, vol. 42, no. 8, pp. 2146–2157, 1994.

[11] D. Manolakis, "Signal processing algorithms for hyperspectral remote sensing of chemical plumes," in *2008 IEEE International Conference on Acoustics, Speech and Signal Processing*, no. 2, pp. 1857–1860, IEEE, Mar. 2008.

[12] D. Manolakis, E. Truslow, M. Pieper, T. Cooley, and M. Brueggeman, "Detection algorithms in hyperspectral imaging systems: An overview of practical algorithms," *IEEE Signal Processing Magazine*, vol. 31, pp. 24–33, Jan. 2014.

[13] L. McWhorter, L. Scharf, and L. Griffiths, "Adaptive coherence estimation for radar signal processing," in *Conference Record of The Thirtieth Asilomar Conference on Signals, Systems and Computers*, vol. 1, pp. 536–540, IEEE Comput. Soc. Press, 1996.

[14] L. Scharf and L. McWhorter, "Adaptive matched subspace detectors and adaptive coherence estimators," in *Conference Record of The Thirtieth Asilomar Conference on Signals, Systems and Computers*, pp. 1114–1117, IEEE Comput. Soc. Press, 1996.

[15] O. Besson, L. L. Scharf, and S. Kraut, "Adaptive detection of a signal known only to lie on a line in a known subspace, when primary and secondary data are partially homogeneous," *IEEE Transactions on Signal Processing*, vol. 54, pp. 4698–4705, Dec. 2006.

[16] I. S. Reed and X. Yu, "Adaptive multiple-band CFAR detection of an optical pattern with unknown spectral distribution," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 38, no. 10, pp. 1760–1770, 1990.

[17] X. Yu, I. S. Reed, and A. D. Stocker, "Comparative performance analysis of adaptive multispectral detectors," *IEEE Transactions on Signal Processing*, vol. 41, no. 8, pp. 2639–2656, 1993.

[18] G. Yanfeng and Y. Zhamg, "Unsupervised subspace linear spectral mixture analysis for hyperspectral images," in *Proceedings 2003 International Conference on Image Processing*, vol. 1, pp. I–801–4, IEEE, 2003.

[19] J. M. P. Nascimento and J. M. Bioucas-Dias, "Vertex component analysis: a fast algorithm to unmix hyperspectral data," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 43, pp. 898–910, Apr. 2005.

[20] J. M. Bioucas-Dias and J. M. P. Nascimento, "Hyperspectral subspace identification," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 46, pp. 2435–2445, Aug. 2008.

[21] B. Moore, "Principal component analysis in linear systems: Controllability, observability, and model reduction," *IEEE Transactions on Automatic Control*, vol. 26, pp. 17–32, Feb. 1981.

[22] G. W. Stewart, "On the early history of singular value decomposition," *SIAM Review*, vol. 35, no. 4, pp. 551–566, 1993.

[23] R. Roger, "A faster way to compute the noise-adjusted principal components transform matrix," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 32, no. 6, pp. 1194–1196, 1994.

[24] A. Green, M. Berman, P. Switzer, and M. Craig, "A transformation for ordering multispectral data in terms of image quality with implications for noise removal," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 26, no. 1, pp. 65–74, 1988.

[25] B. Draper, M. Kirby, J. Marks, T. Marrinan, and C. Peterson, "A flag representation for finite collections of subspaces of mixed dimensions," *Linear Algebra and its Applications*, vol. 451, pp. 15–32, June 2014.

[26] T. Marrinan, J. R. Beveridge, B. Draper, M. Kirby, and C. Peterson, "Finding the subspace mean or median to fit your need," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1082–1089, 2014.

[27] Y. Chikuse, *Statistics on Special Manifolds*. New York, New York, USA: Springer-Verlag, 2003.

[28] A. k. Björck and G. H. Golub, "Numerical methods for computing angles between linear subspaces," *Mathematics of Computation*, vol. 27, no. 123, pp. 579–594, 1973.

[29] A. Edelman, T. a. Arias, and S. T. Smith, "The geometry of algorithms with orthogonality constraints," *SIAM Journal on Matrix Analysis and Applications*, vol. 20, pp. 303–353, June 1998.

[30] P.-A. Absil, A. Edelman, and P. Koev, "On the largest principal angle between random subspaces," *Linear Algebra and its Applications*, vol. 414, pp. 288–294, Apr. 2006.

[31] J. Von Neumann, "Some matrix inequalities and metrization of matrix space," *Tomsk University Review*, vol. 1, no. 11, pp. 286–300, 1937.

[32] L. Qiu, Y. Zhang, and C.-K. Li, "Unitarily invariant metrics on the Grassmann space," *SIAM Journal on Matrix Analysis and Applications*, vol. 27, pp. 507–531, Jan. 2005.

[33] P. J. Schreier and L. L. Scharf, *Statistical Signal Processing of Complex-Valued Data: The Theory of Improper and Noncircular Signals*. New York, New York, USA: Cambridge University Press, 2010.

[34] R. A. Horn and C. R. Johnson, *Topics in Matrix Analysis*. New York City, New York: Cambridge University Press, 1994.

