PRACTICAL CONSIDERATIONS IN EXPERIMENTAL COMPUTATIONAL SENSING

by

Phillip K. Poon

A Dissertation Submitted to the Faculty of the

COLLEGE OF OPTICAL SCIENCES

In Partial Fulfillment of the Requirements
For the Degree of

DOCTOR OF PHILOSOPHY

In the Graduate College

THE UNIVERSITY OF ARIZONA

2016

THE UNIVERSITY OF ARIZONA
GRADUATE COLLEGE

As members of the Dissertation Committee, we certify that we have read the dissertation prepared by Phillip K. Poon titled Practical Considerations in Experimental Computational Sensing and recommend that it be accepted as fulfilling the dissertation requirement for the degree of Doctor of Philosophy.

Date: 9 December 2016
Amit Ashok

Date: 9 December 2016
Rongguang Liang

Date: 9 December 2016
Michael E. Gehm

Final approval and acceptance of this dissertation is contingent upon the candidate's submission of the final copies of the dissertation to the Graduate College.

I hereby certify that I have read this dissertation prepared under my direction and recommend that it be accepted as fulfilling the dissertation requirement.

Date: 9 December 2016
Dissertation Director: Amit Ashok

STATEMENT BY AUTHOR

This dissertation has been submitted in partial fulfillment of requirements for an advanced degree at the University of Arizona and is deposited in the University Library to be made available to borrowers under rules of the Library.

SIGNED: Phillip K. Poon

ACKNOWLEDGEMENTS

Graduate school is an arduous experience. It is difficult by nature. It forces one into a state of mind which embraces the edge of knowledge and trek into the unknown. I was fortunate to have many guides who showed me the path, even when there were times when I wandered off to get my bearings. Along the way I encountered many people who not only helped me with the journey but bestowed kindness and friendship, asking for nothing in return.

My main guide along the journey was Professor Michael Gehm. I first met him when I took a graduate level Linear Algebra course which I found particularly challenging. I often went to his office hours asking for help and his ability to be patient and explain concepts from different perspectives is a gift few teachers have. As an advisor, I would like to thank him for all of the help and guidance he has given me over the years. His generosity for funding my graduate studies as well as trips to conferences is appreciated. He believed in me more than I believed in myself. I consider him not only as a mentor but as a father figure.

I especially want to thank Professor Esteban Vera, who I first met as a postdoctoral researcher in the Laboratory for Engineering Non-Traditional Sensors (LENS) and supervised me for the majority of my graduate studies. Much of the work and results in this disseration is due to his guidance. Even after he started his professorship in Chile, he was willing to review my data and suggest different things to try. He is directly responsible for much of my training as an experimentalist and programmer.

I also thank Doctor Dathon Golish. His approach to work and life was a calming effect in often stressful times. He made major contributions to the Adaptive Feature Specific Spectral Imaging-Classifier (AFSSI-C) and provided valuable feedback on various research projects and conference presentations.

Thank you Professor Mark Neifeld and Professor Amit Ashok for being my advisor and supervisor during my first year as a PhD student. They were the first to introduce me to many of the techniques and subjects related to computational sensing. They taught me fundamental concepts in optics, statistical signal processing and programming. Many of the results in this dissertation would not have been possible without their teachings.

I've also had many other supervisors along the way whose effort should be acknowledged: My undergraduate advisor at San Diego State University, Professor Matthew Anderson. Doctor John Crane, who was my supervisor during my internship at the Lawrence Livermore National Laboratory. Professor Joseph Eberly and Professor Gary Wicks who were my advisors at the Institute of Optics at the University of Rochester.

I would like to formally express gratitude to a number of exceptional teachers throughout my life. Professor Tom Milster who taught Diffraction and Interfer-

ence and allowed me to be a teaching assistant for that course. Professor Masud Mansuripur, whose course in Electromagnetic Waves was the most elegant and well taught version of the classical nature of light that I have ever had the pleasure to experience. Professor Jeff Davis, who first ignited my passion for optics while I was an undergraduate physics student at San Diego State University.

I also want to thank several faculty members who committed time from their busy schedules to help with several milestones of my graduate school experience. Special thanks to Professor Julie Bentley, Doctor James Oliver, and Professor Richard Morris who wrote letters of recommendation for me. Appreciation goes to Professor Tom Milster, Professor Harrison Barrett, Professor Russell Chipman, and Professor John Greivenkamp who formed my oral comprehensive exam committee. Thank you to Professor Rongguang Liang who served on my doctoral dissertation committee.

I would like to thank several members of the Duke Imaging and Spectroscopy Program (DISP) laboratory for their friendship: Patrick Llull, Mehadi Hassan, Evan Chen, and Tsung Han Tsai.

Other graduate students, colleagues, and faculty must also be thanked, for at one time or another they all helped me: Basel Salahieh, Vicha Treeaporn, John Hughes, Myungjun Lee, Sarmad H. Albanna, Professor Lars Furenlid, Doctor Joseph Dagher, Professor Daniel Marks, Professor Janick Roland-Thompson, Mary Pope, Mark Rodriguez, and Amanda Ferris.

I've had the good fortune to form friendships with an amazing set of groupmates as part of the LENS. David Coccarelli invited our family to spend our first Thanksgiving in North Carolina with him and we had many discussions about college basketball and life. I want to express my sincere gratitude to Matthew Dunlop-Gray, who designed and constructed the AFSSI-C which is the foundation for much the work in this disseration. I learned so much from Matthew especially much of my practical skills. Tariq Osman constructed the Static Computational Optical Undersampled Tracker (SCOUT) which is also a major part of this dissertation. Alyssa Jenkins whose combination of sense of humor and intelligence is unmatched. Thank you to Qian Gong for your kindness, positivity, and generousity. Thank you Xiaohan Li for keeping me company that final year of graduate school, conversations about basketball and helping me with my math. Thank you David Landry for helping me with all software and computer programming related issues. Thank you Joel Greenberg for your help and advice. Thank you Kevin Kelly, Adriana DeRoos, Andrew Stevens and Dineshbabu Dinakarababu for your friendship. Finally, I consider Wei-Ren Ng as one of my best friends and as a unofficial brother. Our time in the LENS group was marked by many late nights spent working in the lab and office and mornings in the gym. He was generous in sharing his knowledge and gave me the advice that I often did not want to hear but was true.

Appreciation goes to the all the staff at the College of Optical Sciences at the University of Arizona. It is one of the most friendly and well run academic departments I have ever had the fortune to be a part of. I hope my career will reflect well upon the college.

Finally, I would like to thank my closest friends that I've met throughout the years. They often provided much needed respites during my journey—Christopher MacGahan, Ricky Gibson, Krista MacGahan, Kristi Behnke, Michael Gehl, Carlos Montances, Matthew Reaves, Vijay Parachuru, Eric Vasquez. Thank you for letting me into your lives and being part of mine.

Last but not least, to my family. You make me happy.

DEDICATION

For my wife. We moved from city to city. You stuck with me through the highs and lows. You cooked dinner for me when I came home from a long day. You did the chores so I could concentrate on research. You acted as both mother and father to our son while I wrote. You believed in me even when I did not. You sacrificed your dreams and goals so I could accomplish mine.

You're the real Ph.D.

Contents

Contents – *Continued*

Contents – *Continued*

List of Figures

List of Figures – *Continued*

List of Figures – *Continued*

List of Figures – *Continued*

ABSTRACT

Computational sensing has demonstrated the ability to ameliorate or eliminate many trade-offs in traditional sensors. Rather than attempting to form a perfect image, then sampling at the Nyquist rate, and reconstructing the signal of interest prior to post-processing, the computational sensor attempts to utilize a priori knowledge, active or passive coding of the signal-of-interest combined with a variety of algorithms to overcome the trade-offs or to improve various task-specific metrics. While it is a powerful approach to radically new sensor architectures, published research tends to focus on architecture concepts and positive results. Little attention is given towards the practical issues when faced with implementing computational sensing prototypes.

I will discuss the various practical challenges that I encountered while developing three separate applications of computational sensors. The first is a compressive sensing based object tracking camera, the Static Computational Optical Undersampled Tracker (SCOUT), which exploits the sparsity of motion between consecutive frames while using no moving parts to create a psuedo-random shift variant point-spread function. The second is a spectral imaging camera, the Adaptive Feature Specific Spectral Imaging-Classifier (AFSSI-C), which uses a modified version of Principal Component Analysis with a Bayesian strategy to adaptively design spectral filters for direct spectral classification using a digital micro-mirror device (DMD) based architecture. The third demonstrates two separate architectures to perform spectral unmixing by using an adaptive algorithm or a hybrid techniques of using Maximum Noise Fraction and random filter selection from a liquid crystal on silicon based computational spectral imager, the LCOS Computational Spectral Imager (LCSI).

All of these applications demonstrate a variety of challenges that have been addressed or continue to challenge the computational sensing community. One issue

is calibration, since many computational sensors require an inversion step and in the case of compressive sensing, lack of redundancy in the measurement data. Another issue is over multiplexing, as more light is collected per sample, the finite amount of dynamic range and quantization resolution can begin to degrade the recovery of the relevant information. A priori knowledge of the sparsity and or other statistics of the signal or noise is often used by computational sensors to outperform their isomorphic counterparts. This is demonstrated in all three of the sensors I have developed. These challenges and others will be discussed using a case-study approach through these three applications.

Chapter 1

Introduction

This chapter provides the motivation for the need to address the practical issues in experimental computational sensing. While computational sensing has the ability to ameliorate or eliminate trade-offs in many traditional isomorphic sensors, the sensor engineer is faced with a new of set challenges when designing a computational sensor. For example, while both computational and traditional isomorphic sensors often require calibration, a computational sensor can be more sensitive to calibration error due to the lack of redundancy in the measurement data. It is the author's hope that a full discussion on these various challenges will encourage more research on these issues.

Isomorphic sensing is the concept that the measurement data of a sensor resembles the signal-of-interest [1]. For example, in a camera, the digital image looks like the object. In isomorphic sensing the analog hardware, analog-to-digital converter (ADC), and processing algorithms are separate components, see Figure 1.1.

Computational sensing is the concept that a joint design of the sensor, often though active *coding* (often called structured illumination) or passive *coding* of the analog signal, with inversion algorithms will outperform the isomorphic sensor, see Figure 1.2 [2]. An example of computational sensing is the Adaptive Feature Specific Spectrometer (AFSS), where psuedo-arbitrary spectral filters adapt between measurements to quickly classify the input spectrum [3]. While *isomorphic* sensors can provide flexible sensing in multiple applications, joint design of a *computational sensor* will often lead to performance increases in resource constrained scenarios. Throughout this chapter and the rest of this dissertation, I will provide many exam-

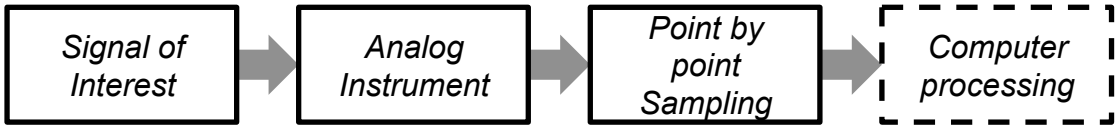

Figure 1.1: The signal-of-interest is incident upon the analog instrument. The analog instrument forms an isomorphism of the signal which is then periodically sampled point-by-point through an analog-to-digital converter (ADC) device. Once the signal is in digital form, post-processing algorithms are often used to perform various tasks such as noise reduction, detection, and classification. Notice that the analog instrument, sampling scheme, and processing are all seperated.

ples that highlight the differences between computational and isomorphic sensing.

Rather than a rigorous discussion, this chapter will discuss some of the major developments and concepts in the field of computational sensing on an intuitive level. This will familiarize the reader with important terminology and techniques common in the field. This chapter will also discuss some of the challenges I and many other experimentalists and engineers have faced when developing computational sensing prototypes. I will close this chapter with a brief look ahead to the rest of the dissertation.

1.1 Isomorphic Sensing

In Greek, the word isomorphic loosely translates to "equal in form." Traditional sensors perform isomorphic sensing. In the context of this dissertation, an isomorphic sensor is any sensor which attempts to produces measurement data that resembles the signal-of-interest. In this paradigm, the analog instrument, sampling scheme, and post-processing algorithms are separate components and processes.

I will discuss three important examples of isomorphic sensors: the pinhole camera, the photographic camera and the optical spectrometer[1]. These sensors have inspired many other optical and non-optical sensors throughout history, so it is natural to use them as examples for comparison when discussing computational sensing.

Before I continue, I want to define *measurement* because it can often be used in

[1]I will call the optical spectrometer just a spectrometer from now on, even though there are many instruments called spectrometers that not concerned with optical spectra

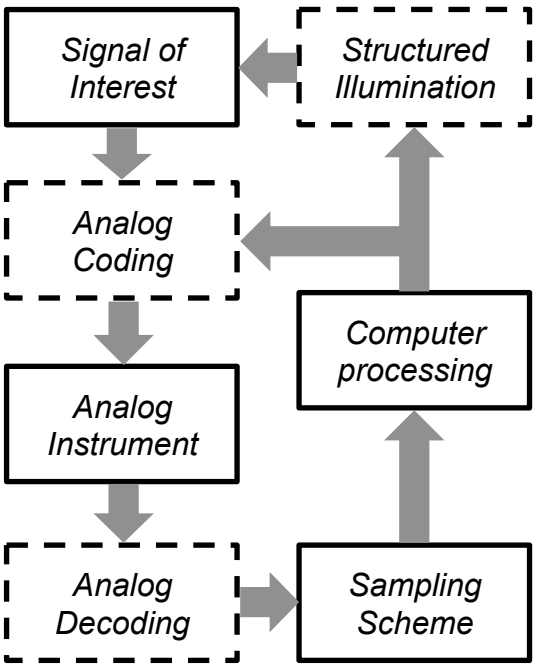

Figure 1.2: In a computational sensor, coding, decoding, and structured illumination are used in addition to the analog instrument, the algorithms, and sampling scheme. The dashed boxes represent optional processes that may or may not be needed.

an informal manner. In this dissertation a measurement has a very specific meaning. A measurement is a process that converts a physical phenomena to a collection of data. The signal-of-interest is the physical phenomena that one is interested in quantifying. I will call the collection of data the measurement data. The word sensing has a less precise meaning and I will use it in an informal manner. Sensing is any act that uses techniques, instruments, or processes that produces measurement data.

In the photographic camera, the signal-of-interest is the intensity distribution of the object. The analog instrument consists of the lenses which are designed and fabricated to produce an image that looks like the object at the FPA. The better the optics the more the image resembles the object. The FPA then samples the image and produces a digital representation of the intensity distribution of the object, the measurement data. If one is interested in performing a task such as detection or classification, the measurement data is sent to a post processing algorithm to perform those tasks.

There are two major sub-systems in the photographic camera which determine how well it performs: the optics and the FPA. Ideally, the optics (the analog instrument in this case) will produce a point-spread function (PSF) which is at the physical limit set by diffraction given the aperture size. For example, in a task such as the detection of a star from several neighboring stars in the night sky, if the PSF is much larger than the center to center seperation of the two stars in the optical image, it will be difficult to detect. Even if the PSF is small enough, the FPA must sample at a fine enough pixel-to-pixel spacing, called the *pixel pitch*, to accurately reproduce the intensity variations at the scale which is pertinent to the task. Intuitively, this makes sense because if the stars are imaged onto a single pixel, then one cannot ever hope to be able to accurately the detect the star without some other prior or side information.

The pinhole camera consists of a small hole and a box which prevents any light except from the pinhole from entering, see Figure 1.3. The pinhole camera is useful for imaging in parts of the electromagnetic spectrum and particles for which there is no direct analog to the refractive lens or reflective mirror. Like the photographic camera, the smaller the PSF diameter, the better the spatial resolution. Unfortunately, in the traditional pinhole camera, the only way to reduce the PSF diameter

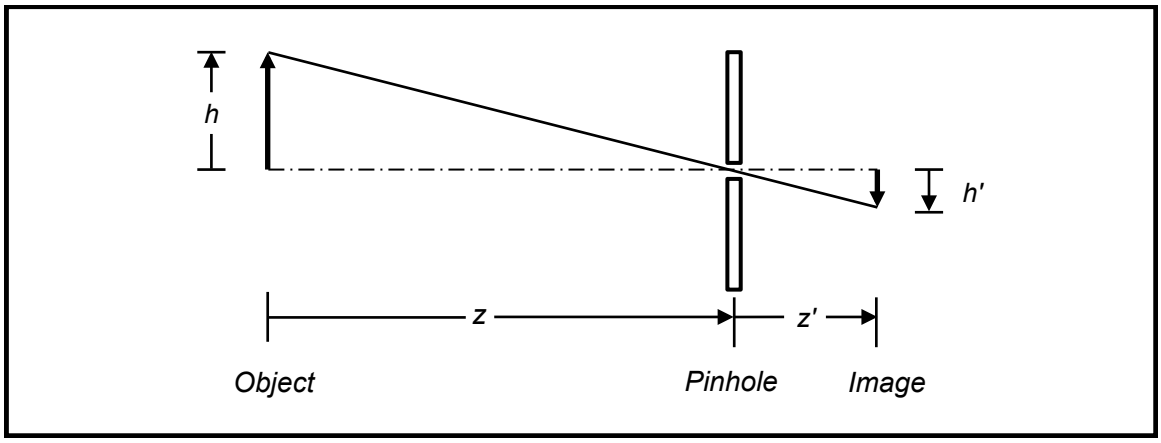

Figure 1.3: A pinhole camera is a simple imaging system that forms an image without a lens or mirror. This is due to the ray nature of light. A small hole will only admit a small amount of rays from an object point that is radiating light. Each point on a object emits light at different angles, and the image formed is a superposition of different rays. The smaller the hole, the less blurry the image. However, small holes also limit the amount of light

is by decreasing the diameter of the pinhole which reduces the amount of light.

In the spectrometer, the signal of interest is the spectrum of the object. The optics are designed to take the incoming light and separate various wavelength components, see Figure 1.4. The part of the spectrometer which is used to physically isolate the wavelengths is called a *monochromator*. The monochromator contains a prism or diffraction grating which creates a wavelength dependent angular seperation. The result is a spectral intensity as a function of position at the FPA. The FPA and post-processing algorithms are used in the same manner as the photographic camera, which is to sample the spatially varying intensity (which is now encoding spectral information) creating a digital version of it and to perform various tasks on the measurement data. For now, I will concentrate on the slit spectrometer, which measures the spectrum at a single spatial location on the object.

In the spectrometer, one of the important performance metrics is *spectral resolution*, which I denote δ_λ. The spectral resolution is the smallest difference in wavelength the instrument can discern. Large spectral resolutions can degrade the spectrometers ability to discern important parts of the spectrum. Similarly with the camera, the FPA must have a pixel pitch which is small enough to correctly sample the variations in the spectrum.

The point-by-point nature of isomorphic sensing is both a strength and a source

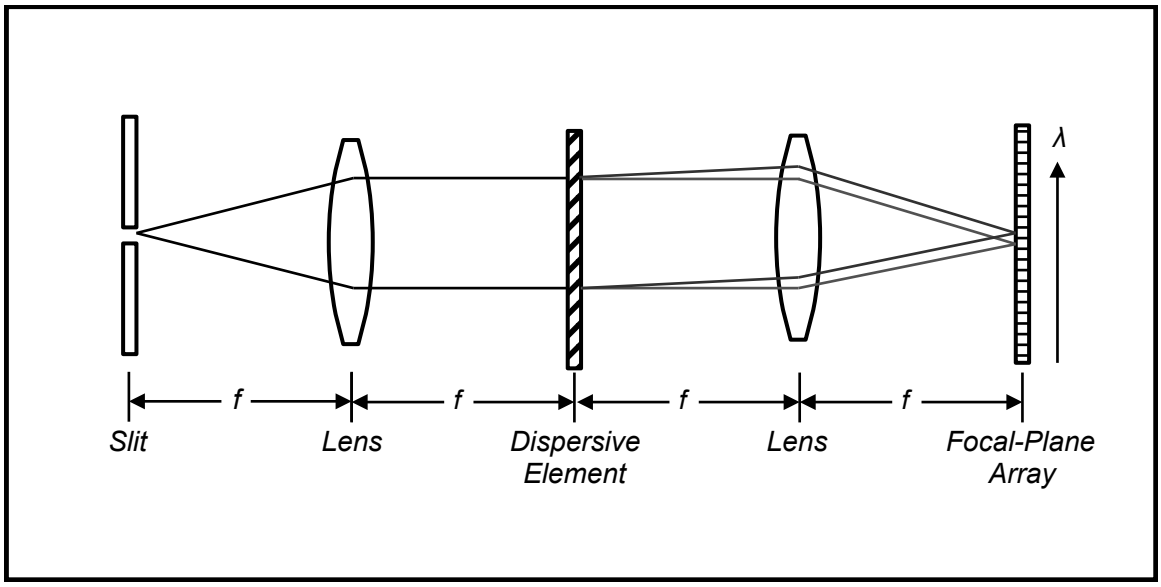

Figure 1.4: An isomorphic slit spectrometer with a 4F configuration. The slit limits the lateral extent of the object (or intermediate image). The light from the slit is collimated and separated into different angles based on the wavelength. A second lens then images wavelength shifted copies of the slit on the image plane where the detector is. As the slit size decreases, less light is allowed, but one gains spectral resolution by rejected light from neighboring locations on the object.

of weakness. The strength comes from the straightforward and intuitive architecture of the isomorphic sensor. Each subsystem: the optics, the focal-plane array (FPA), and the post-processing can be designed and constructed separately as long as they meet their individual specifications. As long as the signal-to-noise ratio (SNR) is sufficient and the sampling rate is high enough, one is guaranteed to recover the signal.

One of the weaknesses of the isomorphic approach is the ability to measure low SNR signals. Because the signal-of-interest is sampled in a completely point-by-point at each exposure, each pixel indepdently contributes a certain amount of noise. If the noise dominates, the measurement fidelity decreases, forcing the operator to increase the exposure time. For weak signals, the exposure time can become prohibitive and for temporally dynamic signals this leads to a loss of temporal resolution. Indeed, one of the major engineering trade-offs faced by traditional spectrometer designers is that when one attempts to increase the light collection (increased slit-width) the spectral resolution δ_λ degrades. Similarly, in the pinhole camera, there is a throughput versus spatial resolution trade-off, increasing the size of the pinhole degrades the PSF.

It would be easy to assume that recent progress in machine learning and statistical signal processing combined with the dramatic increase in computing power that one could simply post-process poor measurements and obtain useful data. However, this is not possible due to the an important theorem in information theory called the *data processing inequality* [4]. The information content of a signal cannot be increased through post-processing.

Another weakness of isomorhic sensing is that the seperation of the analog instrument, the sampling scheme, and the data processing algorithms lead to increased size, weight and power-cost (SWAP-C). As I mentioned in the photographic camera, the optics must be designed to produce a small PSF. For demanding applications, the optical design and fabrication can be the most expensive component of the sensor. While the price of FPAs sensitive to the visible wavelength region have fallen, FPAs sensitive to certain parts of the electromagnetic spectrum can be quite expensive or non-existant [5, 6].

In many cases, the signal is redundant and high resolution sampling becomes a waste of resources, such as data storage and communications bandwidth. A good example is in photography where often the post-processing takes the digital image and applies a compression algorithm which looks for patterns in the signal and reduces the file size, discarding much of the sample data [7].

The isomorphic sensor approach has served humanity well, however with all the weakness that I have discussed, there is a need for sensors which can operate in low-SNR conditions, with fewer measurement time, fewer measurements, or at lower SWAP-C while still producing useful information relavent to the sensor task. I will now begin to discuss some of major techniques in computational sensing that can be used to address some or all of the issues that I just stated.

1.2 Development of Multiplexing in Sensing

Multiplexing in sensing is the idea that each measurement sample is a physical combination of various parts of the analog signal-of-interest. Multiplexing is a powerful tool that can be exploited by the sensor designer to eliminate or relax SNR related trade-offs.

A simple example which illustrates the usefulness of multiplex sensing is weighing

objects. In this example, one needs to weigh 100 sheets of paper. Assume that the measurement error of the scale is insignificant. Isomorphic sensing means one would need to measure each sheet of paper individually, requiring 100 measurements.

If the measurement error of the scale is on the order of the weight of a single sheet, measuring each sheet individually produces a large measurement error. In order to reduce the error to an acceptable SNR one needs to make several measurements per sheet.

However, one can measure all 100 sheets at the same time. Since the weight of all 100 sheets is much larger than the measurement error of the scale, one can dramatically increase the precision of the measurement. If each sheet is the same weight, then the measurement process is finished.

The weighing problem is analogous to the spectroscopy example. As discussed earlier in section 1.1, there is trade-off between light collection and spectral resolution. Increasing the slit-width to increase the amount of light has the effect of degrading the spectral resolution δ_λ. Around the late 1940's and early 1950's, several important papers and inventions demonstrated the effectiveness of multiplexing in spectroscopy. At the time the FPA was non-existant, so in the slit spectrometer shown in Figure 1.4, where the FPA is pictured, there was actually another slit. To record the intensity at each spectral channel, either the dispersive element or the exit slit had to be mechanically translated, making the measurements even slower by a factor of N_λ, the number of spectral channels of interest.

Golay was the first to propose multiplexing the slit spectrometer by creating a pattern of binary (1's and 0's) entrance and exit slits [8]. In the Golay multi-slit spectrometer, the patterns of entrance and exit slits are matched based on mathematically useful properties. Intuitively, the ability to use multiple entrance and exit slits increases the optical throughput of the spectrometer. In communications theory, the process of structuring the data from the source to the receiver is referred to as coding. Similarly, in computational sensing, the transmission of information between an object signal-of-interest and the sensor is considered a coding problem [1]. In the multi-slit spectrometer, the entrance slits act to code the spectrum while the exit slits decode the coded spectrum. Golay's idea dramatically increased the optical throughput without degrading the spectral resolution.

Another example that is pertinent to this dissertation is coded aperture imag-

ing. Coded aperture imaging can be thought of as the multiplexed version of a pinhole camera. As mentioned earlier in section 1.1, there is a trade-off between the throughput and spatial resolution. However in many fields, such as high-energy particle imaging, refractive lenses and reflective mirrors are non-existant or underdeveloped. By using multiple pinholes the throughput is increased without sacrificing spatial resolution. However, the pattern of the pinholes (which is the code) must be carefully designed in order for the reconstruction to be feasible. Fenimore, Canon, and Gottesman were among the first to create an elegant solution to coded aperture design called uniformly redundant arrays [9, 10]. The uniformly redundant array increases throughput without significantly degrading the spatial resolution.

In summary, multiplexing has the ability to eliminate classic trade-offs in isomorphic sensors: signal strength or resolution. Modern researchers are still actively developing novel ways to implement multiplexing to increase resolution and sensitivity in the spatial domain [11, 12], spectral domain [13, 14], and temporal domain [15, 16]. However, multiplexing is not without its own set of challenges. As I mentioned, the coding must often be designed to obtain feasible signal reconstruction. I now discuss inverse problems in computational sensing.

1.3 Forward Models and Inverse Problems

In the computational sensing community, a model that explains the mapping of the signal-of-interest to the measurement data is called the *forward model*. The problem of taking the observed data and calculating a reconstruction of the signal-of-interst or task-specific parameters is called the *inverse problem*.

As you can imagine, solving inverse problems of isomorphic measurements, when one is concerned with reconstruction of the signal-of-interest, tend to be straightforward. In the weighing problem, the measurement is also the reconstruction. In the slit spectrometer, where the forward model can be simply the continuous to discrete mapping of the spectrum. The spectrum is the interpolated measurement.

Of course, one can begin to add levels of complexity to the forward model to account for various physical aspects of the sensor, such as the fact the FPA cannot measure certain wavelength regions or the noise in our measurements. But again, assuming proper sampling and enough SNR, the reconstruction of the isomorphic

signal is the measurement. This simplicity is one reason why isomorphic sensing still dominates at the consumer level despite all of the drawbacks I discussed earlier in section 1.1.

However, the multiplexing of signal information forces one to develop computational steps to solving the inverse problem. In the multiplexed weighing problem, a significant complication occurs when each sheet of paper has a different weight. Now solving the inverse problem is not as straightforward. In algebra, given only 1 equation and 100 unknowns, the problem is underdetermined. Similarly given a 1 measurement of all 100 sheets is also an underdetermined problem. What one can do is try measuring different combinations of the 100 sheets, each new combination provides a new equation to work with reducing the error. Naively, one might assume that randomly choosing 100 unique combinations and solve 100 equations using algebra. This works fine when there is no measurement error. However, in the presence of noise, in many applications including the weighing problem, random combinations are not the best way to conduct the coding. They are sub-optimal in terms of reconstruction error. This lead many to begin working on optimal coding strategies of signals for sensing and is major topic in this disseration.

In summary, the forward model of a sensor is essentially accounting for the physics which govern the measurement. While the solving the inverse problem is a mathematical problem which attempts to either reconstruct the object or to calculate task-specific data from the measurement data. Unfortunately, not all multiplexing forward models codes have mathematically elegant inversion steps. Often the physics of the situation force non-isomorphic measurements which require a computational step to solve the inverse problem.

1.4 Indirect Imaging

While Golay, Fennimore, and others were leveraging multiplexing to eliminate tradeoffs in traditional sensors, an entirely disparate group of researchers were working on imaging techniques for which there was no isomorphic analog. In these cases the physics of the sensing modality prevents a point-by-point sampling of the signal-of-interest. Indirect imaging refers to sensing schemes which include X-ray Computed Tomography (CT), Single-Photon Emission Computed Tomography

(SPECT), Positron Emission Tomography (PET), Magnetic Resonance Imaging (MRI) and certain forms of sonic and radio wave imaging. All require a data-processing or reconstruction step to solve an inverse problem [17].

Perhaps one of the most successful early examples of indirect imaging which led to the rise of inverse problems in sensing is the development of radar. While early radar was concerned with the detection and distance of an object, development of imaging radar began after World War II. Imaging radar and specifically Synthetic Aperture Radar (SAR) can use time delay information combined with the Doppler effect and interference of coherent radio waves to create high resolution images of terrain and buildings.

In medicine, a common imaging modality is X-Ray CT. In X-Ray CT, computational inversion is required to reconstruct a 2 or 3-dimensional function from 1 or 2-dimensional measurement data. The forward model can be simple: In a collimated beam architecture with a 1-dimensional detector array, each sample from each pixel on the array is proportional to the total number of x-ray photons that have not been absorbed by the object [18]. The inversion relies on computing the inverse Radon transform [19].

Indirect imaging is a subfield of computational sensing. Due to the medical and military applications of these computational sensors, there has been an intense push to reduce measurement time and improve task-specific and reconstruction results. Many of the techniques from other subfields of computational sensing have been brought to bear for indirect imaging [20, 21].

I have discussed the development of multiplex sensing and indirect imaging and how the ideas from both subfields are analogous in terms of producing a non-isomorphic measurement. However a major step in practical implementation of computational sensing is being able to obtain the measurements in a quick, reliable and efficient manner. Computational sensing as a field would not exist without the most important invention in optics and photonics of the 20th century, the digital image sensor[2].

[2] staring array, staring-plane array, focal-plane array (FPA), focal-plane, and image sensor are all synonymous

1.5 The Digital Imaging Revolution

The invention of the Charge-Coupled Device (CCD) FPA by Boyle and Smith in 1969 was a major breakthrough for entire fields and industries who depended on the reliable sampling, storage and transmission of optical signals [22]. The CCD is the first integrated circuit device that could reliably convert an optical image to a digital signal. Until then one either had to use film or bulky tubes that required an electron beam to be scanned across an image scene, such as the Image orthicon [23].

The invention of the digital camera by Sasson followed shortly after [24]. Several years later the first digital spectrometer was invented. In the digital spectrometer the exit slit of the monochromator was replaced by the CCD, which allowed for instant and simultaneous measurement of the entire spectrum in a compact architecture [25].

The development of the Complementary Metal–Oxide–Semiconductor (CMOS) FPA was also important. While in scientific settings, it could not rival the quality of the CCD, its cheaper cost brought digital imaging to the consumer level. Other technology like the digital computer and computer networking also provided major contributions to the democratization of imaging and optical sensing. While scientific grade optical instrumentation was and is still expensive, the researcher could at least capture, process, and share measurement data with significantly less effort. Without it, the field of computational sensing would not exist.

Algorithms for efficient and reliable storage and transmission of digital images became more important. Over time the pixel count continued to increase and the sheer volume of digital image and video data being generated and transmitted over networks began to outpace improvements in storage and transmission capacity. While many engineers developed new technology to combat the hardware limitations of storage and transmission. This also led to a renewed effort by researchers to develop more efficient image and video compression algorithms [26, 27].

Compression techniques all follow the same basic process, see Figure 1.5. Once the CCD samples the optical signal and produces the measurement data, the encoder uses the compression algorithm to look for redundancies in the data and produce a compact representation of the signal-of-interest. The compressed data can then either be stored or transmitted or both. The decoder solves the inverse problem of

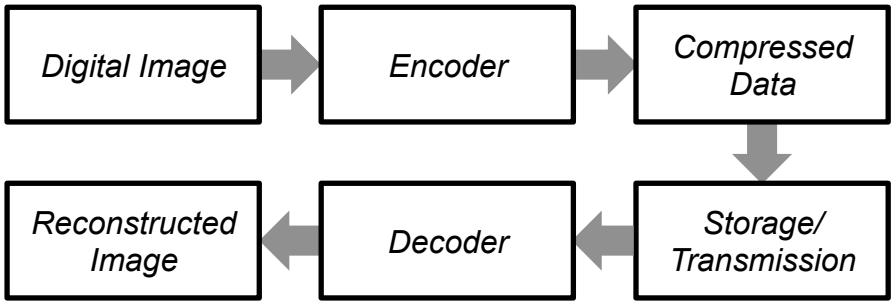

Figure 1.5: A general flowchart for image and data compression techniques. The digital image is analyzed by the encoder and compressed into a compressed form where it can be efficiently stored or transmitted. The decoder takes the compressed image and converts it back into an image that resembles the original digital image.

reconstructing the image.

In the next section I will discuss how many of the techniques and algorithms used in computational sensing are inspired by the techniques used by the image processing and digital communications community. This is because one of the major efforts of computational sensing is to make resource efficient measurements to reduce the total amount of measurements, whereas in image processing and communications, measurement data is often corrupted, missing, or too large for efficient storage and transmission.

1.6 Compressive Sensing

Traditionally, in order to increase the resolution of a sensor, one had to increase the number of measurements. This means that the SWAP-C must also increase. A camera with just a few megapixels FPA costs less than one with hundreds of megapixels. The cost of designing the optics will also need to scale to provide enough optical resolution. In a perfect world, one could capture all the information one needs from just a few measurements.

With a discrete signal one needs at least as many measurements as there are signal elements to solve the inverse problem. If the number of measurements is fewer, then the inverse problem is underdetermined. Conventional signal processing dictates that accurate reconstruction of the signal-of-interest is highly improbable. Fortunately, a signal acquisition technique called *compressive sensing* allows one to

design sensors that solve these types of highly underdetermined inverse problems.

As discussed earlier, much of the data being generated by sensors are redundant. Images, spectra, video, and audio data of real-world signals tend to exhibit patterns or redundancies that can be exploited. This allows a compression algorithm to significantly reduce the amount of data needed to represent the signal.

There is a class of compression algorithms called lossy [28]. In lossy compression, not only are redundancies exploited but data that is deemed insignificant to the signal quality is discarded. Only the most important part of the signal is kept as part of the compressed representation of the original signal. When the signal is uncompressed, the amount of data is less than the original measured data. The difference in quality is often unnoticeable to a human observer. In both lossless and lossy compression, the goal is to obtain a *sparse* representation of the signal. A sparse representation means that the signal can be well approximated with only a few non-zero elements in a representation basis. A representation basis is a basis in which the signal-of-interest is sparse. For example, most natural images are sparse in the Fourier basis. The representation basis is typically not the native basis of the signal-of-interest, i.e. pixel number or spectral channel.

Researchers pointed out that traditional sensors tend to produce vast amounts of measurement data, but often the majority of data is redundant and discarded in the compression step [29, 30, 31]. This led to the idea that sensors can be designed to directly measure the most relavent data in a signal, suggesting a measurement scheme that can measure a compressed form of the signal. This is the idea behind compressive sensing sometimes known as *compressive sampling*. If the measurements are compressive then it should be possible to significantly reduce the number of measurements to accurately reconstruct the signal.

Note that there is a distinction between compressive sensing and the traditional approach of sensing and then compressing. In the traditional approach, compression algorithms operate as a post-processing step. Therefore, a traditional compression algorithm will have access to the entire signal to look for redundancies and convert it into a sparse representation. In compressive sensing, one attempts the compression directly and therefore do not have access to the entire uncompressed signal. The algorithms must assume that the signal has a sparse representation.

The question of how to actually measure or code the analog signal to directly

obtain compressed data is also important. Fortunately, random coding tends to work well in many instances when the signal has a sparse representation. However in many cases, designed codes can significantly outperform random coding. I will discuss other types of coding schemes that can be used to outperform random codes.

The idea of compressive sensing seems to be similar to the concept of multiplex sensing. However, there is an important distinction to be made. In compressive sensing, the aim is to obtain the relavent information in as few measurements as possible. In multiplexing, the goal is to overcome limitations mainly due to lack of SNR. Many compressive sensing schemes also employ multiplexing.

One useful example of compressive sensing versus traditional sensing is the single pixel camera [11]. The single pixel camera is a multiplexing camera architecture that uses time sequential random measurements and recovers the image in significantly fewer measurements (equal to number of exposures $\times$ pixels) than the conventional camera, see Figure 1.6. Another example is the Coded Aperture Snapshot Spectral Imaging (CASSI) architecture [32], which can reconstruct a spectral data cube in significantly fewer FPA exposures than a traditional spectral imaging architecture.

Another important distinction is between reconstruction and task-specific sensing. Task-specific sensing tends to refer to measurement techniques that attempt to directly perform tasks such as detection, classification, and estimation without the intermediate step of reconstructing the high-dimensional signal. Compressive sensing is useful not just of overcoming resolution limitations in reconstruction but for task-specific sensing. For example, in facial recognition the goal is detection of an individual person. Reconstruction of the face image is simply an intermediate step, therefore, one can develop a compressive sensing scheme that is optimal for direct facial detection, skipping the step of image reconstruction [33].

Computational sensors can overcome classic engineering trade-offs in sensor design. However, there is a unique set of challenges related to computational sensing.

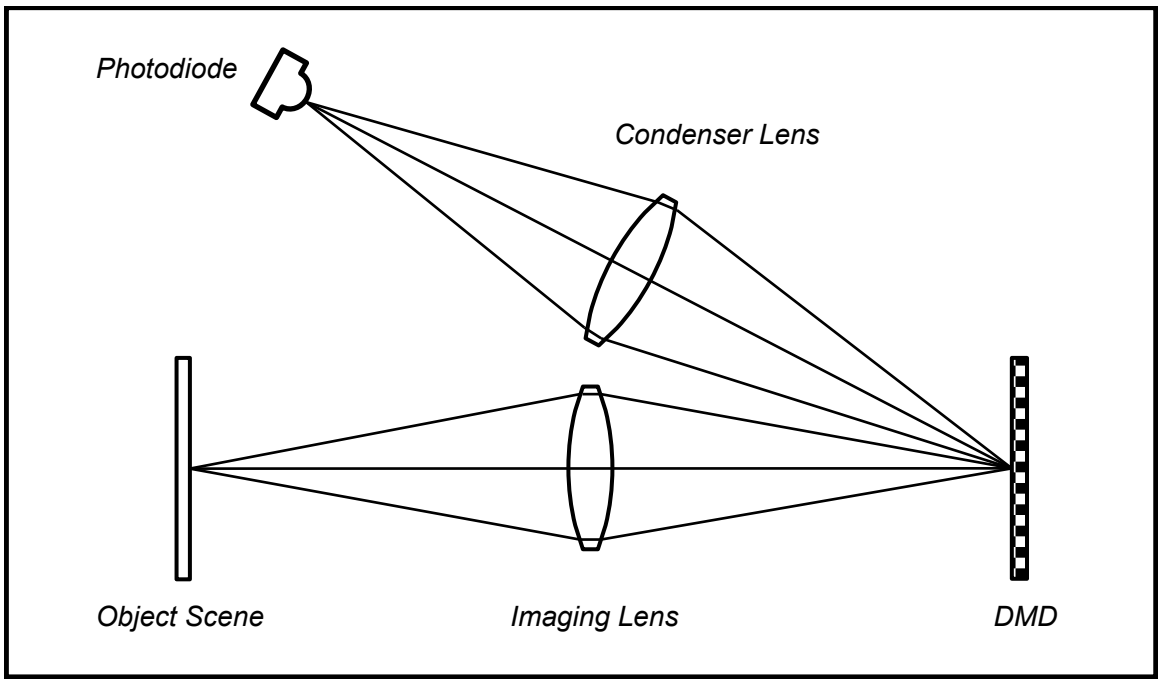

Figure 1.6: A single pixel camera architecture. The object scene (or intermediate image) is imaged onto a Digital Micro-Mirror Display (DMD). The each micro-mirror of DMD reflects light towards the photodiode or to another direction. This acts as a point-by-point multiplication of the discrete image with the DMD pattern. The condenser lens sums any light reflected by the DMD and focuses it onto the photodiode. This can be described mathematically as a vector multiplication of the image with the DMD pattern.

1.7 Practical Considerations in Computational Sensing

So far I have discussed the development of computational sensing techniques and how they are used to ameliorate trade-offs in traditional sensor design. Computational sensing as a field is continuing to grow at a rapid pace. The number of journal publications related to computational sensing has steadily increased every year since 2008 [34]. There is now a major Optical Society of America (OSA) meeting dedicated to computational sensing [35] and textbooks dedicated to its study and development [1, 36]. While it is a powerful approach to radically new sensor architectures, textbooks and papers tend to focus on architecture concepts and positive results. Little attention is given towards the practical issues one faces when implementing computational sensors. For example, calibration is a major topic in this dissertation. In many talks and papers on computational sensing, calibration is barely mentioned or relegated to a minor paragraph.

Calibration is the process of quantifying the response of a sensor in order to produce an accurate forward model. While many traditional sensors also require calibration, computational sensors tend to be more sensitive to calibration error.

There are two main reasons for this. The first reason is simply due to the fact that non-isomorphic measurements require a computational step to solve an inverse problem. The algorithms rely on accurate knowledge of the forward model to separate measurement data due to the instrument and data due to the signal-of-interest.

The second reason is due to the lack of redundancy of measurement data in compressive sensing. The redundancies that are typically deemed wasteful in traditional sensing, can also be used by post-processing algorithms to solve the inverse problem in a robust fashion to correct for missing or corrupted data [37]. In compressive sensing, only a few numbers are used to represent many. If the few numbers are misinterrepted due to poor calibration, it can have a drastic impact on the performance of the estimation algorithm. I will illustrate in this dissertation the calibration challenges in several computational sensors.

Calibration has become a major drawback in compressive sensing. A consumer

cannot be expected to spend time calibrating every time the instrument is physically bumped, the air temperature or pressure changes. In high dimensional compressive sensors like hyperspectral imagers, the calibration time can last hours.

One issue that is encountered in multiplexed sensing is lack of dynamic range. As the amount of light is sensed by the detector increases, it becomes more difficult to discern differences between signals. For example, in a single sinusoidally amplitude modulated signal, there is a DC offset and then the modulation itself. The pertinant information is encoded in the modulation so being able to resolve the peak to peak difference is important. Now imagine another amplitude signal with a different frequency. Physically, the amount of electrons in each pixel well is increased and the total modulation will tend to "average" out. The situation gets worse as the number of signals increases per measurement. This is one of the potential issues faced by single-pixel compressive sensing architectures. The SCOUT architecture attempts to alleviate this by "spreading" the photons onto more pixels for a compressive measurement.