[35] J. W. Boardman, "Geometric mixture analysis of imaging spectrometry data," in *Proceedings of IGARSS '94 - 1994 IEEE International Geoscience and Remote Sensing Symposium*, vol. 4, pp. 2369–2371, IEEE, 1994.

[36] U. Amato, R. M. Cavalli, A. Palombo, S. Pignatti, and F. Santini, "Experimental approach to the selection of the components in the minimum noise fraction," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 47, pp. 153–160, Jan. 2009.

[37] V. Satopaa, J. Albrecht, D. Irwin, and B. Raghavan, "Finding a "kneedle" in a haystack: Detecting knee points in system behavior," in *2011 31st International Conference on Distributed Computing Systems Workshops*, pp. 166–171, IEEE, June 2011.

[38] A. Chakrabarty, O. Choudhury, P. Sarkar, A. Paul, and D. Sarkar, "Hyperspectral image classification incorporating bacterial foraging-optimized spectral weighting," *Artificial Intelligence Research*, vol. 1, pp. 63–83, Aug. 2012.

APPENDIX A

Code

A.1. Algorithms

The following Matlab code implements the algorithms described in the text.

Listing A.1. principal_angles.m–Implementation of algorithms in Figures 2.7 and 2.8.

```
function [ theta, M_A, M_B ] = principal_angles( A, B )
%PRINCIPAL_ANGLES Compute the principal angles between subspaces.
%   A, B -
    econ = 0; % constant

    nargoutchk(1, 3);

    % compute orthonormal bases for A and B
    [ Q_A, ~ ] = qr(A, econ);
    [ Q_B, ~ ] = qr(B, econ);

    [U, S, V] = svd(Q_A' * Q_B);

    theta = acos(diag(S));

    if nargout >= 2
        M_A = Q_A * U;
    end
    if nargout >= 3
        M_B = Q_B * V;
    end
end
```

Listing A.2. detect_signal.m–Implementation of Figure 3.5.

```
function [ h ] = detect_signal( model, P )
%DETECT_SIGNAL compute the detection function for (model, P)
%   model -
%   P -

    assert(size(P, 1) == size(model.S, 1));
    assert(size(P, 2) == model.m);

    try
        theta = principal_angles(P, model.S);
        h = model.g([ zeros(1, model.m - model.a), theta(1:model.a) ]);
    catch
        % We could not calculate h, for some reason.
        % - Generally this occurs because a > size(S, 2).
        h = NaN;
    end
end
```

Listing A.3. recover_signal.m–Implementation of Figure 3.6.

```
function [ S_fixed, S_choice, P_choice ] = recover_signal( model, P )
%RECOVER_SIGNAL Compute the set of h-minimizers for (model, P)
%   model -
```

```
4       %   P -
5
6           vector_norm2 = @(M, axis) sum(M .* M, axis);
7
8           assert(size(P, 1) == size(model.S, 1));
9           assert(size(P, 2) == model.m);
10
11          [ theta, S_hat ] = principal_angles(model.S, model.P);
12          l = find(theta < theta(model.a), 'last');
13          k = find(theta(model.a) < theta, 'first');
14          X_hat = gram_schmidt_basis([ S_hat(1:k), P ]);
15          if theta(model.a) == 0 || model.a == k
16              l = k;
17          end
18
19          S_fixed = S_hat(:, 1:l);
20          S_choice = S_hat(:, (l + 1):k);
21          P_choice = X_hat(:, (k + 1):end);
22      end
23
24      function [ Q ] = gram_schmidt_basis( A )
25          Q(:, 1) = A(:, 1) / norm(A(:, 1));
26          for k = 2:n
27              z = A(:, k) - Q * Q' * A(:, k);
28              len_z = norm(z);
29              if len_z > eps('single')
30                  Q(:, end + 1) = z / len_z;
31              end
32          end
33      end
```

A.2. Experiments

The following Matlab code was used in Chapter 4 to perform the experiments on the system.