Another hurdle in the implementation of practical computational sensing is the need for prior knowledge. For example, in *compressive imaging* one must assume the signal is sparse in some basis. Fortunately most realistic objects can be treated as such with many commonly known bases, such as a type of wavelet basis. However, sometimes one needs to image something that is difficult to represent in any known basis. One needs to resort to generating training data. Training requires one to generate many different examples of the signal. This becomes time and computationally expensive. Another example for the need of prior knowledge is the AFSSI-C, a computational spectral classifier which requires knowledge of the standard deviation of the probability density function of the noise to perform spectral classification in the least number of measurements as possible.

Reducing the number of detector elements is often the goal in computational sensing. A notable example is the single pixel camera is an architecture. However, the single pixel camera requires several time sequential measurements. Each measurement displays a different DMD pattern to create a randomly encoded measurements [11]. The drawback to this architecture is that one must point the camera at the object scene until enough measurements have been collected for proper reconstruction. A complication arises when this architecture is used to image temporally

varying object scenes. One must display the DMD patterns even faster and reduce the exposure time to keep up. As a result the SNR may begin to degrade.

A possible way to mitigate this issue is to do all the encoding in parallel. However, a completely parallel approach would require a lens, a DMD (or coded aperture), and a detector pixel for each measurement. Since each lens uses a different entrance pupil, this means that each detector pixel will have a different view of the object scene. This drastically scales the complexity of the architecture and algorithms. In this dissertation, I will discuss a compromise to parallel coding, in two different computational sensors, by using a common entrance pupil and a CCD.

Much of the optimal measurement codes contain simultaneously positive and negative measurement weights. In reality, with incoherent light one is unable to make negative measurements. One is often forced into situations where one must record two sets of measurements and subtract one set of measurements from the positive set of measurements. This means an additional noise term is added to each effective measurement.

Algorithms engineered to solve inverse problems often do not account for the non-negativity of many physical situations. For example, in spectral unmixing where the problem is to solve for the concentration of each material given a mixed spectrum. A non-negative fractional abundance that sums to one is a physical requirement. However, there is a lack of sparsity promotting algorithms that are able to enforce both the non-negativity and the sum to one constraint.

A major issue in both the theoretical and experimental compressive sensing community is a lack of code design schemes. For the most part random measurements techniques are dominant because they obey well known theoretical results which guarantee reconstruction with high probability [31, 38, 39]. However, designed codes have been shown to outperform random codes in various applications [3, 33, 40, 41, 42]. Intuitively, designed codes which can take into account prior knowledge of the physical limitations of the sensing task and additional statistical assumptions of the signal-of-interest should be able to outperform random measurements. For example, I will demonstrate in the AFSSI-C that adaptively designed Principal Component Analysis (PCA) codes dramatically outperform random codes in low SNR environments.

These issues and others will be discussed in a case study manner throughout this

dissertation. I will now discuss the organization of this dissertation and the three separate computational sensors I have built and how I have tried to mitigate and resolve some of the practical issues associated with computational sensing.

1.8 Dissertation Overview

I used this chapter to cover the major concepts of computational sensing. More importantly, I provided the major topic of discussion for this dissertation, the challenges that remain towards developing practical computational sensors. I will use Chapter 2 as an opportunity to provide a more detailed and mathematically rigorous look at the various coding schemes that are popular in computational sensing, as well as the ones I have used. This includes a discussion of the mathematical methods that I and my collaborators developed and deployed in the AFSSI-C: Bayes rule, Log-Likelihood Ratios, and the Maximum a Posteriori decision technique. A more rigorous treatment of compressive sensing will also be provided which includes several important results from the compressive sensing community. I will also discuss several estimation and task-specific sensing algorithms that I used during the development of the three computational sensors in my work.

Chapter 3 will introduce a new system architecture for compressive target tracking—the Static Computational Optical Undersampled Tracker (SCOUT). This system is designed to overcome a variety of challenges that typically arise in traditional optical sensing approaches. It provides several advantages over the single-pixel camera architecture, notably it can capture many simultaneous encodings of the field-of-view with a single camera exposure. I will present experimental results that validate the performance of the proposed architecture.

Chapter 4 will discuss an important experimental prototype which demonstrates adaptive spectral image classification. The system is a major experimental advancement compared to current computational spectral imaging architectures which focus on reconstruction. Furthermore, the ability to perform adaptive measurements, where each code is designed based on the history of prior measurement data allows this instrument to outperform traditional spectral imaging architectures by a factor of 100 in low SNR scenarios.

Chapter 5 will discuss current efforts to experimentally demonstrate compressive

spectral unmixing. Often in spectral imaging, the spectra present in the field-of-view of each pixel is actually a mixture of several spectra, a mixed spectrum. Spectral unmixing is the task of inferring the fraction of each constituent spectrum in the mixed spectrum. I employ a unique architecture for computational spectral sensing which uses no diffraction gratings or refracting prisms but relies on the wavelength dependent nature of the birefringence in an Liquid Crystal on Silicon (LCOS). I will demonstrate the effectiveness of various compressive sensing measurement schemes to outperform a traditional sensor in significantly fewer measurements.

Chapter 6 will summarize the dissertation and provide my perspective on how the field should the approach the issues in practical computational optical sensing.

Chapter 2

Formalism

I will use this chapter to cover several important mathematical concepts which are required for the reader to understand the advantages and disadvantages of computational sensing. I will first discuss the advantages that Hadamard and S-Matrix multiplexing provide compared to the isomorphic sensing to create a context for the discussion of the Fellgett advantage. Then I will discuss Principal Component Analysis (PCA), a dimensionality reduction technique. I will also cover the basics of Bayesian statistics and how it can be used to update one's belief in a hypothesis given new data. Then will I cover important topics in compressive sensing: sparsity, incoherence, and the restricted isometry property (RIP) and discuss an optimization problem that allows one to reconstruct sparse signals from compressively sampled data.

To simplify the math and notation, I shall try to work with discrete representations of signals when possible. Any discrete linear measurement process can be written as a matrix multiplication

$$\mathbf{g} = \mathbf{H}\mathbf{f} + \mathbf{e} \tag{2.1}$$

where $\mathbf{g}$ is a vector of measurement data and $\mathbf{f}$ is a discrete representation of the object signal-of-interest and $\mathbf{H}$ is the matrix which describes the measurement process and $\mathbf{e}$ is additive noise. For brevity I will refer to $\mathbf{f}$ as the object and $\mathbf{H}$ as either the sensing matrix or the measurement matrix. Equation (2.1) represents the forward model in a wide variety of computational sensors. The object $\mathbf{f}$ is a vector in N dimensional vector space and the measurement $\mathbf{g}$ is a vector in N_m dimensional vector space. In general $N_m \neq N$.

Equation (2.1) is an extremely useful way to represent optical phenomena, since it allows one to take advantage of many attractive numerical techniques which are dedicated to linear systems. The ray and wave description of light can described using the mathematics of linear systems. The solutions to the paraxial wave equation are linear in free space and in most dielectrics. Similarly, paraxial optics can be modeled using a sequence of matrix multiplications. While just an approximation, tracing the paraxial chief and marginal ray is enough to calculate the third-order Seidel aberrations.

2.1 Isomorphic Sensing

In an isomorphic measurement, where the goal is a one-to-one mapping of object points to measurement points, the measurement matrix is represented with the identity matrix

$$\mathbf{H} = \mathbf{I} \tag{2.2}$$

One can get an idea of how much error exists in an isomorphic measurement by invoking the weighing example [43]. Say there are 4 objects with true unknown weights f_1, f_2, f_3, f_4. The measured weights are g_1, g_2, g_3, g_4 with additive noise e_1, e_2, e_3, e_4.

$$\begin{bmatrix} g_1 \\ g_2 \\ g_3 \\ g_4 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \end{bmatrix} + \begin{bmatrix} e_1 \\ e_2 \\ e_3 \\ e_4 \end{bmatrix} \tag{2.3}$$

Since this is isomorphic sensing, the estimated weights $\hat{\mathbf{f}}$ are the measurements $\mathbf{g}$

$$\hat{\mathbf{f}} = \mathbf{g}. \tag{2.4}$$

Where the error between the estimated weights and the actual weights is $\epsilon = \hat{\mathbf{f}} - \mathbf{f}$. For simplicity, I assume a zero mean distribution for the noise. Therefore, the isomorphic measurements are unbiased.[1] For an unbiased estimator, the Mean Squared Error (MSE) of

$$E[\epsilon^2] = E[(\hat{f}_m - f_m)^2] = \sigma^2. \tag{2.5}$$

This puts a lower bound on the MSE for an isomorphic measurement.

[1] In statistics, the bias of an estimator is the difference between this estimator's expected value and the true value of the parameter being estimated. An estimator with zero bias is called unbiased.

2.2 Multiplexing

As I discussed in Chapter 1, multiplexing is a technique for overcoming SNR related trade-offs in isomorphic sensing. In a multiplexed measurement, each element in the measurement vector $\mathbf{g}$ is a weighted linear combination of the elements in the object vector. Therefore the measurement matrix $\mathbf{H}$ is no longer an identity matrix.

2.2.1 Coding Schemes

A Hadamard matrix of order N is defined as a matrix $\mathbf{H}_N$ as the $N \times N$ matrix whose elements consist of $+1$'s and -1's and satisfies the following property:

$$\mathbf{H}_N^T \mathbf{H}_N = \mathbf{H}_N \mathbf{H}_N^T = N\mathbf{I}_N \tag{2.6}$$

where $\mathbf{I}_N$ is an $N \times N$ identity matrix.

In computational spectroscopy and imaging, Hadamard codes are extremely popular. *Hadamard codes are provably optimal in the case where one is allowed to take a full set of measurements with additive noise*, meaning that $\mathbf{f}$ and $\mathbf{g}$ from Equation (2.1) are both vectors in N dimensional space [43]. Even in the presences of additive noise, $\mathbf{e}$

A Hadamard multiplexed measurement is written as

$$\begin{bmatrix} g_1 \\ g_2 \\ g_3 \\ g_4 \end{bmatrix} = \begin{bmatrix} +1+1+1+1 \\ +1-1+1+1 \\ +1-1-1+1 \\ +1-1-1+1 \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \end{bmatrix} \tag{2.7}$$

In the weighing example, negative measurements are realized by using a balancing scale instead of a spring scale. This means that in the first measurement all 4 items are placed in the same pan. In the second measurement items 1 and 3 are in the same pan while items 2 and 4 are in the opposite pan, and so on [43]. With four equations and four unknowns, one can solve for the estimates using basic algebra.

The MSE of the m^{th} measurement

$$E[(\hat{f}_m - f_m)^2] = \frac{1}{4}\sigma^2. \tag{2.8}$$

which is 4 times lower than using an isomorphic measurement scheme. In general, the MSE of a Hadamard measurement is

$$\text{MSE} = \frac{\sigma^2}{N} \tag{2.9}$$

where N is the dimension of the object vector $\mathbf{f}$, which is equal to the number of measurements N_m. Hotelling proved in 1944 that for a measurement matrix with elements $h_{mn} \in \{-1, +1\}$, the lowest possible MSE for the case of a full set of measurements with a linear unbiased estimator is $\text{MSE} = \sigma^2/N$ [1]. Therefore, one cannot possibly do better than Hadamard coding in this case.

In many practical cases in computational sensing, making a code with simultaneous positive and negative modulation of the signal is not possible. While this may be possible using coherent light where a phase delay maybe used to produce a negative electric field. For incoherent light, the system response is linear in intensity [44].

In the case where one has the ability to make a full set of measurements but is limited to elements of +1's and 0's, an S-Matrix code minimizes the MSE [43]. In the weighing example, a spring balance rather than a two pan scale is analogous to this situation.

$$\begin{bmatrix} g_1 \\ g_2 \\ g_3 \\ g_4 \end{bmatrix} = \begin{bmatrix} 0 & +1 & +1 & +1 \\ +1 & +1 & 0 & 0 \\ +1 & 0 & +1 & 0 \\ +1 & 0 & 0 & +1 \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \end{bmatrix} \tag{2.10}$$

So items 2, 3, and 4 are weighed together, then 1 and 2, and so on. Solving the system of equations in a silimar fashion as before in the Hadamard weighing example we find that the mean square error for the m^{th} measurement when weighing 4 items is

$$\text{MSE} = E[(\hat{f}_m - f_m)^2] = \frac{7}{9}\sigma^2. \tag{2.11}$$

Which reduced the MSE compared to the isomorphic measurement scheme but has higher MSE compared to the Hadamard measurement scheme. The MSE of the S-Matrix is approximately a factor of 4 increase compared to the Hadamard matrix coding scheme.

In the case when the full set of measurements are available ($N_m = N$), I have just shown that Hadamard codes are optimal. It would seem as if random coding

would have no use if one is able to utilize Hadamard codes. However, they should not be ignored because in compressive sensing, certain theoretical guarantees exist for random coding that do not exist for Hadamard and S-Matrix codes. Sometimes, the physics of the situation forces a random coding scheme.

2.2.2 The Fellgett Advantage

The *Fellgett advantage* is the improvement in SNR that occurs when an instrument takes multiplexed measurements compared to isomorphic measurements [45, 46]. Physically the Fellgett advantage occurs because a single detector element produces a noise contribution whether it is sampling a single element of the signal or the sum of multiple elements of the signal. Maximizing the signal-to-noise ratio (SNR) of the estimated object signal-of-interest for a given system throughput and detector noise is major design consideration in computational optical sensing particularly in the area of spectroscopy. There are two notable ways to accomplish multiplexing in spectroscopy: Hadamard multiplexing in dispersive spectrometers and interferometric based multiplexing such as the Fourier Transform Spectrometer (FTS).

The FTS architecture is a similar to the Michelson interferometer, see Figure 2.1. The FTS operates by taking the autocorrelation of the complex electric field as a function of time delay by moving one of the mirrors in the interferometer [46]. The Wiener-Khinchin theorem says that for a wide-sense stationary random process, the Fourier transform of the autocorrelation is the power spectral density [47]. Thus a computational post-processing step is required to reconstruct the spectrum from the measured autocorrelation data. Since the FTS measures combinations of multiple wavelengths at each detector readout, it also exhibits the Fellgett advantage. It turns out that in a FTS the MSE obtained is a factor of two greater than the Hadamard multiplexing spectrometer [48].

$$\text{MSE} = 2\frac{\sigma^2}{N_\lambda} \tag{2.12}$$

where N_λ is the number of spectral channels.

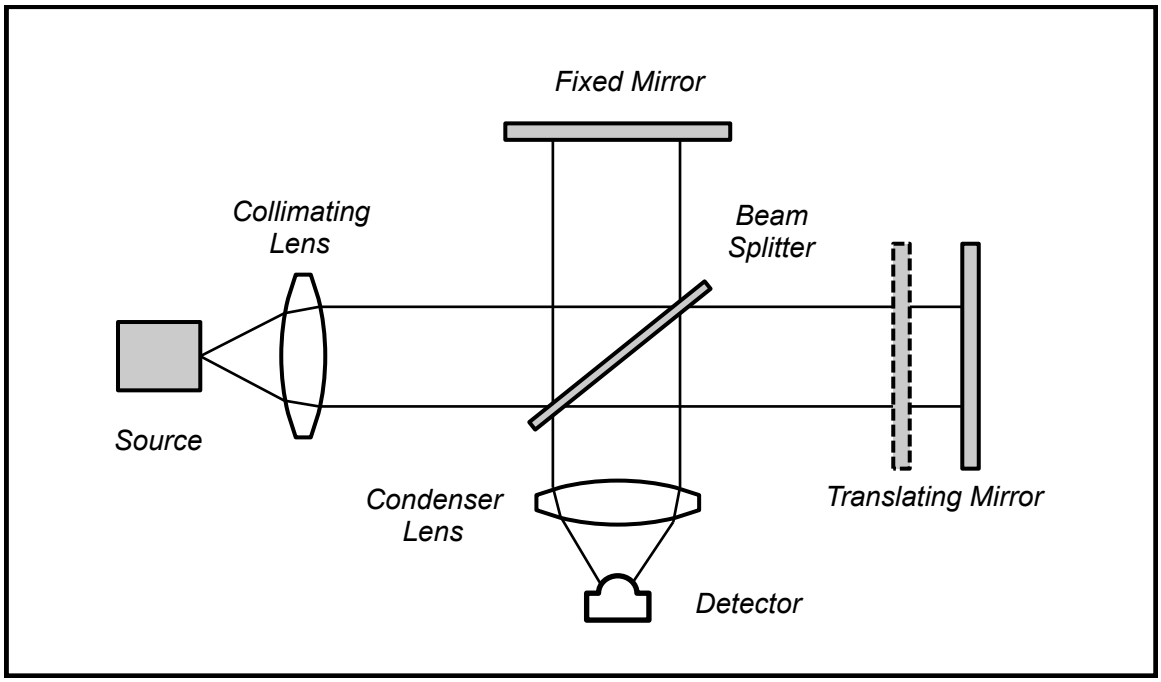

Figure 2.1: The architecture of the Fourier Transform Spectrometer resembles the Michelson interferometer. One of the mirrors is translated back and forth. The interferogram is the detected intensity versus mirror delay which is related to the autocorrelation of signal. The Fourier transform of the autocorrelation provides the spectrum.

2.3 Principal Component Analysis

Principal component analysis (PCA) is a dimensionality reduction technique that attempts to recast a dataset in a manner that finds correlations in data that may not be evident in their native basis and creates a set of basis vectors in which the data has a low dimensional representation. PCA works by producing a set of vectors that point along the direction of maximum variance while being simultaneously orthogonal to each other. These are called the *principal components*. The first principal component vector is parallel to the direction of maximum variance. The second principal component points in the direction of maximum variance that is not explained by the first principal component. And so on.

Imagine one has a dataset $\mathbf{S}$ which consists of N spectra with N_λ spectral channels. Instead of looking at the data as just intensity versus spectral channel, PCA attempts to construct a set of new vectors (also called features) that show as much variation in the spectra as possible. In other words, first direction (principal component) is used to recast the data to look as different (uncorrelated) as possible.

This allows us to discriminate the data, as best we can with just one direction. The second principal component is the direction that provides the second most ability to discriminate the data, and so on.

Now I will discuss some of the math behind PCA. The covariance matrix is defined as

$$\mathbf{C_S} = \frac{1}{N}\mathbf{S}\mathbf{S}^T. \tag{2.13}$$

Each element in the covariance matrix $C_{\mathbf{S}mn}$ is the covariance of the m^{th} spectrum $\mathbf{s}_m$ and the n^{th} spectrum $\mathbf{s}_n$.

$$C_{\mathbf{S}mn} = \frac{1}{N}\mathbf{s}_m\mathbf{s}_n^T. \tag{2.14}$$

A large covariance means they look alike and therefore are difficult to disambiguate. Geometrically, it means that $\mathbf{s}_m$ and $\mathbf{s}_n$ tend to point in the same direction (assuming their length is approximately the same). If the entire collection of spectra $\mathbf{S}$ were mutually orthogonal, being able to tell one spectrum apart from another would be easy. You would just have a collection of spikes at different spectral channels. The covariance matrix in this case would be a diagonal matrix. Therefore, it is desirable to have covariance matrices that are diagonal matrices since they indicate low correlation between the datasets and improves one's ability to distinguish between the data.

Since there is typically some redundancy between spectra, the off-diagonal elements of the covariance matrix will be non-zero. PCA finds a basis in which the covariance matrix is diagonalized. In this basis, the data is uncorrelated.

$$\mathbf{Y} = \mathbf{P}\mathbf{S} \tag{2.15}$$

where $\mathbf{Y}$ is the data projected onto the principal component basis $\mathbf{P}$. The *rows* of matrix $\mathbf{P}$ are the principal components. The covariance of the projected data

$$\mathbf{C_Y} = \frac{1}{N}\mathbf{Y}\mathbf{Y}^T \tag{2.16}$$

is a now diagonal matrix. Indeed, the principal components are the optimal way to discriminate the spectra in the dataset [49]. It turns out the principal components are the *eigenvectors of the covariance matrix* $\mathbf{C_S}$ *are the principal components* [50].

Note that each eigenvector has an associated eigenvalue. The eigenvalues are also informative because the relative magnitude of the eigenvalue can tell us how

many principal components are really needed to discriminate all of the spectra [51]. The magnitude of the eigenvalue tells us how well useful the associated eigenvector is at discriminating the data.

Since the full set of principal components forms a basis, each spectra $\mathbf{s}$ in $\mathbf{S}$ can be written as a superposition of principal components without any error

$$\mathbf{s} = \sum_{n=1}^{N_\lambda} \alpha_n \mathbf{p}_\lambda \tag{2.17}$$

Where α_n is a scalar coefficient value associated with eigenvector $\mathbf{p}_n$. In many cases, only a few of the first principal components are needed in the summation to approximate the original data well, $M \ll N_\lambda$.

This is another reason why PCA is so useful. It can be used as a way to perform dimensionality reduction. In other words, seemingly complicated data can be summarized by only a few principal components by exploiting the correlations between the data. This concept is analogous to lossy compression in signal processing. Simply project the data onto the first several principal components associated with the largest M eigenvalues. The data now has an approximately sparse representation.

I will show in Chapter 4, that the AFSSI-C relies on a variation of Principal Component Analysis (PCA) for discriminating between spectra. In addition to PCA, the AFSSI-C uses a Bayesian technique to create adaptive codes. I will now discuss some of fundamentals of Bayesian statistics.

2.4 Bayesian Statistics

A hypothesis is nothing more than a claim or premise that one is interested in verifying. In imaging and spectroscopy, one example is that at a certain location in the field of view, the hypothesis is that a spectrum is present. Another hypothesis is that the mean value of the signal is some value. Often times one is interested in estimating parameters of stochastic processes, which we denote θ.

Bayesian statistics allows one to treat the hypothesis or parameters as random variables rather than deterministic constants. At the heart of Bayesian approaches is Bayes' theorem, which is a way of computing probabilities of a hypothesis given some evidence which are related to the hypothesis. The idea is that one can make

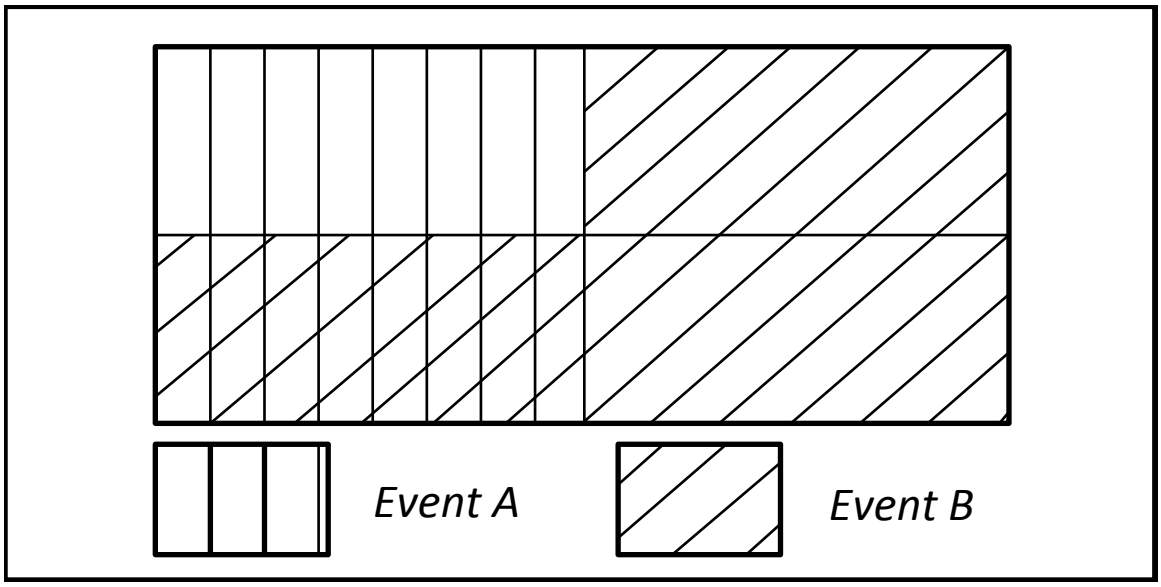

Figure 2.2: Graphical demonstration of joint probability. The probability of event A is $P(A) = 3/4$. The probability of event B is $P(B) = 3/4$. The joint probability of events A and B is $P(A \cap B) = 1/4$. The probability of event A occuring given that event B has occured is $P(A \,|\, B) = 1/3$. This is consistent with the equation $P(A \cap B) = P(A \,|\, B)P(B)$

a more informed calculation of probability if one is able to update the probability given some new piece of evidence that one may have not had at the beginning.

The derivation of Bayes' theorem follows from the definition of conditional probability. The conditional probability of event A occurring given that B occurred is:

$$P\left(A \,\middle|\, B\right) = \frac{P\left(A \cap B\right)}{P\left(B\right)} \tag{2.18}$$

this can be seen graphically in Figure 2.2. Solving for the joint probability $P\left(A \cap B\right)$ gives

$$P\left(A \cap B\right) = P\left(A \,\middle|\, B\right) P\left(B\right) \tag{2.19}$$

since the joint probability commutes $P\left(A \cap B\right) = P\left(B \cap A\right)$, we can also write

$$P\left(A \cap B\right) = P\left(B \,\middle|\, A\right) P\left(A\right) \tag{2.20}$$

equating the right hand sides of Equation 2.19 and Equation 2.20 gives use Bayes' theorem (also called Bayes' rule)

$$P\left(A \,\middle|\, B\right) = \frac{P\left(A\right) P\left(B \,\middle|\, A\right)}{P\left(B\right)} \tag{2.21}$$

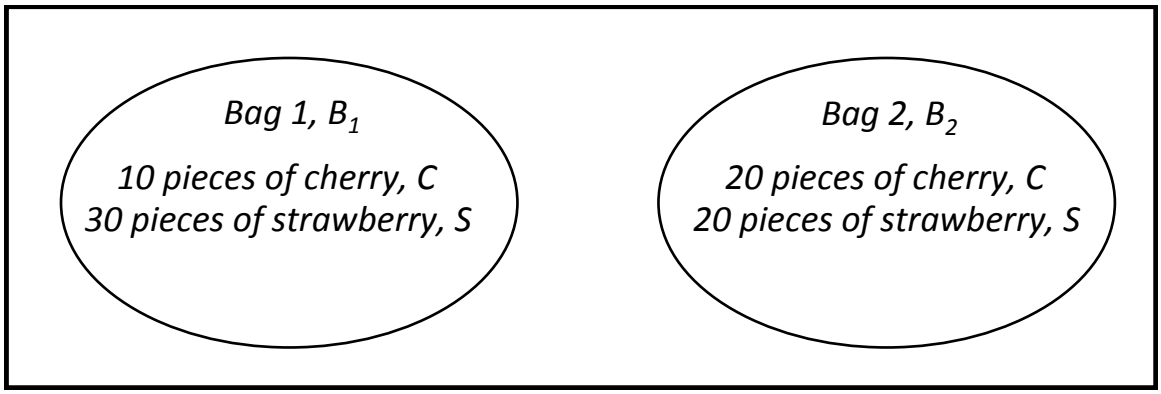

Figure 2.3: Two bags are present: B_1 and B_2. The probability of choosing either bag is $P(B_1) = P(B_2) = 1/2$. Bag 1 has 40 pieces of candy: 10 pieces of cherry candy so $P(C|B_1) = 1/4$ and 30 pieces of strawberry so $P(S|B_1) = 3/4$. Bag 2 has 40 pieces of candy: 20 pieces of cherry candy so $P(C|B_2) = 2/4$ and 20 pieces of strawberry so $P(S|B_2) = 2/4$. If someone selects one of the bags at random then selects a piece of candy from that bag. If the candy is identified but the bag is not, then one can use Bayes' theorem to update the probability of either bag being selected.

One interpretation of Bayes' theorem is called the Diachronic interpretation, which says that conditional probability of the hypothesis or parameter given knowledge of some evidence or measurement data is given by

$$P\left(\theta \mid g\right) = \frac{P\left(\theta\right) P\left(g \mid \theta\right)}{P\left(g\right)} \tag{2.22}$$

The term $P\left(\theta \mid g\right)$ is called the *posterior*. It represents the belief in the hypothesis given the data. The term $P\left(\theta\right)$ is called the *prior*. $P\left(g \mid \theta\right)$ is called the *likelihood*. $P\left(\theta\right)$ is called the normalizing constant, which is computed to ensure that the posterior probabilities sum to 1. The normalizing constant can be written as

$$P\left(\theta\right) = \sum_i P\left(\theta_i\right) P\left(g_i \mid \theta_i\right) \tag{2.23}$$

One can repeatedly apply Bayes' theorem given new measurement data.

2.4.1 Example: Updating Probabilities with Bayes' Theorem

I will use a simple example to illustrate how to update probabilities using Bayes' theorem, see Figure 2.3. Imagine I have two bags of candy, Bag 1, which I denote B_1, and Bag 2, which I denote B_2. Bag 1 has 10 pieces of cherry flavored candy,

denoted as C, and has 30 pieces of strawberry flavored candy, denoted as S. Bag 2, B_2, has 20 pieces of cherry candy and 20 pieces of strawberry candy. At the beginning, the prior probability of selecting bag 1 or bag 2 is both equal

$$P\left(B_1\right) = P\left(B_2\right) = \frac{1}{2} \tag{2.24}$$

Someone then picks a bag at random and takes out a piece of candy that turns out to be strawberry flavor. They do not state which bag was selected, only that the candy they randomly selected from the bag was strawberry. What is the probability that bag 1 was the one selected given that I know the candy is strawberry? I can use Bayes' theorem to compute this

$$P\left(B_1 \mid S\right) = \frac{P\left(B_1\right) P\left(S \mid B_1\right)}{P\left(S\right)} \tag{2.25}$$

$P\left(S \mid B_1\right)$ is the probability of selecting strawberry given that they pick bag 1, which is 3/4. $P\left(S\right)$ is the probability of selecting a strawberry candy from either bag 1 or bag 2, which 5/8. Thus

$$P\left(B_1 \mid S\right) = \frac{\left(\frac{1}{2}\right)\left(\frac{3}{4}\right)}{\left(\frac{5}{8}\right)} = \frac{3}{5} \tag{2.26}$$

Similarly, the probability that the person chose bag 2 is $P\left(B_2 \mid S\right) = 2/5$. Qualitatively, this makes sense, since bag 1 contained more strawberry flavored candy, the probability that bag 1 was chosen should increase since it has more strawberry candy and the probability that bag 2 was chosen should decrease.

Now to continue the example. Imagine the first piece of candy is put back into the bag from where it came from. Then another piece of candy is drawn from the same bag, which turns out to be cherry flavor. Now one must update the probabilities with this new piece of information. One can keep using Bayes' theorem. The posteriors from the last draw $m-1$ are now used as the priors for the current update.

$$P\left(B_1 \mid C\right)_m = \frac{P\left(C \mid B_1\right) P\left(B_1 \mid S\right)_{m-1}}{P\left(C\right)} \tag{2.27}$$

$$P\left(B_2 \mid C\right)_m = \frac{P\left(C \mid B_2\right) P\left(B_2 \mid S\right)_{m-1}}{P\left(C\right)} \tag{2.28}$$

The probability of drawing a cherry flavored candy assuming bag 1 was chosen remains constant since the ratio of cherry to strawberry did not change for either bag, so $P\left(C \mid B_1\right) = 1/4$ and the probability of drawing a cherry flavored candy assuming bag 2 was chosen is 1/2. We now must use Equation 2.23 to compute the normalizing constant, otherwise the posterior probabilities will not sum to 1. In this case $P(C) = 7/20$. Plugging these numbers into Equations 2.27 and 2.28 gives the updated posterior probabilities

$$P\left(B_1 \mid C\right)_m = \frac{3}{7} \approx 0.43 \tag{2.29}$$

$$P\left(B_2 \mid C\right)_m = \frac{4}{7} \approx 0.57 \tag{2.30}$$

Intuitively, drawing a cherry flavored candy has reduced our belief that bag 1 was chosen since bag 1 consist of only 1/4 cherry flavor candy while bag 2 consisted of 1/2 cherry candy. This sequence can be continued, until a threshold has been reached for one of the posterior probabilities.

2.4.2 Maximum A Posteriori

Imagine a situation similar to the candy example, where we are given a set of hypotheses, $\{h_1, h_2, \cdots, h_{N_R}\}$, and we are interested in finding which hypothesis is the most likely, after new measurement g_m is made. The method of Maximum A Posteriori (MAP) says the hypothesis h_k which maximizes the posterior probability is the most likely one.

$$\hat{\mathbf{h}}_{map} = \arg\min_k \; p\left(h_k \mid g\right) \tag{2.31}$$

Using Bayes' theorem I can rewrite Equation 2.31 as

$$\hat{\mathbf{h}}_{map} = \arg\min_k \; \frac{p\left(g \mid h_k\right) p\left(h_k\right)}{p\left(g\right)} \tag{2.32}$$

Maximizing the posterior is now equal to maximizing the likelihood and prior. In certain cases, one needs to decide between two sets of parameters or hypotheses. One can do an analogous technique of comparing the posteriors by using a ratio

$$\frac{p(h_i|g)}{p(h_j|g)} = \frac{p(g|h_i)}{p(g|h_j)} \frac{p(h_i)}{p(h_j)} \tag{2.33}$$

If the ratio is larger than some threshold value then one choses parameter h_i and if the ratio is less than the threshold then one choses h_j. Similar to the earlier example of updating the posterior based on new data, one can update the Maximum A Posteriori (MAP) decision based on new data. Define the likelihood ratio of the m^{th} measurement as

$$\Lambda_m = \frac{p(g_m|h_i)}{p(g_m|h_j)} \tag{2.34}$$

After each new set of measurement data g_m is collected, one can update the posterior ratios by multiplying the likelihood ratio from the old set of data with the likelihood ratio of the new set of data. The ratio which includes all the updates from measurement $m = 1$ to measurement $m = N_m$ is written as

$$\frac{p(h_i|\{g\}_{N_m})}{p(h_j|\{g\}_{N_m})} = \prod_{m=1}^{N_m} \Lambda_m \frac{p(h_i)}{p(h_j)} \tag{2.35}$$

where the notation $\{g\}_{N_m}$ represents the set of all data from measurement m to N_m.

In summary, Bayesian statistics is a useful way to update one's belief in a hypothesis or estimate a set of parameters. Bayes' theorem can be used to merge new measurement data and the probability of a hypothesis before the data was known. This is very similar the approach taken by the Adaptive Feature Specific Spectrometer (AFSS) and the AFSSI-C [3, 40].

2.5 Compressive Sensing

The fundamental approach of compressive sensing is that rather than sampling at a high rate and then compressing the sampled data, one can dramatically reduce the number of samples if each sample is in a compressed form. In Chapter 1, I gave some intuitive understanding of how compressive sensing works, but there are certain mathematical concepts that the reader should understand in order to have a full appreciation of the practical challenges in computational sensing. So, I will now discuss some of the formalism of compressive sensing and some techniques to compressively sampling signals. In order to understand why compressive sensing is so powerful I will first discuss the conventional sampling strategy.

2.5.1 The Nyquist-Shannon Sampling Theorem

One of the most important results concerning the *sampling* of continuous signals is the Nyquist-Shannon sampling theorem (often referred to as the *sampling theorem* for short). The sampling theorem says that exact reconstruction of a continuous *bandlimited signal* $f(x)$ is possible if the sampling frequency f_s is atleast twice the maximum frequency B of the signal [52]. Assuming that a bandlimited signal $f(x)$ has been sampled according to the sampling theorem, then exact recovery from the discrete samples f_n is guaranteed.

However, if the sampling frequency is less than twice the maximum bandwidth $f_s < 2B$, then aliasing may occur in the reconstruction. Aliasing is the effect that high frequencies in the original signal will be represented as lower frequencies after reconstruction and information contained in the high frequencies will be potentially lost [53].

Given a spatial length l, the number of measurement samples is

$$N_m = f_s l \tag{2.36}$$

Since f_s must be at least twice the maximum frequency of the signal there is a lower bound on the number of samples needed to recover the signal with no loss of information

$$N_m \geq 2l \cdot \max \left| F\left\{ f\left(x\right) \right\} \right| \tag{2.37}$$

where $F\left\{ f\left(x\right) \right\}$ is the Fourier Transform of the signal $f(x)$.

It is important to clarify a small but important distinction between the meaning of sampling and the meaning of a measurement. A measurement is any process that maps physical phenomena which contains a signal-of-interest into measurement data. The measurement data may or may not be discrete. Sampling has a more precise mathematical definition. It is the process of mapping a continuous signal into a sequence of discrete numbers which are called the samples.

2.5.2 Sparsity, Incoherence, and the Restricted Isometry Property

There are two definitions of compressive sensing: The definition based on sparsity, incoherence, and the restricted isometry property (RIP) and the practical definition.

The *sparsity-incoherence definition* asserts that true compressive sensing is when the number of measurements is much less than the dimensionality of the signal, $N_m \ll N$, and certain properties called sparsity, incoherence, and RIP are satisfied. Under this definition, in the absence of noise, several theorems guarantee high probability of exact reconstruction given a specific type of optimization algorithm. Under this framework, random measurement codes tend to have theoretically attractive properties that make them ideal for compressive sensing [54, 29, 36, 38].

The *practical definition* asserts that compressive sensing occurs anytime the number of elements in the measurement is much less than the dimensionality of the original signal and the signal is reconstructed with a small amount of error.

At first glance, compressive sensing seems to go against the Nyquist-Shannon sampling theorem, however the sampling theorem's guarantee of exact reconstruction of a continuous signal relies on the assumption of a bandlimited signal and uniform periodic sampling.

Any continuous signal $f(x)$ can be written as a discrete summation of orthonormal basis functions

$$f(x) = \sum_{n=1}^{N} x_n \Psi_n(x), \tag{2.38}$$

where x_n are the coefficients. I will call the vector $\mathbf{x} = [x_1\, x_2\, \ldots\, x_N]^T$ the representation vector of the signal and $\mathbf{\Psi}$ the representation basis. This allows us to rewrite the signal as a matrix multiplication

$$\mathbf{f} = \mathbf{\Psi x}. \tag{2.39}$$

As I discussed in Chapter 1, any vector $\mathbf{x}$ is sparse when all but a few of its entries are zero. A vector is called K-sparse when it has at most K non-zero entries.

$$\mathbf{x} : \|\mathbf{x}\|_0 \leq K \tag{2.40}$$

In real situations, signals with strictly sparse representation vectors are unlikely. Fortunately, it is possible to have approximately sparse representation vectors, which are called *compressible*. In other words, the sorted magnitudes of the coefficients $|x_n|$ quickly decay. When a signal has an expansion in terms of a compressible representation vector, we can intuitively understand Equation (2.38). Discarding

smaller coefficients will not significantly degrade the information in the signal [38].[2]

The concept of *sparsity* is important in compressive sensing. Sparsity determines how efficiently one can acquire the signals. All things being equal, if K decreases, then the probability of achieving exact recovery increases. Sparse representations of the signal are not the only important prerequisite for high probability reconstruction.

In practice, the signal is sampled in a different basis than the representation basis. For example, while many natural signals have a sparse or compressible representation in various wavelet bases or Fourier bases, sampling using the representation basis is not practical in many cases. Often a sensing basis $\mathbf{H}$ is used to perform the sampling. In traditional Nyquist-Shannon sampling, the sensing basis is a collection of delta functions. In many compressive imaging experiments, the sensing basis is the binary random coded aperture mask [11]. In an LCOS based compressive sensing spectrometer, the sensing basis is a finite set of spectral filters [55, 56]. In short, the representation basis allows a signal to be represented as a sparse vector, while the sensing basis is composed of the functionals that one samples with. The equation which combines these concepts to model a compressive measurement is

$$\mathbf{g} = \mathbf{H}\mathbf{\Psi}\mathbf{x} = \mathbf{H}\mathbf{f} \tag{2.41}$$

where $\mathbf{g}$ is a $N_m \times 1$ measurement vector, $\mathbf{H}$ is a $N_m \times N$ measurement vector, $\mathbf{\Psi}$ is a $N \times N$ matrix and $\mathbf{x}$ is an $N \times 1$ vector.

One important concept in compressive sensing is coherence. The coherence between the sensing basis $\mathbf{H}$ and the representation basis $\mathbf{\Psi}$ is

$$\mu\left(\mathbf{H}, \mathbf{\Psi}\right) = \sqrt{n} \max_{1 \leq k, 1 \leq n} |\langle h_k, \psi_j \rangle|, \tag{2.42}$$

which defines coherence as a measure of the largest correlation between any two columns of $\mathbf{H}$ and $\mathbf{\Psi}$.

Ideally, one would like to use as few measurements as possible without degrading the reconstructed signal. In compressive sensing, the amount of measurements needed is a function of the sparsity and the coherence. Given a coefficient sequence $\mathbf{x}$ that is K-sparse then one needs

$$N_m \geq C \cdot \mu^2(\mathbf{H}, \mathbf{\Psi}) \cdot K \cdot \log n \tag{2.43}$$

[2]From now on, I will use the words sparse and compressible interchangeably. But it is important to realize there is a difference.

for a high probability of reconstruction, where C is just a constant and N is the dimensionality of the original signal [38].

Equation (2.43) shows the importance of sparsity and coherence in compressive sensing. Lower coherence and sparsity allows one to use fewer measurements [11]. The more incoherent, and therefore lower correlation, the two bases are, the higher the probability of succesful reconstruction of compressive measurements. It turns out that random matrices have a high probability of being incoherent with any basis [38].

The isometery constant δ_K of a matrix $\mathbf{A}$ is the smallest number such that

$$(1 - \delta_K) \|\mathbf{x}\|_{\ell_2}^2 \leq \|\mathbf{A}\mathbf{x}\|_{\ell_2}^2 \leq (1 + \delta_K) \|\mathbf{x}\|_{\ell_2}^2 \tag{2.44}$$

for all K-sparse vectors $\mathbf{x}$. If δ_K is not too close to one then the matrix $\mathbf{A}$ obeys the *restricted isometry property* (RIP) of order K [38]. If RIP is satisfied, the matrix $\mathbf{A}$ approximately preserves the Euclidean length of signals. The RIP is an important theoretical result which allows robust compressive sensing when signals are corrupted by noise. Sensing matrices which have random entries obey the RIP with high probability [38, 11, 36].

All of the concepts I have just discussed can be understood at an intuitive level. Sparsity is the idea that the information content of a signal is relatively small compared to the amount of data which originally described the signal. Coherence extends the concept of how invertible a matrix is, the more linearly independent the system of equations, the easier it is to invert the matrix. The restricted isometry property is basically saying that any matrix with a small isometry constant will keep the distance between signal vectors the same. Why is this important? Think of a geometric picture, imagine noise as a sphere of uncertainty around the signal vectors. When the noise is small, two signal vectors can be mapped to a small part of the measurement space and still fit in that space. When one extracts the signal from the compressed measurements, one can tell them apart. As the noise increases one wants the distance between measured signal vectors to at least stay the same and certainly not dramatically decrease, otherwise it will be difficult to tell the signals apart. The RIP is basically a way of seeing if a measurement matrix will pack the signal vectors with the same distance between them. This is important because one does not want a small amount of noise to result in a large reconstruction error.

2.5.3 Solving Inverse Problems For Compressive Sensing

Now that I have discussed what compressive sensing is mathematically, it is time to address how to actually solve the problem of extracting $\mathbf{x}$ from

$$\mathbf{g} = \mathbf{A}\mathbf{x} + \mathbf{e} \tag{2.45}$$

where $\mathbf{g}$ is the measurement vector, $\mathbf{x}$ is the sparse representation of the signal, $\mathbf{A}$ is the linear mapping from $\mathbf{x}$ to $\mathbf{g}$, $\mathbf{e}$ is the noise vector, and the number of measurements N_m is much less than the dimensionality of the sparse representation vector N.