Listing A.4. `experiment.m`–Function to run an experiment.

```
1       function [ confusion ] = experiment( name, options )
2           %EXPERIMENT
3           %
4           %   Inputs:
5           %       name.
6           %       options.
7           %
8           %   Outputs:
9
10          import hsi.*
11
12          % Helper functions.
13          % - How many of each value in A?
14          count = @(A, values) arrayfun(@(value) sum(A(:) == value), values);
15          % - Get the tile (all bands) from the image.
16          %   The matrix is indexed by (bandIndex, pixelIndex).
17          % - TODO!  Select a random m-subset of the pixels when m < tileSize.
18          extract_tile = ...
19              @(image, tile) (reshape(image(tile(1):tile(3), tile(2):tile(4), :), ...
20                                      [], size(image, 3)))';
21
```

```
22          % Assign defaults for optional arguments.
23          defaultOptions = struct('dataRoot', '~/Data/HSI/HyperspectralRemoteSensingScenes/'
                , ...
24                                  'ntrials', 30, ...
25                                  'L', 3, ...
26                                  'overlapTiles', false, ...
27                                  'trainCount', false, ...
28                                  'trainPercent', false, ...
29                                  'method', 'PCA', ...
30                                  'zeroMean', false, ...
31                                  'a', [ 1 ], ...
32                                  'm', false, ...
33                                  'g', @d_geo, ...
34                                  'tasks', { { [] } }, ...
35                                  'generateLabelImage', false, ...
36                                  'labelImageRoot', '~/tmp/HSI/', ...
37                                  'labelText', false);
38          options = hsi.copyoptions(options, options, defaultOptions);
39          % Some defaults/conversions require additional processing.
40          % - options.m
41          if not(options.m)
42              % By default, m is the number of pixels in (full) tile.
43              if numel(options.L) == 1
44                  options.m = options.L ^ 2;
45              else
46                  options.m = prod(options.L);
47              end
48          end
49          % - options.g
50          if isa(options.g, 'char')
51              % We support a few common g functions, accessible by name string.
52              switch upper(options.g)
53                  case 'D_GEO'
54                      options.g = @d_geo;
55                  case 'D_CHORD'
56                      options.g = @d_chord;
57                  otherwise
58                      error('Unknown␣g␣function.');
59              end
60          end
61          assert(isa(options.g, 'function_handle'));
62
63          % Brittle!  We assume MAT files contain a single variable.
64          s = load([ options.dataRoot, name, '.mat' ]);
65          sFieldNames = fieldnames(s);
66          assert(numel(sFieldNames) == 1);
67          hsiImage = getfield(s, sFieldNames{1});
68          s = load([ options.dataRoot, name, '_gt.mat' ]);
69          sFieldNames = fieldnames(s);
70          assert(numel(sFieldNames) == 1);
71          labelImage = double(getfield(s, sFieldNames{1}));
72
73          % - Compute the detection score matrix.
74          %   The matrix is indexed by (tile, model).
75          compute_detection_matrix = ...
76              @(models, tiles) ...
77              reshape(cell2mat(cellfun(@(model) ...
78                                       cellfun(@(tile) ...
79                                                detect_signal(model, ...
```

```
                                                        extract_tile(hsiImage, ...
                                                                     tile)), ...
                                        num2cell(tiles, 2)), ...
                                    models, ...
                                    'uniformOutput', false)), ...
                [ size(tiles, 1), numel(models) ]);

        % Translate Labels to be 1:L.
        minLabel = min(labelImage(:));
        maxLabel = max(labelImage(:));
        % - Convert [] to 1:maxLabel
        options.tasks{cellfun(@(A) numel(A) == 0, options.tasks)} = ...
            [ minLabel:maxLabel ];
        % - Some labels start at 0, which cannot be used as an index.
        labelOffset = 1 - minLabel;
        minLabel = minLabel + labelOffset;
        maxLabel = maxLabel + labelOffset;
        labelImage = labelImage + labelOffset;
        options.tasks = cellfun(@(task) task + labelOffset, ...
                                options.tasks, ...
                                'uniformOutput', false);

        % Generate tiles.
        tiles = hsi.finduniformtiles(labelImage, ...
                                    struct('L', options.L, ...
                                           'overlapTiles', options.overlapTiles));

        if isa(options.labelText, 'cell')
            % Add 'unknown' as class label text for labels originally below 1.
            options.labelText = ...
                [ repmat({ 'unknown' }, [1, labelOffset]), ...
                  options.labelText ];
        else
            % Generate default class label text.
            options.labelText = ...
                arrayfun(@(label) sprintf('%d', label), ...
                         [ minLabel:maxLabel ] - labelOffset, ...
                         'uniformOutput', false);
        end

        if not(options.ntrials)
            % Generate baseline labelImage from ground truth...
            labelImageFilename = sprintf('%s/%s-%s.%s', ...
                                         options.labelImageRoot, ...
                                         name, ...
                                         'BASELINE', ...
                                         'eps');
            tileLabelImage = NaN(size(labelImage));
            for labelIndex = minLabel:maxLabel
                labelTiles = tiles{labelIndex};
                for tileIndex = 1:size(tiles{labelIndex}, 1)
                    tileLabelImage(labelTiles(tileIndex, 1):labelTiles(tileIndex, 3), ...
                                   labelTiles(tileIndex, 2):labelTiles(tileIndex, 4)) =
                                       ...
                           labelIndex;
                end
            end
            pcolor(tileLabelImage(end:-1:1, :));
            caxis([ minLabel, maxLabel ]);
```

```
138             % A better color scale is constructed using
139             %caxis([ minLabel - 0.5, maxLabel + 0.5 ]);
140             %colormap(hsv(maxLabel - minLabel + 1));
141             shading flat;
142             axis image;
143             axis on;
144             set(gca, 'xtick', [], 'ytick', []);
145             colorbar('YTick', minLabel:maxLabel, ...
146                      'YTickLabel', options.labelText);
147             set(gca, 'FontSize', 14);
148             set(gcf, 'renderer', 'painters');
149             print('-depsc', '-cmyk', labelImageFilename);
150             % ...and tile counts from ground truth
151             tileCounts = ...
152                 [ [ minLabel:maxLabel ] - labelOffset ; ...
153                   cellfun(@(tileList) size(tileList, 1), tiles) ]
154         end
155
156         % Run random trials.
157         confusionByTrial = cell([0, 0]);
158         for trial = 1:options.ntrials
159             % Partition the tiles into train and test sets.
160             [ trainTiles, testTiles ] = ...
161                 hsi.partitiontiles(tiles, ...
162                                    struct('trainCount', options.trainCount, ...
163                                           'trainPercent', options.trainPercent));
164             % Build a subspace model for each label.
165             S = cellfun(@(tiles) hsi.computesubspace(hsiImage, ...
166                                                      tiles, ...
167                                                      struct('method', options.method, ...
168                                                             'zeroMean', options.zeroMean, ...
169                                                             'm', options.m)), ...
170                         trainTiles, ...
171                         'uniformOutput', false);
172             for a = options.a
173                 models = cellfun(@(S) struct('S', S, ...
174                                              'a', a, ...
175                                              'm', options.m, ...
176                                              'g', options.g), ...
177                                  S, ...
178                                  'uniformOutput', false);
179                 % Compute the score for each test tile using each model.
180                 scores = ...
181                     cellfun(@(tiles) compute_detection_matrix (models, tiles), ...
182                             testTiles, ...
183                             'uniformOutput', false);
184                 for task = 1:numel(options.tasks)
185                     taskSize = numel(options.tasks{task});
186                     % For each tile in the task, pick the best model.
187                     % - This produces indices in task, not labels.
188                     [~, best] = ...
189                         arrayfun(@(label) min(scores{label}(:, options.tasks{task}), ...
190                                               [], ...
191                                               2), ...
192                                  options.tasks{task}, ...
193                                  'uniformOutput', false);
194                     % Generate the confusion matrix for {taskIndex, a}(:, :, trial).
195                     % - Matrix indexed by (trueIndex, bestIndex)
```

```
196                     confusionByTrial {task, a}(:, :, trial) = ...
197                         cell2mat(cellfun(@(b) count(b, 1:taskSize)', ...
198                                  best, ...
199                                  'uniformOutput', false))';
200                     if options.generateLabelImage
201                         % Generate a label image
202                         labelImageFilename = sprintf('%s/%s-%s-%d-%d-%d.%s', ...
203                                                      options.labelImageRoot, ...
204                                                      name, ...
205                                                      upper(options.method), ...
206                                                      a, ...
207                                                      task, ...
208                                                      trial, ...
209                                                      'eps');
210                         taskLabelImage = NaN(size(labelImage));
211                         for labelIndex = 1:numel(options.tasks{task})
212                             labelTiles = testTiles{options.tasks{task}(labelIndex)};
213                             for tileIndex = 1:numel(best{labelIndex})
214                                 taskLabelImage(labelTiles(tileIndex, 1):labelTiles(
                                        tileIndex, 3), ...
215                                                labelTiles(tileIndex, 2):labelTiles(
                                                    tileIndex, 4)) = ...
216                                     options.tasks{task}(best{labelIndex}(tileIndex));
217                             end
218                         end
219                         pcolor(taskLabelImage(end:-1:1, :));
220                         % A better color scale is constructed using
221                         %caxis([ minLabel - 0.5, maxLabel + 0.5 ]);
222                         %colormap(hsv(maxLabel - minLabel + 1));
223                         shading flat;
224                         axis image;
225                         axis on;
226                         set(gca, 'xtick', [], 'ytick', []);
227                         set(gca, 'FontSize', 14);
228                         set(gcf, 'renderer', 'painters');
229                         print('-depsc', '-cmyk', labelImageFilename);
230                     end
231                 end
232             end
233         end
234
235         % Compute min, max, and average confusion matrices for {taskIndex, a}.
236         confusion.min = cellfun(@(confusion) min(confusion, [], 3), ...
237                                 confusionByTrial, ...
238                                 'uniformOutput', false);
239         confusion.max = cellfun(@(confusion) max(confusion, [], 3), ...
240                                 confusionByTrial, ...
241                                 'uniformOutput', false);
242         confusion.mean = cellfun(@(confusion) mean(confusion, 3), ...
243                                  confusionByTrial, ...
244                                  'uniformOutput', false);
245     end
246
247     function [ distance ] = d_geo(theta)
248         distance = norm(theta);
249     end
250
251     function [ distance ] = d_chord(theta)
252         distance = norm(sin(theta));
```

```
253     end
```