The least squares (LS) estimator attempts to solve this inverse problem by minimizing the objective function

$$\sum_{n=1}^{N} (\mathbf{g} - \mathbf{A}\mathbf{x})^2 = \|\mathbf{g} - \mathbf{A}\mathbf{x}\|_2^2. \tag{2.46}$$

which is the ℓ_2 norm between the given data $\mathbf{g}$ and the forward model for the data $\mathbf{A}\mathbf{x}$. Notice, there are no probabilistic assumptions about the measurement data. It turns out that a closed form version of the LS estimator exists as

$$\hat{\mathbf{x}} = \left(\mathbf{A}^T\mathbf{A}\right)^{-1}\mathbf{A}^T\mathbf{g}. \tag{2.47}$$

where $\hat{\mathbf{x}}$ is the estimated value of $\mathbf{x}$.[3]

The derivation of the closed form of the least squares estimator is given in Appendix A. Alternatively, if the vector $\mathbf{x}$ is very large, solving the inverse problem in closed form maybe too computationally intensive. In this case, one may use the gradient descent algorithm or other types of iterative optimization algorithms to solve the least squares problem. While the LS estimator may provide a solution when $N_m \ll N$, these solutions tend to overfit the data in these situations, do not take advantage of the prior knowledge of sparsity to reduce the solution space, and $\mathbf{A}\hat{\mathbf{x}}$ is not unique. Therefore, an unconstrained LS approach does not work well for the compressive sensing problem.

[3]Note that it is important to realize that in practice one should never use the actual inverse of $\mathbf{A}^T\mathbf{A}$ due to the various numerical computing issues. One should resort to computing the psuedoinverse.

There are a vast number of algorithms that are designed for compressive sensing and new ones are being published constantly. A full discussion of each algorithm is not within the context of this dissertation. The fundamental concept of the ones used by the author will be briefly discussed now and implementation details will be provided in the relevant chapters.

Many of the algorithms for compressive sensing are optimization algorithms. Optimization algorithms are problems which serve to minimize (or maximize) an objective function F_0

$$\hat{\mathbf{x}} = \underset{\mathbf{x} \in \mathbb{R}^{\mathbf{N}}}{\arg\min} \; F_0(\mathbf{x}), \tag{2.48}$$

which is called an unconstrained optimization problem. Similarly, solving the problem

$$\hat{\mathbf{x}} = \underset{\mathbf{x} \in \mathbb{R}^{\mathbf{N}}}{\arg\min} \; F_0(\mathbf{x}) \quad \text{subject to} \quad F_n(\mathbf{x}) \leq b_n \quad \forall\, n \tag{2.49}$$

is a constrained optimization problem. If $F_n \; \forall \; n$ are convex then the problem is a convex optimization problem. If $F_n \; \forall \; n$ are linear functions, then it is called a linear program (program is synonymous with optimization) [36].

Given measurements $\mathbf{g}$ and knowledge that $\mathbf{x}$ is sparse or compressible, solving an optimization problem of the form

$$\hat{\mathbf{x}} = \underset{\mathbf{x}}{\arg\min} \; \|\mathbf{x}\|_0 \quad \text{subject to} \quad \mathbf{A}\mathbf{x} = \mathbf{g} \tag{2.50}$$

where $\|\mathbf{x}\|_0$ is the ℓ_0 norm, which is equal to the number of nonzero entries in vector $\mathbf{x}$.[4] Equation (2.50) returns estimate $\hat{\mathbf{x}}$ that resembles $\mathbf{x}$ as long as the measurement noise is small. This is referred to as ℓ_0-minimization. The constraint ensures the solution agrees with the observed measurement data. In other words, one wants the sparsest solution possible that agrees with the measurement data.

Unfortunately, ℓ_0-minimization is nonconvex which means that iterative methods may not converge to a solution. This is an important feature for inverse problems. A convex problem has the property that any local minimum is also a global minimum. In fact, it has been shown that for a general matrix $\mathbf{A}$, ℓ_0 minimization is intractable [57]. This means that one can do no better than a brute force search for the answer.

Fortunately, one can minimize the ℓ_1 norm

$$\hat{\mathbf{x}} = \underset{\mathbf{x}}{\arg\min} \; \|\mathbf{x}\|_1 \quad \text{subject to} \quad \mathbf{A}\mathbf{x} = \mathbf{g} \tag{2.51}$$

[4] The ℓ_0 norm is not strictly a norm and is actually a quasi-norm.

where $\|\mathbf{x}\|_1 = \sum_n^N |x_n|$, and get excellent reconstruction in the case where Equation (2.43) and the Equation (2.44) are satisfied. Not only is Equation (2.51) convex, it can be reformulated as a convex quadratic optimization problem which means that it can be solved by standard optimization methods [31, 29, 36]. Equation (2.51) is the problem that much of the theoretical framework of compressive sensing provides guarantees for.

A more practical version of Equation (2.51) is written as

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}} \ \|\mathbf{A}\mathbf{x} - \mathbf{g}\|_2^2 + \tau\|\mathbf{x}\|_1 \tag{2.52}$$

where τ is a non-negative number. τ is called the regularization parameter and serves to change the sparsity level of the solution and can be used to account for noise, by setting small noise contributions to zero, which increases the robustness of the optimization.[5] In this form, the problem is called ℓ_1 regularized least squares (LS). ℓ_1 regularization also appears in the context of basis pursuit denoising [58].

In statistics, ℓ_1 regularized LS is often referred to as the *lasso* regression method or the lasso problem. The original paper for lasso describes both the problem and the lasso algorithm. However, there are multiple algorithms for solving the ℓ_1 regularized LS problem, such as Least Angle Regression (LAR) [59], truncated Newton interior point methods [60], and Conjugate Gradients algorithm [54].

A related regression technique which is also used to regularize statistical models to prevent overfitting called is *ridge regression*,

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}} \ \|\mathbf{A}\mathbf{x} - \mathbf{g}\|_2^2 + \tau\|\mathbf{x}\|_2. \tag{2.53}$$

In the context of optimization this is called the Tikhonov regularization problem [60, 61]. As with least squares, ridge regression seeks coefficient estimates that fit the data well. In ridge regression the regularization is performed by using an ℓ_2 term, which also tends to shrink the coefficients the towards zero as τ increases but does not set certain elements to zero like in *lasso* regression.

In Figure 2.4, I show an example of sparse signal recovery using the built-in MATLAB `lasso` function. In this case the signal $\mathbf{f}$ is sparse in the native basis

[5] Most papers in compressive sensing use λ to denote the regularization parameter. I choose to use τ to prevent confusing it with wavelength since I will be discussing spectral compressive sensing later on.

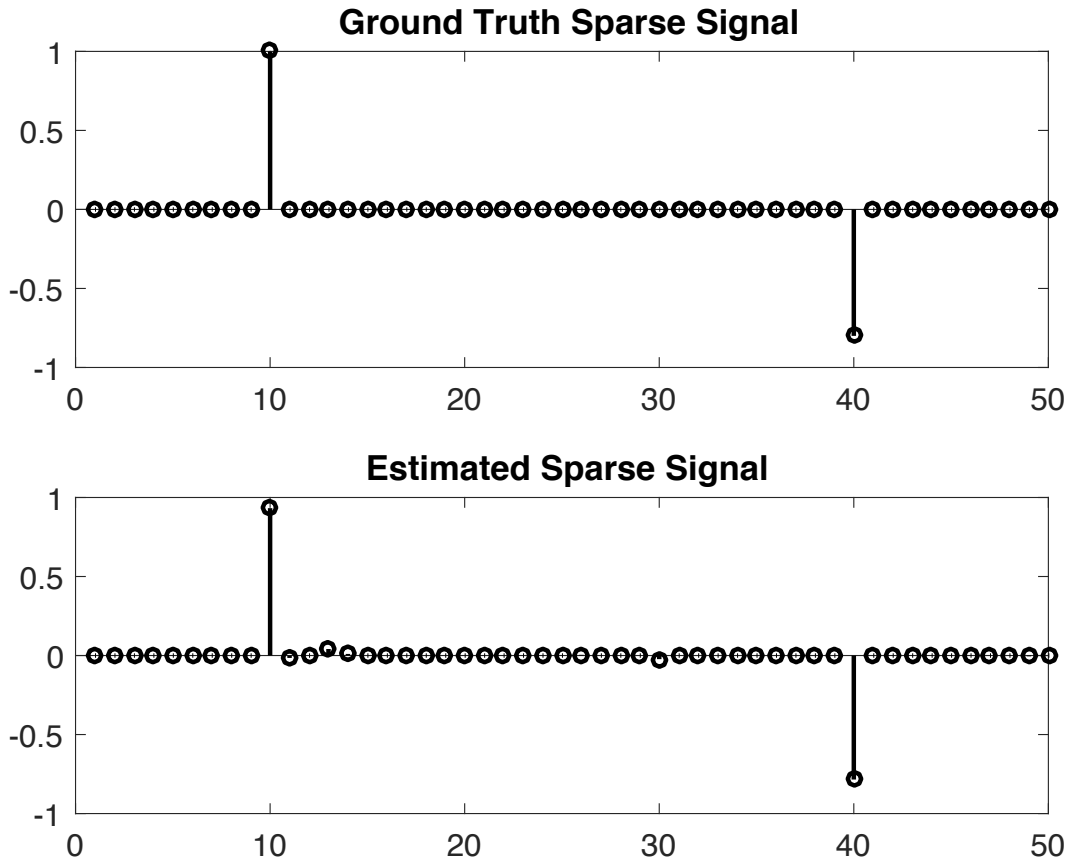

Figure 2.4: Example of sparse vector recovery using ℓ_1 regularized least squares algorithms. SNR = 100. The top plot shows the ground truth signal $\mathbf{x}$. The bottom plot shows the estimated signal $\hat{\mathbf{x}}$ and $\tau = 0.02$. The number of measurements is $N_m = 10$. The dimensionality of the signal is $N = 50$.

so the representation basis can be written as the identity matrix $\mathbf{\Psi} = \mathbf{I}$. The SNR is the ratio of the variance of the signal to the variance of the noise and is set to SNR = 10. The top plot shows the ground truth signal $\mathbf{f}$. The bottom plot shows the estimated signal $\hat{\mathbf{f}}$. The regularization parameter is set to $\tau = 0.02$. The sensing matrix $\mathbf{H}$ is shown in Figure 2.5. The number of measurements $N_m = 10$. The dimensionality of a the signal $N = 50$.

One can gain some geometric intuition of why the ℓ_1 regularized least squares produces sparse solutions by looking at Figure 2.6(a). The dot indicates the unconstrained LS solution. The contours represent equal values of the objective function: $\|\mathbf{Ax} = \mathbf{g}\|_2^2$. The solution is when the ℓ_1 constraint intersects one of the contours. Because of the shape of the ℓ_1 ball, this tends to occur along one of the coordinate axes. Figure 2.6(b) represents the situation with ℓ_2 regularized LS, ridge regression,

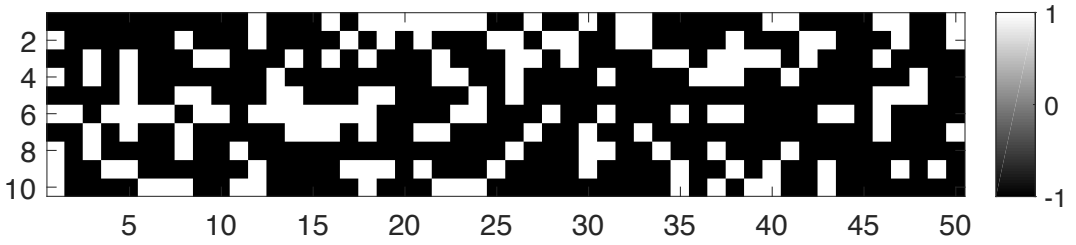

Figure 2.5: The sensing matrix $\mathbf{H}$ used for Figure 2.4. Since the signal was already sparse, the representation basis can be written as the identity matrix $\mathbf{\Psi} = \mathbf{I}$. The number of rows correspond to the number of measurements N_m and the number of columns correspond to the number of elements in the signal vector $\mathbf{f}$.

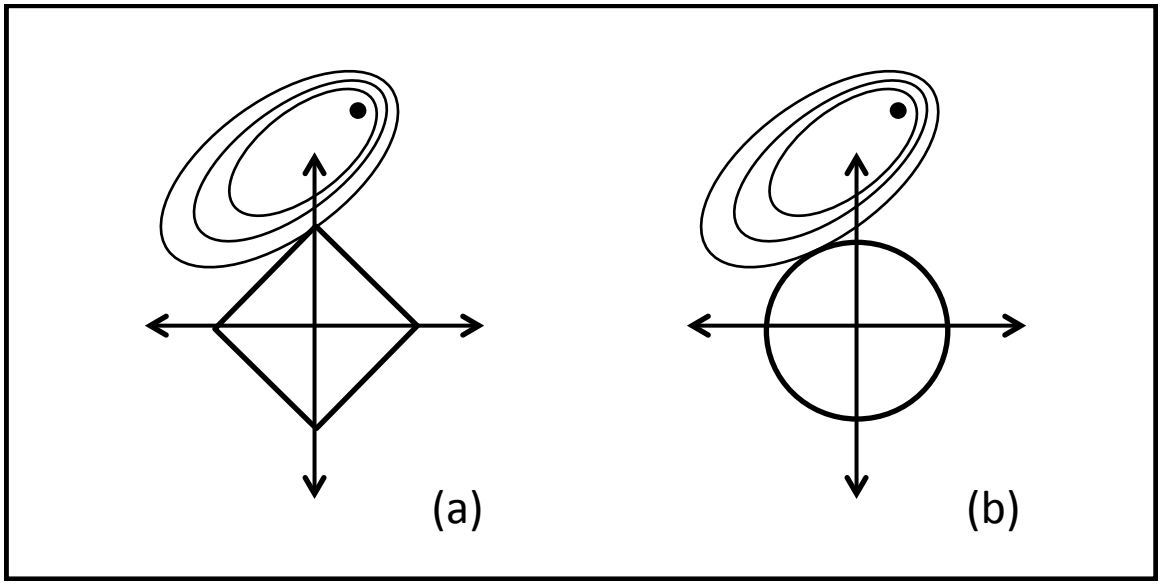

Figure 2.6: (a) ℓ_1 regularized least squares which is called lasso regression. (b) ℓ_2 regularized least squares which is called ridge regression.

because of the shape of the ℓ_2 ball the solutions may or may not occur along the coordinate axes.

The lasso regression technique in statistics is a way to perform variable selection without overfitting the data [62, 63]. In simple terms, overfitting means that given some seemingly complicated data, overfitting occurs when the model attempts to fit all of the data producing a model with very low error with the given data but will tend to have poor prediction results when new data is obtained. By regularizing the objective function, the model will tend to have better prediction accuracy by producing simpler models even though it may have a larger error trying to fit the current data.

Algorithms for solving the lasso improve the accuracy of the estimate by selecting only a subset of the columns of the matrix A which we denote as the A_s. An example of the LAR algorithm which is often used to solve the lasso problem is provided in [59]. By forcing the sum of the absolute value to be less than some fixed value, ℓ_1 regularized least squares forces certain values of $x_n = 0$, effectively choosing a simpler model out of all the possible solutions provided by LS.

2.6 Conclusion

I used this chapter to cover several important mathematical concepts which are required for the reader to understand the advantages and disadvantages of computational sensing. I first discussed the advantages that Hadamard and S-Matrix multiplexing provided compared to the isomorphic sensor which provided a context for the discussion of the Fellgett advantage. Then I discussed Principal Component Analysis (PCA), a useful technique for finding useful discriminating features from seemingly complicated data. I also covered the fundamentals of Bayesian statistics and how it can be used to update estimates of statistical parameters given new data. Then I covered important topics in compressive sensing: sparsity, coherence, and the restricted isometry property (RIP) and discussed the ℓ_1 regularized LS problem which is used to promoted sparsity.

Chapter 3

Static Computational Optical Undersampled Tracker

3.1 Motivation for the Static Computational Undersampled Tracker

In large-area persistent surveillance, traditional isomorphic sensing systems must acquire, process, store, and transmit large amounts of data to achieve high spatial and temporal resolution. These sensors are typically on airborne platforms or orbiting satellites which leads to a large size, weight and power-cost (SWAP-C). However, if one is interested in estimating the location of moving objects (tracking) rather than reconstructing the entire object scene, the information content in the signal-of-interest is relatively low. One can significantly lower the SWAP-C by using a task-specific approach. To address the challenges of traditional target tracking, several colleagues and I designed, built, and demonstrated such a device called the Static Computational Optical Undersampled Tracker (SCOUT) [12, 64]. This chapter will provide details in to how we developed the SCOUT.

Most of the current experimental demonstrations of compressive imaging try to reconstruct the entire object scene or spatial-temporal datacube. For example, the Coded Aperture Compressive Temporal Imaging (CACTI) sensor uses a temporally varying psuedo-random coded aperture during a single FPA exposure and attempts to reconstruct a spatial-temporal datacube [16]. In another experimental compres-

sive imager, a rotating cylindrical lens measures scene data using an optical Radon transform and then reconstructs a static object scene [65]. While these architectures demonstrate the rapid progress that researchers have made towards compressive imaging, they emphasize reconstructing high dimensional data and require another post-processing step to extract task-specific information. Furthermore, this data will then need to be compressed again for storage or transmission. If one is concerned with only tasks, then reconstructing the entire scene as an intermediate step is unnecessary and even wasteful.

Another example of compressive imaging is the single pixel camera, discussed in Section 1.6. This sensor uses a Digital Micro-Mirror Display (DMD) to measure in any arbitrary basis, but must do so over many time-sequential measurements [11]. Each measurement is an integrated point-by-point multiplication of the scene locations with the DMD array values. For temporally static scenes, this architecture allows an arbitrarily long exposure time (within the limit of detector saturation) to increase the SNR for each measurement. However, with temporally varying scenes it needs to record all the projections for each frame before the object moves. Increasing the rate at which projections are made is possible, but this reduces exposure time and SNR. One way to overcome this is by implementing a parallel architecture of single pixel cameras, each with a different projection, see Figure 3.1. Clearly this will significantly increase the SWAP-C of the architecture. Another issue with a fully parallel architecture, is that each camera will have a different entrance pupil location, producing parallax.

One issue many computational sensor designs must deal with is "over multiplexing". All real optical ADC devices have a certain amount of dynamic range, in which linearity is valid. As the amount of light that is recorded increases, the amount of dynamic range being used fills up. Architectures which only use a single pixel are at greater risk for over multiplexing.

A computational sensing architecture dedicated to target tracking rather than full object scene reconstruction, can significantly ameliorate these design trade-offs. We developed the Static Computational Optical Undersampled Tracker (SCOUT) with the goal that measurements must be acquired "single-shot" using a conventional FPA [12, 66]. The SCOUT system is an important step toward practical low-cost computational sensing for optical imaging. The system enables parallel "single-

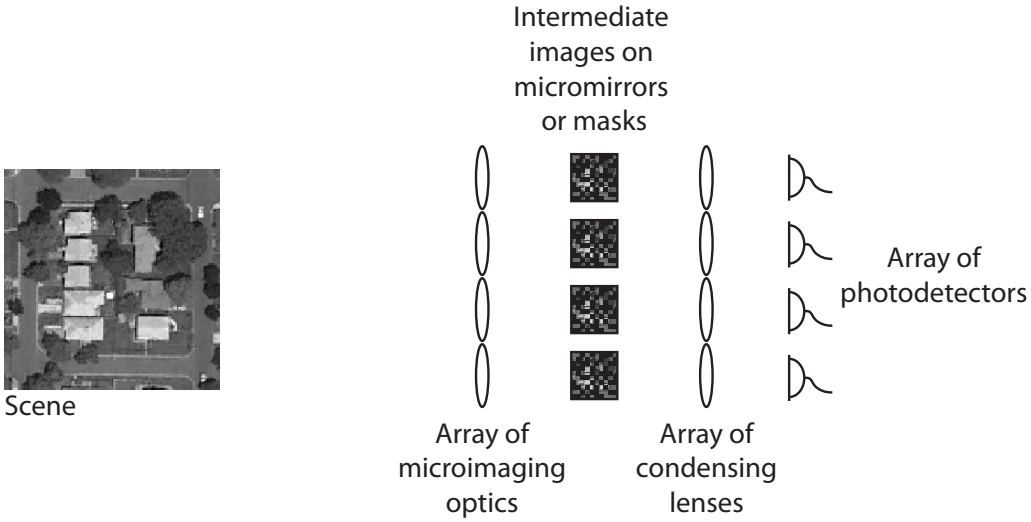

Figure 3.1: An example of a typical parallel optical CS architecture. Capturing N_m simultaneous projections requires using N_m spatial light modulators or masks and N_m detector elements.

shot" acquisition of compressed, task-specific sensing oriented data, using a static (no moving parts) architecture.

A related static approach uses optical Radon projections [67]. This instrument relies on a several cylindrical lenses to integrate the optical intensity from the object plane onto lines on the FPA. The number of detector elements is much less than the native dimensionality of the object scene. For a single moving point in the field-of-view, two perpendicular Radon projections is enough to compute the change from frame to frame. However, the researchers found heuristically that four cylindrical lens are better for reconstructing the change information for an scene that included up to ten moving objects, called movers.