LISTING A.5. `experimentsall.m`–Matlab script to run experiments and perform analysis.

```
1       path('~/Dropbox/Documents/MATLAB', path)
2       path('~/Dropbox/Research/Code', path)
3       path('~/Dropbox/Research/Code/Dissertation', path)
4       import hsi.*
5
6       % Generate the baseline images and tile counts.
7       experiment('Indian_pines', ...
8                  struct('ntrials', 0, ...
9                         'labelText', {{ 'alfalfa', ...
10                                        'corn-notill', ...
11                                        'corn-mintill', ...
12                                        'corn', ...
13                                        'grass-pasture', ...
14                                        'grass-trees', ...
15                                        'grass-pasture-mowed', ...
16                                        'hay-windrowed', ...
17                                        'oats', ...
18                                        'soybean-notill', ...
19                                        'soybean-mintill', ...
20                                        'soybean-clean', ...
21                                        'wheat', ...
22                                        'woods', ...
23                                        'buildings-grass-trees-drives', ...
24                                        'stone-steel-towers' }} ));
25      experiment('Salinas', ...
26                 struct('ntrials', 0, ...
27                        'labelText', {{ 'broccoli-green-weeds-1', ...
28                                        'broccoli-green-weeds-2', ...
29                                        'fallow', ...
30                                        'fallow-rough-plow', ...
31                                        'fallow-smooth', ...
32                                        'stubble', ...
33                                        'celery', ...
34                                        'grapes-untrained', ...
35                                        'soil-vinyard-develop', ...
36                                        'corn-senesced-green-weeds', ...
37                                        'lettuce-romaine-4wk', ...
38                                        'lettuce-romaine-5wk', ...
39                                        'lettuce-romaine-6wk', ...
40                                        'lettuce-romaine-7wk', ...
41                                        'vinyard-untrained', ...
42                                        'vineyard-vertical-trellis' }} ));
43      experiment('PaviaU', ...
44                 struct('ntrials', 0, ...
45                        'labelText', {{ 'asphalt', ...
46                                        'meadows', ...
47                                        'gravel', ...
48                                        'trees', ...
49                                        'painted-metal-sheets', ...
50                                        'bare-soil', ...
51                                        'bitumen', ...
52                                        'self-blocking-bricks', ...
53                                        'shadows'}} ));
54
```

```
% Run the experiments on the datasets.
%load('confusion.mat');
confusion.pca.IndianPines = experiment('Indian_pines', ...
                                       struct('a', [ 1, 2 ], ...
                                              'ntrials', 30, ...
                                              'tasks', { { [], ...
                                                           [1:16], ...
                                                           [2, 5], ...
                                                           [10, 11] } }, ...
                                              'trainCount', 4, ...
                                              'method', 'pca', ...
                                              'generateLabelImage', true));
confusion.mnf.IndianPines = experiment('Indian_pines', ...
                                       struct('a', [ 1, 2 ], ...
                                              'ntrials', 30, ...
                                              'tasks', { { [], ...
                                                           [1:16], ...
                                                           [2, 5], ...
                                                           [10, 11] } }, ...
                                              'trainCount', 4, ...
                                              'method', 'mnf', ...
                                              'generateLabelImage', true));
confusion.flag.IndianPines = experiment('Indian_pines', ...
                                        struct('a', [ 1, 2 ], ...
                                               'ntrials', 30, ...
                                               'tasks', { { [], ...
                                                            [1:16], ...
                                                            [2, 5], ...
                                                            [10, 11] } }, ...
                                               'trainCount', 4, ...
                                               'method', 'flag', ...
                                               'generateLabelImage', true));
save('confusion.mat', 'confusion');
confusion.pca.Salinas = experiment('Salinas', ...
                                   struct('a', [ 1, 2 ], ...
                                          'ntrials', 30, ...
                                          'tasks', { { [], ...
                                                       [1:16], ...
                                                       [1, 2], ...
                                                       [3, 4, 5, 6], ...
                                                       [11, 12, 13, 14], ...
                                                       [15, 16] } }, ...
                                          'trainCount', 4, ...
                                          'method', 'pca', ...
                                          'generateLabelImage', true));
confusion.mnf.Salinas = experiment('Salinas', ...
                                   struct('a', [ 1, 2 ], ...
                                          'ntrials', 30, ...
                                          'tasks', { { [], ...
                                                       [1:16], ...
                                                       [1, 2], ...
                                                       [3, 4, 5, 6], ...
                                                       [11, 12, 13, 14], ...
                                                       [15, 16] } }, ...
                                          'trainCount', 4, ...
                                          'method', 'mnf', ...
                                          'generateLabelImage', true));
confusion.flag.Salinas = experiment('Salinas', ...
                                    struct('a', [ 1, 2 ], ...
```

```
                                                'ntrials', 30, ...
                                                'tasks', { { [], ...
                                                            [1:16], ...
                                                            [1, 2], ...
                                                            [3, 4, 5, 6], ...
                                                            [11, 12, 13, 14], ...
                                                            [15, 16] } }, ...
                                                'trainCount', 4, ...
                                                'method', 'flag', ...
                                                'generateLabelImage', true));
    save('confusion.mat', 'confusion');
    confusion.pca.PaviaU = experiment('PaviaU', ...
                                      struct('a', [ 1, 2 ], ...
                                             'ntrials', 30, ...
                                             'tasks', { { [], ...
                                                          [1:9], ...
                                                          [1, 3, 6, 7], ...
                                                          [2, 4], ...
                                                          [1, 2] } }, ...
                                             'trainCount', 4, ...
                                             'method', 'pca', ...
                                             'generateLabelImage', true));
    confusion.mnf.PaviaU = experiment('PaviaU', ...
                                      struct('a', [ 1, 2 ], ...
                                             'ntrials', 30, ...
                                             'tasks', { { [], ...
                                                          [1:9], ...
                                                          [1, 3, 6, 7], ...
                                                          [2, 4], ...
                                                          [1, 2] } }, ...
                                             'trainCount', 4, ...
                                             'method', 'mnf', ...
                                             'generateLabelImage', true));
    confusion.flag.PaviaU = experiment('PaviaU', ...
                                       struct('a', [ 1, 2 ], ...
                                              'ntrials', 30, ...
                                              'tasks', { { [], ...
                                                           [1:9], ...
                                                           [1, 3, 6, 7], ...
                                                           [2, 4], ...
                                                           [1, 2] } }, ...
                                              'trainCount', 4, ...
                                              'method', 'flag', ...
                                              'generateLabelImage', true));
    save('confusion.mat', 'confusion');

    % Analyze the results.
    % - Compute accuracies.
    accuracy = struct();
    methods = fieldnames(confusion);
    for methodIndex = 1:numel(methods)
        filenames = fieldnames(getfield(confusion, methods{methodIndex}));
        for filenameIndex = 1:numel(filenames)
            eval(sprintf('%s.%s.%s␣=␣%s;', ...
                         'accuracy', ...
                         methods{methodIndex}, ...
                         filenames{filenameIndex}, ...
```

```
171                         'cellfun(@hsi.accuracy,␣getfield(getfield(getfield(confusion,␣
                                methods{methodIndex}),␣filenames{filenameIndex}),␣''mean''))'
                                ));
172         end
173     end
174     save('accuracy.mat', 'accuracy');
175     % - Output accuracies.
176     % - Brittle!  Assumes accuracy.pca exists and that all accuracy fields have
177     %   exactly the same fields.
178     filenames = fieldnames(accuracy.pca);
179     for filenameIndex = 1:numel(filenames)
180         filenames{filenameIndex}
181         cell2mat(cellfun(@(method) getfield(getfield(accuracy, method), ...
182                                             filenames{filenameIndex}), ...
183                         fieldnames(accuracy)', ...
184                         'UniformOutput', false))
185     end
```