A systems level flowchart of the SCOUT is shown in Figure 3.2. Like many computational sensors, it relies on the coding of the analog signal-of-interest prior to the ADC step and processes the measurement data using prior information and calibration data. The analog instrument was optimized using a custom ray-based simulation which evaluates a metric based on the simulated system matrix. In this chapter, I will discuss the SCOUT architecture in detail in Section 3.2, which uses a defocused imaging system with two binary amplitude masks. The FPA samples at a much lower resolution than the native resolution of the object scene. While other compressive imaging systems have to reconstruct entire images, the SCOUT

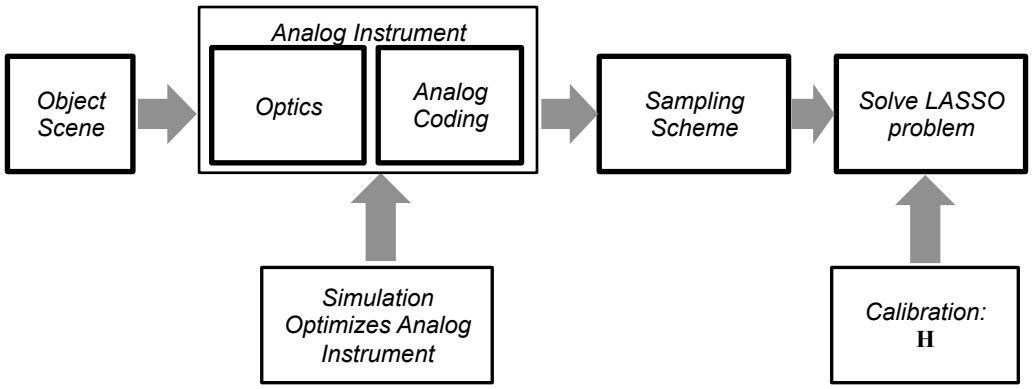

Figure 3.2: A systems level flowchart of the SCOUT system. Light from the object scene propagates to the optical instrument which consists of a coded aperture (mask 1), an objective lens, and another coded aperture (mask 2), a focal-plane array (FPA) then undersamples the coded object scene, measurement data is used to solve a lasso problem. The parameters of the optical instrument is optimized using simulations to minimize the probability of tracking error. Calibration provides accurate measurement of the system matrix **H** to lasso optimization algorithm.

only reconstructs frame-to-frame differences. As a result the SCOUT requires significantly less bandwidth to transmit to a base station where the post-processing step can occur.

3.2 SCOUT Architecture

The goal of the SCOUT was to demonstrate a low SWAP-C compressive target tracking sensor. The SCOUT architecture is designed to avoid the hardware scaling issues of the single-pixel camera. The trade-off for the ability to measure parallel projections is the loss of flexibility to implement arbitrary projections i.e. by using a Spatial Light Modulator (SLM). However, rather than fully designing the projections themselves, I describe a process for optimizing the optical instrument in Section 3.3. Previous prototypes of the SCOUT architecture are described in [64, 68].

The SCOUT system takes measurements that are both compressive and multiplexed: The number of measurements is less than the number of scene locations $N_m \ll N$, and each measurement must contain information about many scene locations. In this architecture, the number of measurements is the number of pixels in the FPA. Intuitively this means that the system matrix must exhibit a many-to-few

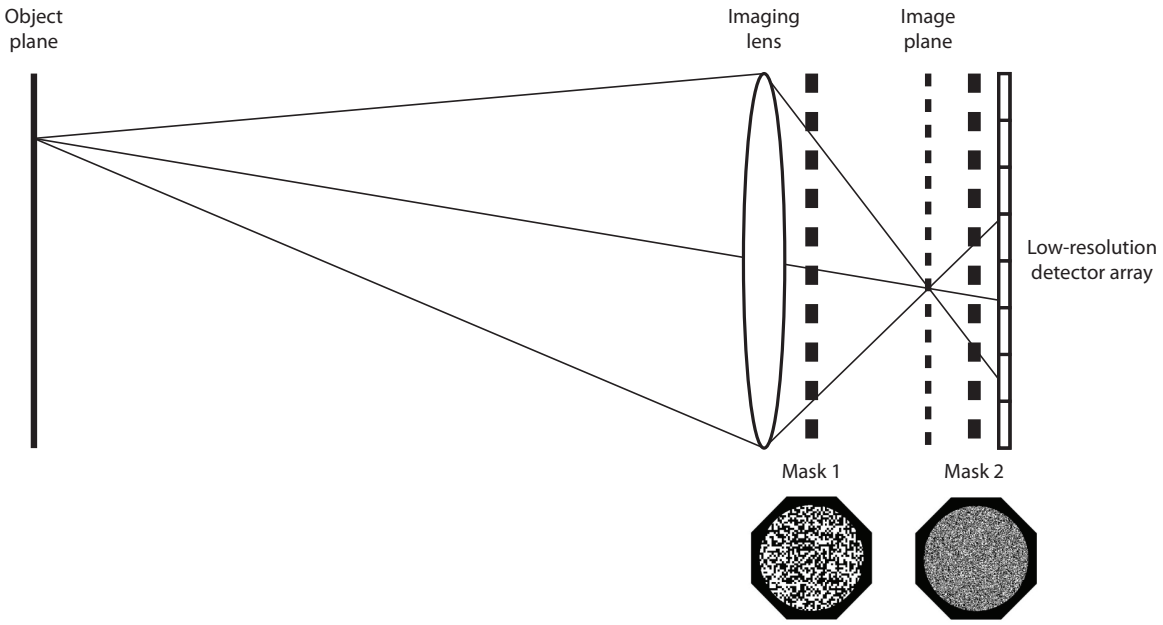

Figure 3.3: A diagram of the SCOUT architecture. A lens projects light through a pair of binary occlusion masks onto a low-resolution sensor which is defocused from the nominal image plane. This creates a spatially multiplexed shift-variant PSF incident on the FPA.

mapping from scene locations to FPA elements.

The SCOUT architecture is shown in Figure 3.3. In this architecture the multiplexing occurs in the spatial domain by mapping multiple object scene locations to only a few detector pixels. To accomplish this, we created a structured blur. This blur allows the light from a single object point to be spread to several pixels on the FPA. The most straightforward way to achieve a blur is by defocusing the image so that the PSF is broad, spanning many FPA pixels. Since the number of FPA pixels are less than the object scene resolution, the measurement is compressive.

Like any aberration, defocus significantly reduces contrast of high spatial frequencies, so the measurements will be poorly conditioned for reconstruction. Therefore, we created high-frequency structure in the PSF by using two pseudo-random binary occlusion masks. Each mask is placed at different positions between the lens and the sensor. The separation between masks results in a spatially varying point-spread function.

As with many linear computational sensing architectures, the forward model is written in the form

$$\mathbf{g} = \mathbf{H}\mathbf{f} + \mathbf{e} \tag{3.1}$$

where $\mathbf{f}$ is the discrete representation of the object signal-of-interest, $\mathbf{g}$ is the measurement, $\mathbf{H}$ is the matrix which describes mapping of the object to the measurement, and $\mathbf{e}$ is the noise at each measurement. In this situation the signal-of-interest is actually the difference between two subsequent object scenes (frames) $\Delta\mathbf{f}$:

$$\begin{aligned} \mathbf{g}_k &= \mathbf{H}\mathbf{f}_k + \mathbf{e}_k \\ \mathbf{g}_{k+1} &= \mathbf{H}\mathbf{f}_{k+1} + \mathbf{e}_{k+1} \end{aligned} \tag{3.2}$$

Where the subscripts represent the k^{th} readout from the FPA.

$$\Delta\mathbf{g} = \mathbf{g}_{k+1} - \mathbf{g}_k = \mathbf{H}(\mathbf{f}_{k+1} - \mathbf{f}_k) + \mathbf{e}_{k+1} - \mathbf{e}_k \tag{3.3}$$

so the forward model can be written as

$$\Delta\mathbf{g} = \mathbf{H}\Delta\mathbf{f} + \Delta\mathbf{e} \tag{3.4}$$

In this equation, the 2-dimensional difference frame and object scene have a resolution of $R_x \times R_y$ elements, and are lexicographically reordered into a $N \times 1$ column vector, $\Delta\mathbf{f}$ and $\mathbf{f}$. Similarly, the difference measurement and measurement $\Delta\mathbf{g}$ and $\mathbf{g}$ are $N_m \times 1$ vectors representing a 2-dimensional FPA readout with a $r_x \times r_y$ matrix which represents the low resolution measurement. The detector noise is represented by the $N_m \times 1$ vector $\mathbf{e}$. The system matrix $\mathbf{H}$ is thus an $N_m \times N$ matrix, where $N_m \ll N$ in order to the system to be considered compressive. The n^{th} column of the matrix is the PSF of the n^{th} location in the object scene. The resulting system matrix $\mathbf{H}$ demonstrates that the SCOUT is a spatially variant optical system and presents a block structure, as seen in the example shown in Figure 3.4.

Referring back to the diagram of the SCOUT architecture shown in Figure 3.3. The object distance is much larger than the focal length of the lens so image of the scene occurs approximately one focal length from the lens f_E. The FPA is placed some distance d_{im} from the focal plane thus the total distance from the lens to the FPA is $f_E + d_{im}$. Two binary occlusion masks, mask 1 and mask 2, are placed at distances d_1 and d_2 from the sensor. Mask 1 and 2 have associated fill factors F_1 and F_2 and pitch p_1 and p_2, respectively.

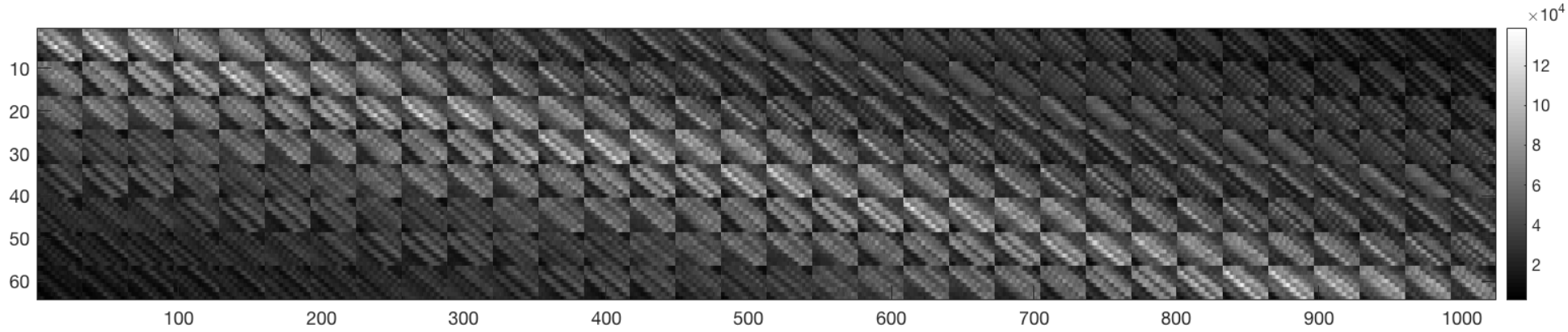

Figure 3.4: An example of an experimentally measured of the system matrix of the SCOUT system. The approximate block-Toeplitz structure is clearly evident, as is the deviation from the Bernoulli or Gaussian ensembles typically considered in CS treatments.

3.3 Optimizing the SCOUT

While the SCOUT lacks the ability to implement arbitrary measurement codes, we are able to adjust various physical parameters of the system such as defocus distance and mask fill factor to minimize reconstruction error. Adjusting each parameter in the actual prototype requires too much time, a more practical approach is to simulate the SCOUT architecture. In this section, I will discuss the simulation and define a metric that we used to evaluate different design parameters of the SCOUT without the need to reconstruct the difference frames.

3.3.1 Simulating a SCOUT System

We developed a paraxial ray based simulation for the SCOUT. The simulation allows us to model the effects on the measurement as light travels through the lens and two masks and onto the detector plane. The lens is modeled as a single thin lens with transmittance function t_f and the two masks have transmittance functions t_1 and t_2. From calibration measurements, the mask has a transmittances of 0 where the mask is black and 0.88 where the mask is clear. The lateral magnification of the scene and the two masks is calculated using similar triangles.

A simulated calibration occurs, in other words the simulation records the $r_x \times r_y$ PSF from each scene location in order to obtain the system matrix $\mathbf{H}$. Once $\mathbf{H}$ is known, we use it to simulate the low resolution measurements, $\mathbf{g}$, of the higher resolution scenes, $\mathbf{f}$. Subsequent simulated measurements are subtracted to find $\Delta\mathbf{g}$. We also did not add any noise. I have attached the simulation code in Appendix D.

3.3.2 Quantifying Reconstruction Error

The MSE error metric is not suitable for our application because it weights all errors equally. For the task of motion tracking we classify errors into three types. A false positive occurs when the estimate shows an object where there is none. A false negative occurs when the estimate fails to show an object where one exists. As shift error occurs when an object is being tracked but appears in the wrong location. We developed a custom tracking error metric that weighs false negatives and false positives more than shift errors. This is because false negatives and false positives

indicate a serious failure in the motion tracking task, while shift errors are less serious. We define tracking error as

$$P = \frac{|\mathbf{a} \otimes \epsilon|}{2N_{mv}} \tag{3.5}$$

where

$$\mathbf{a} = \begin{bmatrix} 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \end{bmatrix} \tag{3.6}$$

where the error ϵ is the difference between the true and reconstructed difference frames

$$\epsilon = \Delta\mathbf{f} - \Delta\hat{\mathbf{f}} \tag{3.7}$$

and N_{mv} is the number of movers in the scene. To reduce the penalty for one-pixel shift errors, the error frame is convolved with a three pixel averaging kernel $\mathbf{a}$. The absolute value is taken in order to count positive and negative errors equally, and the error is divided by $2N_{mv}$ to make the metric independent of the number of movers.

3.3.3 Optimizing Optical System Parameters

Now that I have explained the simulation and the custom error metric for tracking, I can finally begin to discuss how we optimized some of the parameters to reduce the reconstruction error. Both the simulation and experimental study demonstrates a relationship between mask position and pitch: the tracking error is very sensitive to the *projected* mask pitch on the FPA. Furthermore, increasing the mask seperation reduced tracking error, generating a highly space-variant PSF. With these observations in mind, we focus our study on the mask pitches (p_1, p_2) as well as the defocus distance d_{im}.

A brute force search, using the lasso solver at each step is computationally intensive. We wanted a simple to compute metric which can predict tracking error. So we developed a custom metric inspired by the coherence parameter from the compressive sensing community, see Equation (2.42). We created a customized coherence parameter μ,

$$\mu = \max |\langle h_i, h_j \rangle|; \qquad i \neq j \tag{3.8}$$

which is the maximum absolute value of the inner product between unique columns h_i and h_j of $\mathbf{H}$. The columns are unnormalized because their relative magnitude is related to the physical light throughput.

Notice that although system matrices with nearly pairwise orthogonal columns will result in small coherence values, system matrices with numerically small entries can accomplish the same. Optimizing $\mathbf{H}$ for minimum coherence would drive total system throughput down. To eliminate this effect we normalized $\mathbf{H}$:

$$\mathbf{H}_{norm} = \frac{\mathbf{H}}{\sum_{m=1}^{N_m} \sum_{n=1}^{N} h_{m,n}} \tag{3.9}$$

where N_m and N are the total number of rows and columns in the system matrix. Physically, this normalization represents division by the sum of each PSF's light throughput. The coherence of a system matrix normalized in this way cannot be biased by reducing throughput. One consequence of this normalization is that mask fill factor cannot be optimized.

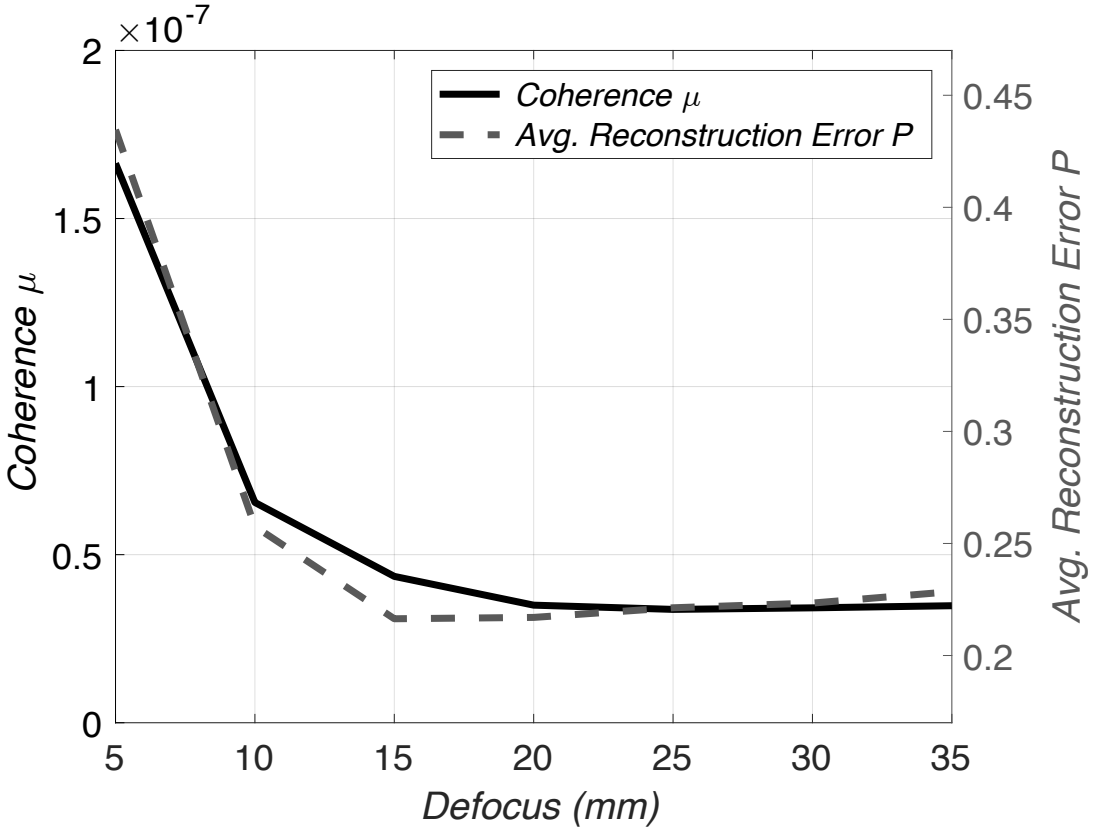

Figure 3.5: The coherence μ (left vertical axis - black) and reconstruction error P (right vertical axis - dashed gray) plotted as a function of defocus distance $d_i m$.

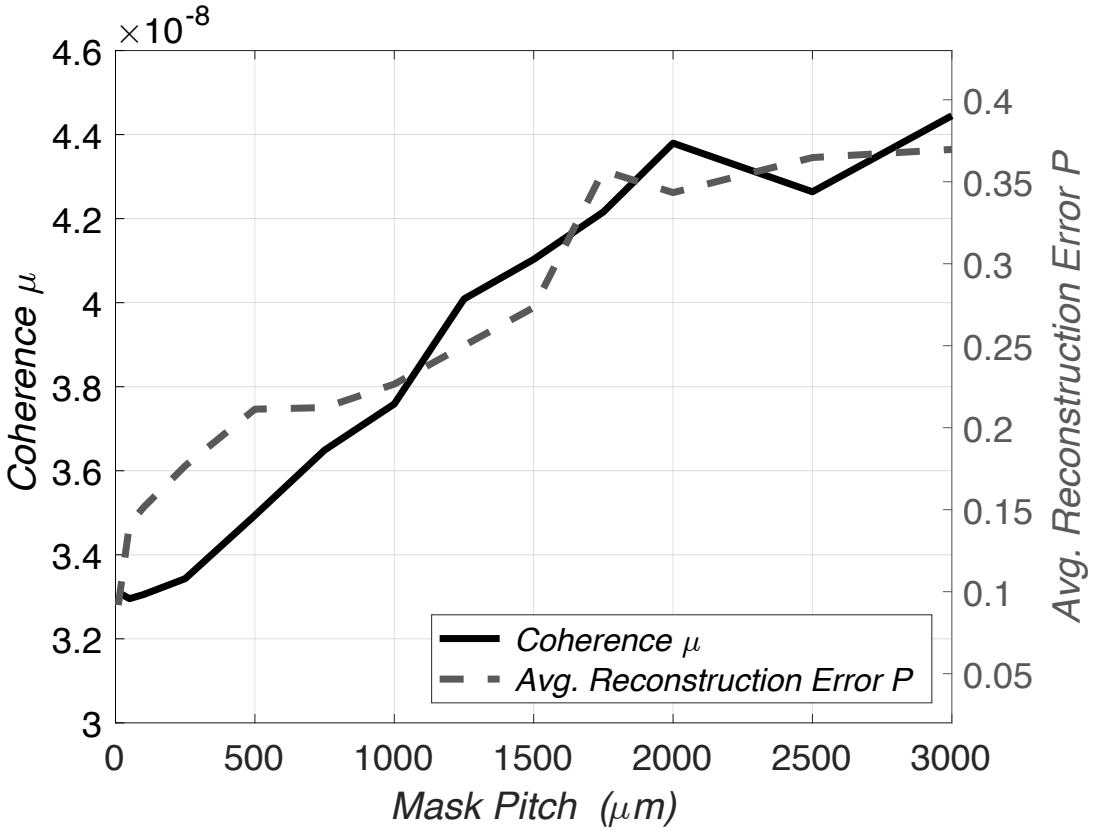

Figure 3.6: The coherence μ (left vertical axis - black) and the reconstruction error P (right vertical axis - dashed gray) is plotted as a function of the pitch of mask 2

To demonstrate the effectiveness of the modified coherence parameter as a predictor of reconstruction error trends, I ran several simulations with complete reconstructions in order to compare reconstruction error to coherence. Figure 3.5 shows that reconstruction error and the coherence parameter follow similar trends for different defocus distances when all other parameters are held constant. Figure 3.6 shows similar agreement when varying values of mask pitch p_2.

The simulations demonstrate the viability of the architecture and provide an efficient way to optimize most architecture parameter values using the simulated system matrix coherence. Mask throughput cannot currently be optimized because the custom coherence metric is normalized by full system throughput. The problem of finding optimal mask throughput warrants further investigation.

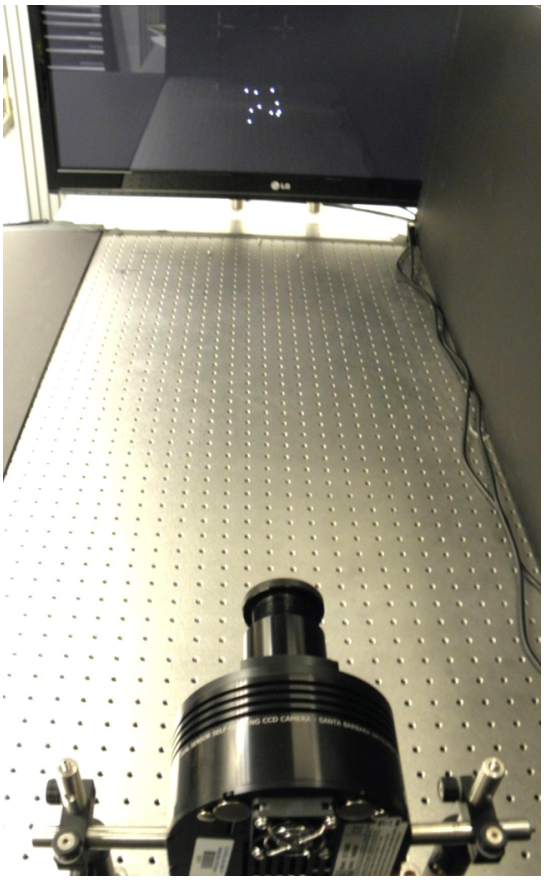

Figure 3.7: The camera captures images of scenes displayed on a plasma television approximately 2 meters away..

Figure 3.8: The camera disassembled to show the camera body, the optical tube and the custom fabricated lens holder which goes inside the lens tube.

3.4 Experiment

3.4.1 Experimental Setup

For the experiment, the captured image resolution $r_x \times r_y$ is 8×8, while the ground-truth and reconstructed frame differences have a resolution $R_x \times R_y$ of 32×32. To

simulate a low-resolution detector, the camera captures the scenes at 128×128 sensor pixels and the images are binned down to 8×8 before being used in reconstruction.

We used a SBIG Model ST-7XMEI CCD camera with modified optics. The object scenes were displayed on a plasma television monitor. Figures 3.7 and 3.8 shows a photograph of the experimental setup. The optical system of the camera includes a 35 mm focal length lens and two random amplitude binary masks. Each mask was printed on transparencies using a high resolution laser printer. Mask 1 has pitch $p_1 = 30\,\mu\text{m}$ and fill factor $F_1 = 0.4$ and is located at a distance $d_{m1} = 14\,\text{mm}$ from the sensor. Mask 2 has pitch $p_2 = 500\,\mu\text{m}$ and fill factor $F_2 = 0.2$ and is located at a distance $d_{m2} = 57\,\text{mm}$ from the sensor. These parameters were chosen based on the aforementioned optimization process. A plasma monitor was used because it provides a higher contrast compared to the traditional liquid crystal displays (LCD). However, the black background of the plasma monitor still produced a small amount of irradiance, which is a source of systematic error and potentially reduces the usable amount of dynamic range.

3.4.2 Calibration

Instruments based on compressive sensing rely heavily on accurate calibration. Especially important is the knowledge of the system matrix $\mathbf{H}$ to prevent reconstruction or task-specific sensing errors. The SCOUT architecture is no different. Inaccurate calibration dramatically affects the tracking error. Furthermore, because of the non-isomorphic nature of compressive sensing, it is difficult to look at the system matrix and intuitively tell whether it will lead to good results. In this section, I will describe the calibration procedure for the SCOUT architecture.

Conceptually the calibration procedure is straightforward. The goal is to experimentally determine the system matrix $\mathbf{H}$. Each column of the matrix is the point-spread function (PSF) due to a "point" at the n^{th} location in the object scene. All one needs to do is display the point at each location and store the measurement in the respective column of $\mathbf{H}$.

There are several practical issues in the calibration process for the SCOUT. Calibration itself is a measurement process, noise is present in each measurement. To mitigate the effects of noise, we cooled the CCD in the camera to 0 degrees Celsius

using the built-in thermoelectric cooler (TEC). We also increased the exposure time to 1.0 seconds. While it is possible to continue increasing the exposure time, we found that increasing the exposure time past this did not significantly reduce the tracking error metric.

To eliminate any systematic error due to light pollution, we constructed a light-tight box using 80/20 aluminum framing, black poster board, and black gaffer tape. This box enclosed the entire SCOUT and the plasma monitor. We also found that the plasma monitor emitted a certain amount of light even though it is set to zero. To mitigate this, we take several dark frame measurements, which is a measurement with the plasma monitor set to all zero and then averaged. This averaged dark frame measurement is subtracted from each PSF measurement.

Another issue with the plamsa is that the intensity varies after a few minutes. Therefore, every 60 seconds we pause the calibration procedure and set the entire screen to all white. This resets the intensity levels and a new set of dark frame measurements is recorded. The calibration sequence is then allowed to continue.

Another issue with the plamsa monitor is that when a particular pixel is illuminated, the intensity of the adjacent pixels change. So when the next pixel is illuminated, that intensity is different compared to the intensity we measure if the adjacent pixel had not been turned on. In other words, turning on pixel n changes the intensity at pixel $n+1$ when it is turned on. In order to mitigate this effect, we created a psuedo-random sequence so that after a certain amount of time, the effect of a neighboring pixels activity is reduced. The total time to calibrate the SCOUT is approximately 20 minutes.

Remember that an isomorphic sensor is represented by the identity matrix. In comparison, in the SCOUT, the spatially varying blurred PSF leads to an approximate block-Toeplitz structure for the system matrix, with approximate Toeplitz structure within individual blocks due to the shifting PSF. This circulant structure is modified by random variations corresponding to the differing projections created by the two masks. Psuedo-code describing the calibration is given in Appendix C.

3.4.3 Reconstruction: ℓ_1 regularized Least Squares Minimization

Given $\Delta\mathbf{g}$ and $\mathbf{H}$, reconstructing the difference frame $\Delta\mathbf{f}$, is a highly underdetermined problem given no other prior knowledge. As discussed in Section 2.5, several important theoretical results show that it is possible to accurately recover $\Delta\mathbf{f}$ if the sparsity K is low relative to the number of measurements N_m and the RIP is satisfied. Inspired by these results, we turn to algorithms designed to solve the ℓ_1 regularized LS (lasso) problem:

$$\Delta\hat{\mathbf{f}} = \arg\min_{\Delta\mathbf{f}} \|\mathbf{H}\Delta\mathbf{f} - \Delta\mathbf{g}\|_2^2 + \tau\|\Delta\mathbf{f}\|_1 \tag{3.10}$$

Given $\Delta\mathbf{g}$ and $\mathbf{H}$, the reconstruction algorithm finds a solution $\Delta\hat{\mathbf{f}}$ that minimizes this objective function.

We used the `l1_ls` toolbox for MATLAB which implements an optimization technique based on Interior-Point methods [60]. The `l1_ls` function requires several input arguments: $\Delta\mathbf{g}$, $\mathbf{H}$, τ the regularization parameter, and a parameter called the relative tolerance, `rel_tol`. As I discussed in Section 2.5, τ is a tuning parameter that is used to tell the optimization algorithm how much to weight the sparsity of the solution. Large τ tend to drive the solutions towards lower values of sparsity, K. The `rel_tol` controls how well the solution should agree with the data. Low values of `rel_tol` tend to force the `l1_ls` to run many iterations until the a threshold has been reached. While large values of `rel_tol` tend to produce poorer reconstruction results but less optimization iterations.

We found that a regularization parameter of $\tau = (1 \times 10^{-9})\,\|2\,\mathbf{H}_{cal}^T\,\Delta\mathbf{g}\|_\infty$ works well for experimental reconstruction. Where $\mathbf{H}_{cal}$ is the system matrix measured from calibration and $\|\cdot\|_\infty$ is the infinity norm. The `rel_tol` is set to 1×10^{-4}.

Finding the correct regularization parameter is one of the major issues for many algorithms designed for compressive sensing. In our experiment, we had to run the reconstruction over many iterations with varying τ in order to find the appropriate value. Unfortunately this also depends on the sparsity of the signal-of-interest. Therefore, large numbers of movers may have a different optimal value for τ. The value of τ, we reported works well from one to ten movers in our experiment.

3.4.4 Experimental Results

Experimental results for a ten difference frame sequence is shown in Appendix B. The object scenes contains two dots (movers) changing position on a black background. Difference frame 1 of this sequence is shown in Figure 3.9. The top row shows two consecutive, before and after, $\mathbf{f}_1$ and $\mathbf{f}_2$, frames of the scene and the ground-truth difference frame $\Delta\mathbf{f}$, all at 32×32 resolution. The bottom row shows the difference of corresponding 8×8 measurement frames, $\Delta\mathbf{g}$. Finally, the 32×32 reconstructed difference frame is shown at the bottom right, $\Delta\hat{\mathbf{f}}$.

Initially, the amplitude of the movers in the estimated difference frame to did not agree qualitatively with the amplitude of the ground truth difference frame. We realized that this was due to the fact that the exposure time during calibration was not the same as the exposure time during the actual experiment. By normalizing the system matrix obtained during calibration by the ratio of exposure times, we were able to demonstrate quantitative agreement with the ground-truth:

$$\mathbf{H}_{recon} = \frac{t_{exp}}{t_{cal}}\mathbf{H}_{cal} \tag{3.11}$$

where t_{exp} and t_{cal} are the experiment and calibration exposure times, respectively. This scaling accounts for the physical effect of increased photon collection (and hence photodetector counts) as a function of increased exposure time. The resulting peaks are easily identified against background noise.

In the reconstruction of the ninth difference frame, the amplitude of one the movers had a lower amplitude, shown in Figure 3.10. Poor reconstruction tends to occur when two movers are located adjacent to each other in the ground-truth difference scene. This is an issue that can be traced to the system response matrix $\mathbf{H}$, locations next to each other have are more likely to have larger correlations in their respective PSF.

We also performed a more realistic experiment in which a mover simulates a vehicle driving on a street. This demonstrates that the SCOUT works well in situations with non-zero backgrounds. This sequence with the results for a single mover is also shown in Appendix B. The first difference frame of this sequence is shown in Figure 3.11. As seen in ground truth difference frame $\Delta\mathbf{f}$ shown in Figure 3.11(c), the amplitude of the past and present mover locations in the ground-truth are lower

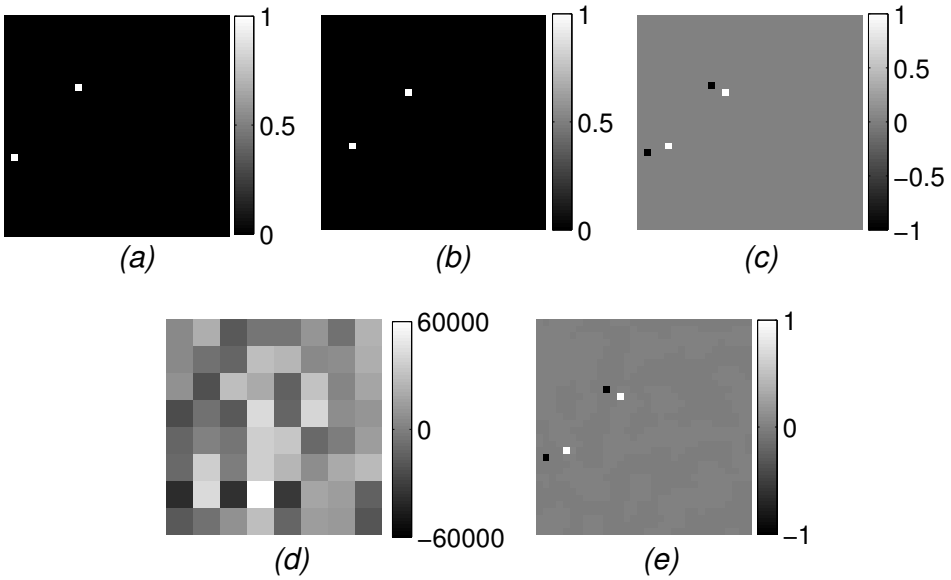

Figure 3.9: A reconstruction of a 32×32 scene with two movers of equal amplitude on a black background. (a) ground-truth scene 1 (b) ground-truth scene 2 (c) ground-truth frame difference and (d) measured 8×8 frame difference, scaled so that it is discernible (e) reconstructed 32×32 difference frame

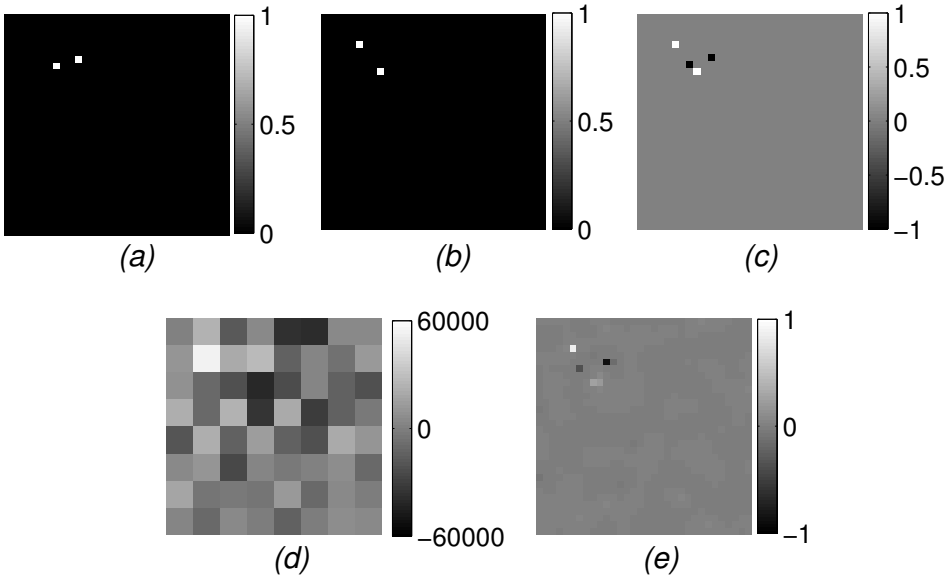

Figure 3.10: A reconstruction of 32×32 scene with two movers of equal amplitude on a black background. This frame shows the results when the past and present mover locations are adjacent in the difference frame. (a) ground-truth scene 1 (b) ground-truth scene 2 (c) ground-truth frame difference and (d) measured 8×8 frame difference, scaled so that it is discernible (e) reconstructed 32×32 difference frame

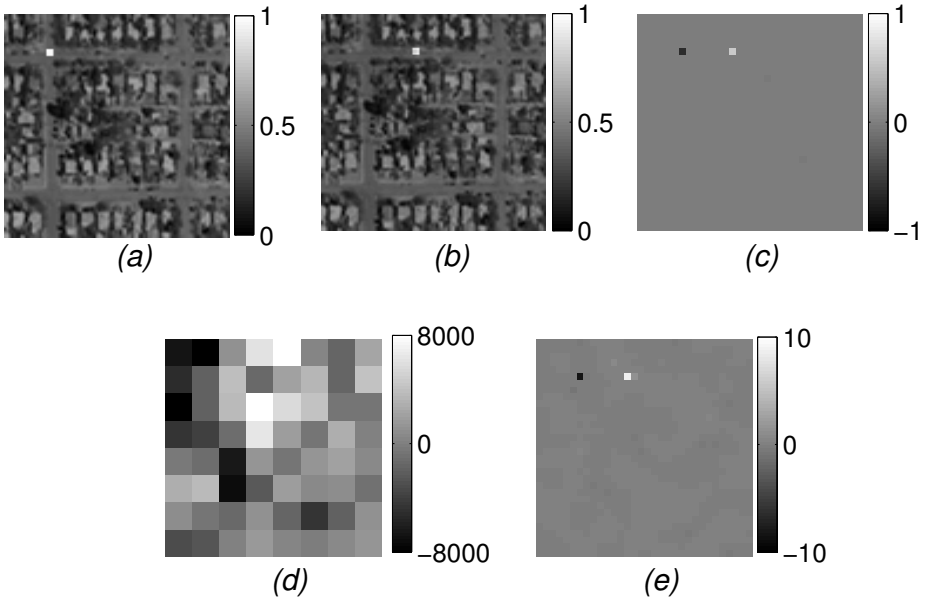

Figure 3.11: Difference frame 1 of a video demonstration of compressive tracking of a 32×32 difference scene with a non-zero background. Scene background ©2012 Google. (a) ground- truth scene 1 (b) ground-truth scene 2 (c) ground-truth frame difference and (d) measured 8×8 frame difference, scaled so that it is discernible (e) reconstructed 32×32 difference frame

than the zero background case. Therefore there is also less contrast in $\Delta\mathbf{g}$ the difference measurements in Figure 3.11(d), which makes this case more sensitive to noise.

The most notable feature in the non-zero background results is reduced quantitative agreement with the ground-truth, even when the calibration matrix is scaled according to Equation (3.11). There are several reasons for this: Nonlinearities in the overall system response that result from a nonlinear monitor "gamma" (mapping from pixel value to output brightness) and inter-pixel interactions that effect brightness. These effects are not captured during calibration as that is performed point-by-point (thus reducing inter-pixel effects) and with pixels that are fully-on or -off (thus avoiding effects from monitor gamma). Another possible reason is over-multiplexing, since the total light from each frame is increased, there is less dynamic range in the FPA, and therefore detector non-linearity may be a source of error. Despite the lack of quantitative agreement, qualitative agreement is excellent and the movers are clearly identifiable against the background in Figure 3.11(e).

3.5 Conclusion

While the SCOUT architecture is well-suited for tracking applications, it does have limitations which make it less useful for general imaging applications. Without sparse scene motion, the priors used in reconstruction will lead to incorrect results. Reconstructions only show the locations of moving objects, and the sensing platform must be stationary relative to the scene so that frame differences are sparse. However, more sophisticated techniques could potentially estimate platform motion. Despite the limits of the SCOUT, the architecture is well-suited for applications such as fixed-camera wide-area surveillance where bandwidth, data volume, and cost are key concerns.

Many of the theoretical guarantees for compressive sensing are not specialized or tuned for the block-circulant system matrix. As I mentioned in Chapter 2, random coding has several theoretical properties that make them useful for compressive sensing. There has been some research work to investigate system matrices with Toeplitz and circulant structure [69, 70, 71], however there has been relatively little work published discussing the approximately block-Toeplitz structure that naturally arises in optical systems such as SCOUT and theoretical guarantees like the ones for random coding. Two exceptions are [72, 73], which provide both theoretical evidence for the viability of CS system matrices with block-Toeplitz structure.

A completely parallel compressive imager would require as many encoding optical elements as simultaneous measurements. The SCOUT architecture eliminates this scaling issue by giving up the ability to implement arbitrary projections. Using a pair of masks at different distances to create a block-circulant system matrix, the system makes compressive measurements and reconstructs frame differences. The system can be optimized by adjusting system parameters such as mask pitch and defocus distance. Simulations demonstrated the use of the a modified coherence parameter as an efficient predictor of system matrix performance to optimize these parameters. An experimental system based on the SCOUT architecture successfully performed compressive motion tracking on scenes with zero and nonzero backgrounds in most instances. However, the reconstruction of difference scenes with adjacent mover locations caused issues due to the design or calibration of the system matrix. The system showed promising results using a general ℓ_1 regularized

least squares minimization algorithm and I believe that further research on sparse reconstruction with block-circulant system matrices may decrease reconstruction error. We also believe that non-isomorphic calibration techniques and adding further degrees of freedom in the design parameters could result in significant performance gains.

Chapter 4

Adaptive Feature Specific Spectral Imaging-Classifier

4.1 Motivation

Spectral imaging allows for improved discrimination of objects in a scene by measuring both spatial and spectral data [74, 75, 76]. By combining the spectrometer with the camera, the spectral imager produces a spectral datacube, which consists of two spatial dimensions and a spectral dimension [77, 78], see Figure 4.1(a). In this chapter, I will introduce the Adaptive Feature Specific Spectral Imaging-Classifier (AFSSI-C), a computational spectral imaging system which directly classifies the spectrum at each spatial location in a scene.