A.3. Package hsi

The following Matlab code is the subset of our hsi package which the system detailed in Chapter 4 utilizes.

Listing A.6. +hsi/accuracy.m

```
1       function [ result ] = accuracy( confusion )
2       %ACCURACY Compute accuracy based upon the confusion matrix.
3           result = sum(diag(confusion)) / sum(confusion(:));
4       end
```

Listing A.7. +hsi/copyoptions.m

```
1       function obj = copyoptions(obj, options, defaultOptions)
2       %COPYOPTIONS Copy default and overloaded options into obj
3       %   Copy fields from defaultOptions or, if they exist, options into obj.
4
5           % Verify the option names
6           for s = (fieldnames(options))'
7               if ~isfield(defaultOptions, s)
8                   warning('HSI:invalidOptionName', ...
9                           '"%s"␣is␣not␣a␣valid␣option␣name', s);
10              end
11          end
12
13          % Copy default options or overloaded values into the structure
14          for s = (fieldnames(defaultOptions))'
15              if isfield(options, s)
16                  obj.(char(s)) = options.(char(s));
17              else
18                  obj.(char(s)) = defaultOptions.(char(s));
19              end
20          end
21
22      end
```

Listing A.8. +hsi/computesubspace.m

```
1       function [ S, rho ] = computesubspace( image, tiles, options )
2       %COMPUTESUBSPACE Compute the subspace the tiles best fit.
3       %
```

```
4       %   Inputs:
5       %       image.
6       %       tiles.
7       %       options.
8       %
9       %   Outputs:
10      %       S.
11
12          import hsi.*
13
14          % Helper functions.
15          % - Check to see if the vector is effectively zero.
16          iszerov = @(v) all(v < eps('single'));
17
18          % Assign defaults for optional arguments
19          defaultOptions = struct('method', 'PCA', ...
20                                  'zeroMean', false, ...
21                                  'm', false, ...
22                                  'N', false, ...
23                                  'threshold', false, ...
24                                  'test', false);
25          options = hsi.copyoptions(options, options, defaultOptions);
26
27          Z = tilestosamples(image, tiles);
28          if options.zeroMean
29              % - Mean-center the data.
30              meanZ = mean(Z, 2);
31              Z = Z - repmat(meanZ, [ 1, size(Z, 2) ]);
32          end
33          if options.test
34              % We are just running a test on a randomly generated basis.
35              % - S is
36              % - rho is
37              [ S, rho ] = test(options);
38          elseif numel(Z) == 0
39              % There is no data, so create 0-dimensional subspace.
40              S = NaN([ size(image, 3), 0 ]);
41          else
42              % There is data, so create a subspace as requested.
43              switch upper(options.method)
44                  case 'PCA'
45                      [ S, rho ] = computesubspacewithpca(Z, options);
46                  case 'MNF'
47                      [ S, rho ] = computesubspacewithmnf(Z, options);
48                  case 'FLAG'
49                      [ S, rho ] = computesubspacewithflag(Z, options);;
50                  otherwise
51                      error('Unknown␣method');
52              end
53              if options.zeroMean && ~iszerov(S \ meanZ)
54                  % - When centering the data, we adjoin the mean to the subspace.
55                  %   We only do this when this increases span(S).
56                  S = [ meanZ, S ];
57                  % - Since there is no real meaning to it, we assign the mean's rho
58                  %   to NaN.
59                  rho = [ NaN, rho ];
60              end
61          end
62      end
```

```
63
64      function [ samples ] = tilestosamples(image, tiles)
65          % Make a data matrix out of the tiles in the image.
66          % - samples is indexed by (bandIndex, pixelIndex).
67          nbands = size(image, 3);
68          samples = NaN([ size(image, 3), 0 ]);
69          for i = 1:size(tiles, 1)
70              tile = reshape(image(tiles(i, 1):tiles(i, 3), ...
71                                   tiles(i, 2):tiles(i, 4), ...
72                                   :), ...
73                             [], nbands);
74              % - We want a data point to form a column, so transpose.
75              samples(:, end + [1:size(tile, 1)]) = tile';
76          end
77      end
78
79      function [ k ] = samplecurvature(y)
80          % Naive difference approximations of the derivatives.
81          % - dy spans two samples to stay centered on the same samples.
82          dy = (y(3:end) - y(1:(end - 2))) / 2.0;
83          % - ddy is the difference between adjacent half-sample dy's.
84          ddy = y(3:end) + y(1:(end - 2)) - 2.0 * y(2:(end - 1));
85          % - Pad the curvature series to be index-compatible with y.
86          k = [ 0, (ddy .* ( 1.0 + dy .^ 2) .^ 1.5)' , 0 ];
87      end
88
89      function [ N ] = computeN(rho, options)
90          if numel(options.N) == 1
91              % We are just taking the first N dimensional subspace.
92              if not(options.N)
93                  % Automatically determine the best dimensions of the subspace.
94                  if options.threshold
95                      [ ~, options.N ] = find(rho > options.threshold, 'last');
96                  else
97                      % Cut off at the knee of the scree plot.
98                      % - This is based on Satopaa, et. al. "...Kneedle...", 2011.
99                      % - Unlike Kneedle, we do not smooth the data.  We assume that
100                     %   the distribution of values and the finite difference
101                     %   approximation does enough smoothing.
102                     [ ~, options.N ] = max(abs(samplecurvature(rho)));
103                 end
104             end
105             N = [ 1:options.N ];
106         else
107             N = options.N;
108         end
109     end
110
111     function [ S, rho ] = computesubspacewithpca(Z, options)
112         [Uz, Sz, Vz] = svd(Z, 'econ');
113         rho = diag(Sz);
114         N = computeN(rho, options);
115         S = Uz(:, N);
116     end
117
118     function [ S, rho ] = computesubspacewithmnf(Z, options)
119         % Helper functions.
120         % - Reverse the vector.
121         reversev = @(v) v(end:-1:1);
```

```
122         % - Reverse the columns of a matrix, then return the k-indexed columns.
123         reversemk = @(M, k) M(:, end - k + 1);
124         % - Give the columns unit length.
125         tounit = @(M) M ./ repmat(sqrt(sum(M .^ 2)), [ size(M, 1), 1 ]);
126         % - Check to see if the matrix appears illconditioned
127         isillconditioned = @(M) (rcond(M) < (max(size(M)) * max(eps(M(:)))));
128
129         % - dZ' * dZ is an estimate of the noise covariance.
130         dZ = Z - circshift(Z, [1, 0]);
131         [Uz, Vz, Xz, Cz, Sz] = gsvd(Z, dZ);
132         % - Noise is on the left, so reverse; now 1 is signal and end is noise.
133         % - Rho is the (estimated) signal-to-noise ratio of the new basis.
134         rho = reversev(diag(Cz) ./ diag(Sz));
135         N = computeN(rho, options);
136         if isillconditioned(Xz)
137             S = tounit(reversemk(Z * pinv(Xz)', N));
138         else
139             S = tounit(reversemk((Xz \ Z')', N));
140         end
141     end
142
143     function [ S, rho ] = computesubspacewithflag(Z, options)
144         X = cell2mat(cellfun(@orth, ...
145                              mat2cell(Z, ...
146                                       [ size(Z, 1) ], ...
147                                       options.m * ones(1, size(Z, 2) / options.m)), ...
148                              'uniformOutput', false));
149         [ Ux, Sx, Vx ] = svd(X, 'econ');
150         rho = diag(Sx);
151         N = computeN(rho, options);
152         S = Ux(:, N);
153
154     end
155
156     function [ SEst, rho ] = test(options)
157         import hsi.*
158         % Helper functions.
159         % - Give the columns unit length.
160         tounit = @(M) M ./ repmat(sqrt(sum(M .^ 2)), [ size(M, 1), 1 ]);
161         % - Check to see if the vector is effectively zero.
162         iszerov = @(v) all(v < eps('single'));
163
164         % - N is [ ambientDimension, subspaceDimension, nDataPoints ].
165         N = [ 30, 5, 15 ];
166         sigma = 1e-2;
167         STrue = sin((pi / N(1)) * [1:N(1)]' * randi(10, [ 1, N(2) ]));
168         Z = STrue * tounit(randn(N(2:3))) + sigma * tounit(randn(N([1, 3])));
169         if options.zeroMean
170             % - Mean-center the data.
171             meanZ = mean(Z, 2);
172             Z = Z - repmat(meanZ, [ 1, size(Z, 2) ]);
173         end
174         switch upper(options.method)
175             case 'PCA'
176                 SEst = computesubspacewithpca(Z, options);
177             case 'MNF'
178                 SEst = computesubspacewithmnf(Z, options);
179             case 'FLAG'
180                 SEst = computesubspacewithflag(Z, options);
```

```
181             otherwise
182                 error('Unknown␣method');
183         end
184         if options.zeroMean && ~iszerov(SEst \ meanZ)
185             % - When centering the data, we adjoin the mean to the subspace.
186             %   We only do this when this increases span(S).
187             SEst = [ meanZ, SEst ];
188         end
189         rho = hsi.principalangles(STrue, SEst);
190     end
```

LISTING A.9. +hsi/finduniformtiles.m

```
1       function [ tiles ] = finduniformtiles ( labelImage, options )
2       %FINDUNIFORMTILES Generate a list of tiles with uniform label.
3       %
4       %   Inputs:
5       %       labelImage. A (number of rows) x
6       %                     (number of cols) label image.
7       %       options
8       %       .ignoreOutliers. (Optional).
9       %       .L. (Optional).  The size of the tile used in [rows, cols].
10      %       .mask. (Optional).  A binary matrix the size of the image with 1
11      %                           for pixels to include and 0 for pixels not to
12      %                           include in building the flag.
13      %       .overlapTiles. (Optional).
14      %
15      %   Output:
16      %       tiles. A (max label) element cell array of 4 x # tile locations
17
18          import hsi.*
19
20          % Assign defaults for optional arguments
21          defaultOptions = struct('ignoreOutliers', false, ...
22                                  'L', [3, 3], ...
23                                  'mask', [], ...
24                                  'overlapTiles', false);
25          options = hsi.copyoptions(options, options, defaultOptions);
26
27          % L should be [rows, cols].  If L is a scalar, use an L x L tile.
28          if numel(options.L) == 1
29              options.L = [options.L, options.L];
30          end
31          L_2 = fix(options.L / 2);
32
33          imageSize = size(labelImage);
34
35          % If none was given, create a mask of the data of interest.
36          % (use = 1, ignore = 0)
37          if isempty(options.mask)
38              options.mask = single(true(imageSize(1:2)));
39          end
40          assert(size(options.mask, 1) == imageSize(1) && ...
41                 size(options.mask, 2) == imageSize(2));
42          % Create outlier mask.  (good = 1, outlier = 0)
43          if options.ignoreOutliers
44              outlierMask = ~hsi.finddropoutsstats(image, options.L, 22);
45          else
46              % Don't ignore outliers.  Use a trivial mask.
```

```
47              outlierMask = true(imageSize(1:2));
48          end
49          imageMask = options.mask & outlierMask;
50
51          % We do not want to end up with tiles trimmed by the image edge.
52          % - This is organized the same as a tile bound--[ ij_min, ij_max ].
53          bounds = [ (L_2 + 1), imageSize - L_2 ];
54          % Select the stride based upon whether the tiles should overlap or not.
55          if options.overlapTiles
56              stride = [ 1, 1 ];
57          else
58              stride = options.L;
59          end
60
61          % Find uniform tiles for each label/class.
62          % - initialize 1:max(label)
63          tiles{max(labelImage(:))} = [];
64          for i = bounds(1):stride(1):bounds(3)
65              for j = bounds(2):stride(2):bounds(4)
66                  ij = [ i, j ];
67                  % - We used to allow tiles to straddle the image boundary.  Use
68                  %   these two lines, if we return to that approach.
69                  %       ij_min = max([ (ij - L_2) ; 1, 1 ]);
70                  %       ij_max = min([ (ij + L_2) ; imageSize ]);
71                  ij_min = (ij - L_2);
72                  ij_max = (ij + L_2);
73                  uv_min = ij_min - ij + L_2 + 1;
74                  uv_max = ij_max - ij + L_2 + 1;
75                  tile_ij = labelImage(ij_min(1):ij_max(1), ij_min(2):ij_max(2));
76                  tile_labels = ...
77                      tile_ij(imageMask(ij_min(1):ij_max(1), ij_min(2):ij_max(2)));
78                  tile_label = tile_labels(1);
79                  if (all(tile_labels == tile_label))
80                      tiles{tile_label}(end + 1, :) = [ ij_min, ij_max ];
81                  end
82              end
83          end
84
85      end
```