One of the major limitations of isomorphic sensing techniques in spectral imaging is due to the fact that one must acquire a three-dimensional spectral datacube using a two-dimensional focal-plane array (FPA) [77]. Traditional isomorphic systems rely on a point-by-point acquisition technique to acquire the entire spectral datacube. A *whiskbroom* technique simultaneously measures the entire spectrum from a single spatial location. This is repeated for each location until the spectral datacube is completely acquired, see Figure 4.1(b) [79]. A *pushbroom* technique measures the spectrum of an entire spatial row or column at a time, see Figure 4.1(c). This is repeated until all the rows (or columns) in the spectral datacube is acquired [80, 79]. A *tunable filter* technique, such as the Fabry-Perot interferometric filter [81, 82, 83], simultaneously measures a single spectral channel over the entire field-of-

view (FOV), scanning through the spectral dimension, see Figure 4.1(d) [84]. The

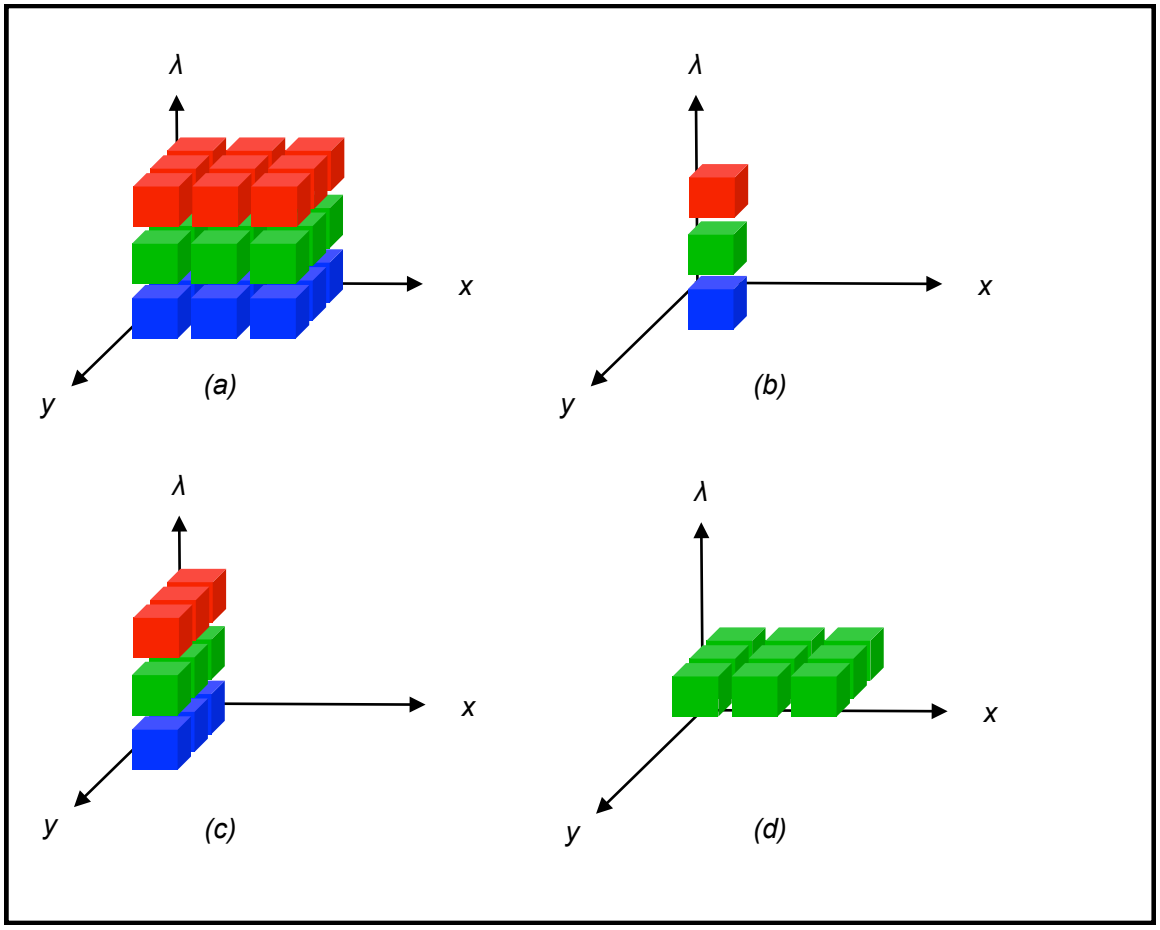

Figure 4.1: (a) A discrete spectral datacube with $R_x = 3, R_y = 3, N_\lambda = 3$. (b) The whiskbroom technique measures the entire spectrum one spatial location at a time. (c) The pushbroom technique measures the entire spectrum on an entire spatial row or column at a time. (d) The tunable filter technique measurements an entire monochromatic image one spectral channel at a time.

problem with the traditional ismorphic technique is that a typical spectral datacube has a significant amount of measurement samples. The Airborne Visible/Infrared Imaging Spectrometer (AVIRIS) system, acquires $R_x \times R_y = 677$ spatial locations (pixels) and $N_\lambda = 224$ spectral channels [85], producing a spectral datacube with $N \approx 10^9$ measurement samples in 10 minutes.

Just like with the traditional spectrometer and camera, researchers have turned computational sensing for spectral imaging. These architectures use the Fellgett advantage (multiplexing) and *Jacquinot advantage* (open aperture) to improve the signal-to-noise ratio (SNR) and reduce acquisition time and use a computational step to solve an inverse problem to reconstruct the spectral datacube D from non-

isomorphic measurements. Some spectral imaging architectures also leverage compressive sensing.

One of the early examples of computational sensing in spectral imaging is the Computed Tomography Imaging Spectrometer (CTIS) [86], see Figure 4.2. The CTIS can reconstruct the spectral datacube D from a single FPA exposure by recording multiple measurements of the spectral datacube simultaneously. The CTIS uses several gratings to create two-dimensional projections of the three-dimensional hyperspectral datacube, see Figure 4.3. According to the central slice theorem, the two-dimensional Fourier Transform of each projection is a plane through the three-dimensional frequency space representation the spectral datacube. Ideally, one must collect enough projections to fully reconstruct the three dimensional frequency representation of the spectral datacube. The three-dimensional real-space distribution of the spectral datacube is then recovered through an inverse three-dimensional Fourier transform. This is similar to how computed tomography medical imaging works. However, in practice only a finite subset of projections are recorded and missing information is must be inferred. In the CTIS, the missing information is recovered by maximizing the likelihood of the measurement data using the expectation-maximization algorithm [87, 88]. According to the practical definition of compressive sensing, the CTIS maybe considered a compressive sensing technique since the number of measurement samples is less than the object dimensionality of the spectral datacube. However, it does not take advantage of sparsity or incoherence in order to reconstruct the spectral datacube.

In another example of computational sensing applied to spectral imaging is the Coded Aperture Snapshot Spectral Imaging (CASSI) sensor, see Figure 4.4. In the CASSI, a coded aperture spatially codes the spectral datacube D. The dispersive element then creates a projection of the spectral datacube that maps three-dimensional information into a two-dimensional image at the FPA. Unlike the CTIS, only one two-dimensional projection is recorded per FPA image which allows for higher spatial resolution for the same detector array. Using calibration data and prior knowledge of sparsity, the post-processing step solves the lasso problem to reconstruct the spectral datacube [32, 89]. Often a total-variation regularization is also invoked to improve reconstruction of the spatially varying image [90, 91].

All traditional and computational spectral imaging architectures including the

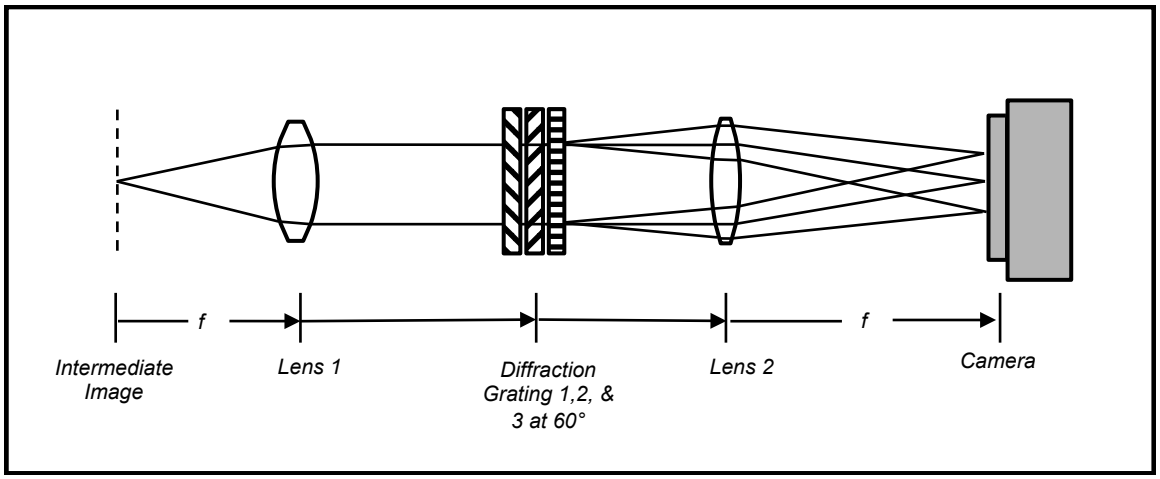

Figure 4.2: The architecture of the Computed Tomography Imaging Spectrometer (CTIS) consists of a collimating lens, several diffraction gratings, an imaging lens, and a focal-plane array (FPA). Each diffraction grating produces three two-dimensional projections of the three-dimensional spectral datacube (Two first order and one zeroth order). In this example, gratings are rotationally seperated by 60 degrees to produce multiple projections. [86]

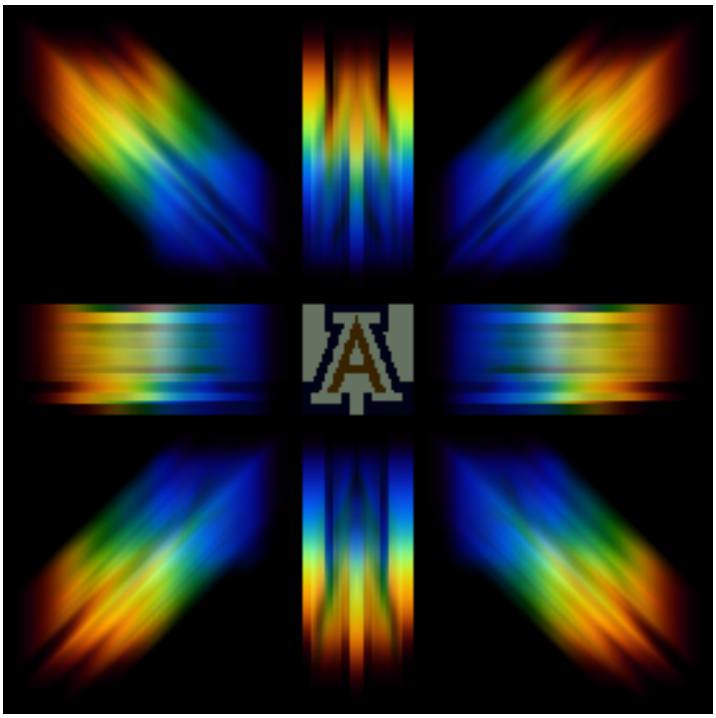

Figure 4.3: The optical image before being sampled by the FPA in the CTIS. The CTIS uses multiple gratings to create projections of the spectral datacube at the FPA. Note the center image is the zeroth order and is simply the undiffracted color image of the object scene [86].

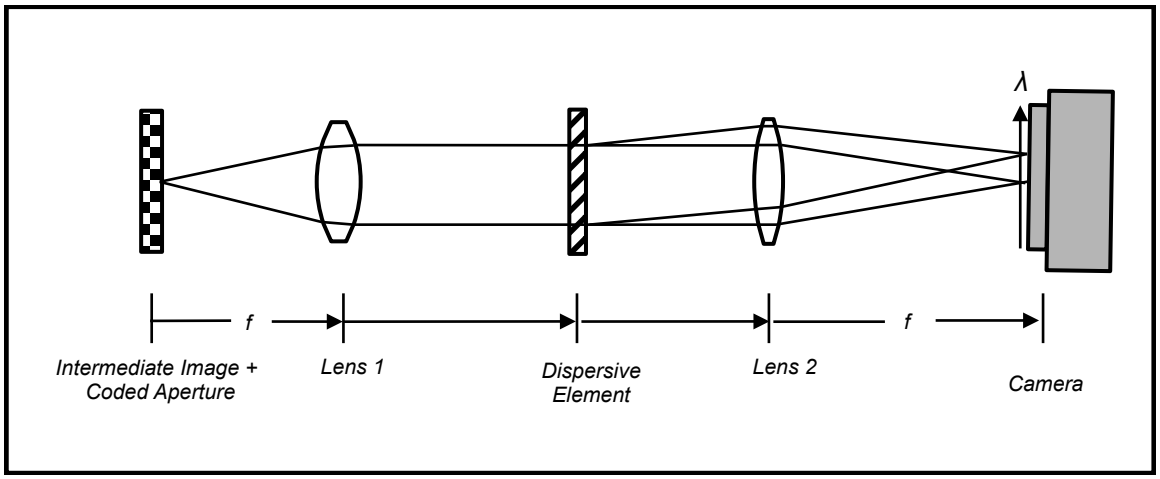

Figure 4.4: The architecture of the Coded Aperture Snapshot Spectral Imaging (CASSI) consists of a collimating lens, a disperive element, an imaging lens, and a focal-plane array (FPA). The coded aperture spatially codes each wavelength layer in the three-dimensional spectral datacube. Then the dispersive element creates a wavelenght depedent spatial shift, shearing the spectral datacube. The monochromatic image of the FPA creates a coded two-dimensional projection of the three-dimensional spectral datacube.

CTIS and the CASSI only reconstruct the spectral datacube [92]. This produces a significant amount of data, which is only used as an intermediate step. One is typically interested in determining the which chemical or material is responsible for a spectrum at a specific location. This is called spectral classification [74, 93, 94]. Therefore, an additional post-processing step needed. If however, one could directly classify the spectrum ,one could signficantly reduce the amount of data storage and communication resources required to operate the instrument.

Previously my colleuges developed a computational spectrometer called the Adaptive Feature Specific Spectrometer (AFSS) [3]. Shown in Figure 4.5, the AFSS was the first experimental computational sensor that make use of an adaptive scheme which uses measurement data to design spectral filters (codes). This allowed the spectrometer to directly classify the chemical or material responsible for the spectrum without the need to perform a reconstruction step. By combining the Fellgett advantage with an adaptive algorithm to create custom spectral filters, the AFSS was able to demonstate significant reduction in the number of measurements to classify a spectrum compared to non-adaptive multiplexed spectrometers in low SNR scenarios. In this context, the spectral filters act as feature vectors, which are computed using variation of PCA.[1]

[1]In the context of the AFSS and AFSSI-C the terms spectral filter, code, and feature vector

The Adaptive Feature Specific Spectral Imaging-Classifier (AFSSI-C) extends the concept first demonstrated by the AFSS to spectral imaging. The AFSSI-C directly classifies the spectrum at each spatial location without the need to reconstruct the spectral datacube. By adopting a task-specific sensing approach, the AFSSI-C greatly improves classification accuracy while simultaneously reduces the amount of bandwidth and storage for data. Furthermore, the architecture of the AFSSI-C leverages both the Fellgett and adaptive measurement scheme like the AFSS, while also adding the Jacquinot advantag to outperform all traditional and currently known computational spectral imaging instruments in terms of number of measurements to correct classifcation.

4.2 Architecture

I want to quickly review the architecture of the Adaptive Feature Specific Spectrometer (AFSS) before discussing the architecture of the AFSSI-C. The AFSS, shown in Figure 4.5, is the earliest known computational spectrometer to use adaptive spectral filters to classify spectra [3]. Unlike the traditional slit spectrometer, the AFSS images the dispersed slit onto a DMD. The mirrors of the DMD can then be adjusted to selectively reflect certain wavelengths towards the condensor lens, which then focuses light onto a single detector element [3]. By reflecting or "turning on" multiple DMD mirrors and only using a single detector element, we achieved the Fellgett advantage. By using measurement data to actively design spectral filters, the AFSS outperforms non-adaptive schemes by eliminating spectra that are improbable and turns it attention towards trying to classify the remaining high probability spectra.

In order to create an imaging version of the AFSS, one may naively attempt to extend the AFSS architecture by forming a parallel array of AFSS sensors to achieve classification across a spatial scene. However, just like in the proposed parallel single-pixel camera design in Chapter 3, this approach would significantly increase the SWAP-C of the design. The AFSSI-C provides a similar feature-based measurement approach and Bayesian framework but with a more compact architecture. Rather than a fully parallel version of the AFSS, the optical design of the AFSSI-C uses a single DMD and a single set of lenses and dispersive elements sharing a common

are synonymous.

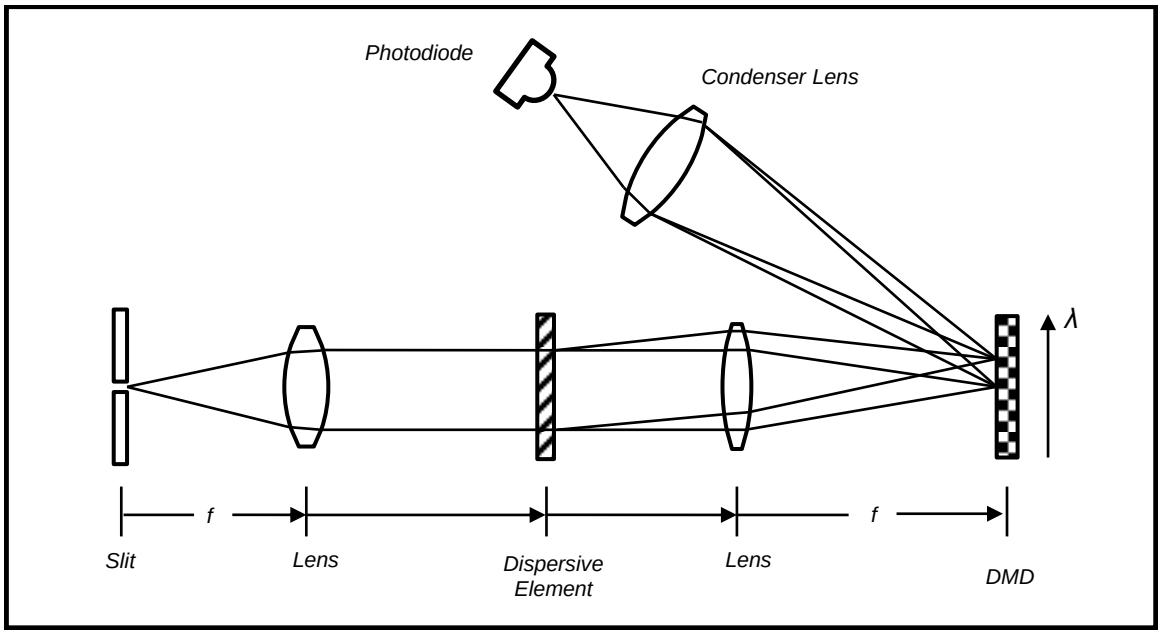

Figure 4.5: The slit blocks light from all but one spatial location. A lens collimates the light passed from the slit and a dispersive element creates monochromatic images of the slit at different columns on the DMD, corresponding to different spectral channels. The DMD can then selectively reflectives combinates of each spectral channels into the condenser lens. The condensor lens focuses all the light onto a single detector element, a photodiode.

entrance pupil. This approach shares resources across multiple spatial locations.

The design of the AFSSI-C is essentially two $4f$ open-aperture monochromators seperated by a Digital Micro-Mirror Display (DMD), shown in Figure 4.6. In our experiment, we used an objective lens, which is not shown, to create an intermediate image at the input plane of the instrument. At this point in the system one can imagine the source spectral datacube as depicted in Figure 4.7(a), where x and y are the spatial axes, and λ the spectral axis. The first lens of the system collimates the light from the input plane. The first grating disperses the light. The second lens then images dispersed copies of the intermediate image on the DMD. Just prior to being reflected from the DMD, one can imagine the spectral dimension of the datacube as being sheared at an angle, in the direction of the dispersion, see Figure 4.7(b). As in the AFSS, each mirror on the DMD either reflects light into the second arm or reflects light away into a beam dump (not shown). One can visualize this by looking at Figure 4.7(c), a mirror that reflects light away, towards the beam dump, deletes columns in the sheared spectal datacube. The light that reflected into the second arm is then collimated before hitting the second grating,

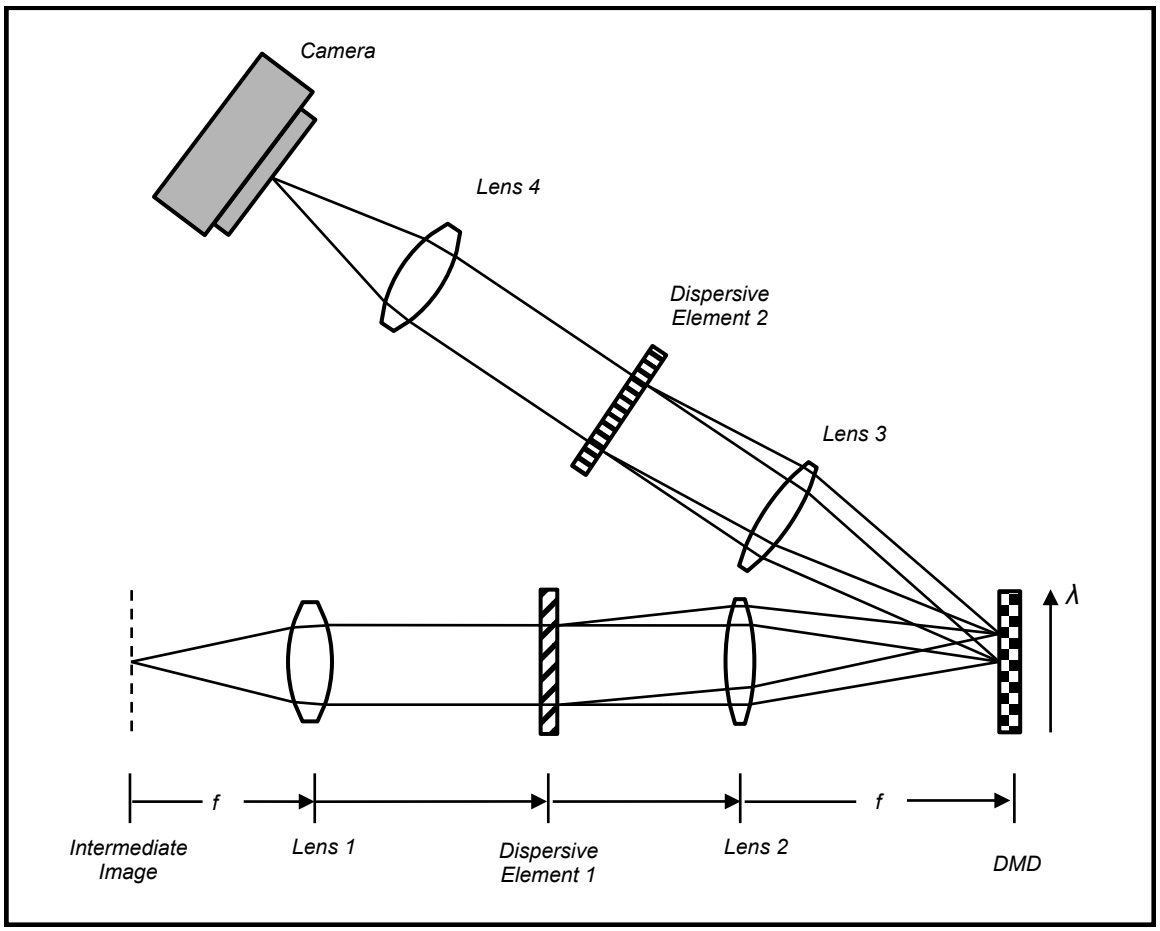

Figure 4.6: An objective lens forms the intermediate image of the object scene at the input plane of the instrument. The first lens collimates the light and a diffraction grating disperses the light. A second lens images spectrally dispersed versions of the image of the object scene onto the DMD. The DMD directs light to a beam dump or reflects the light towards the third lens on a mirror-by-mirror basis. Collimated light from third lens is sent through a second grating, and finally imaged onto the detector by the fourth lens.

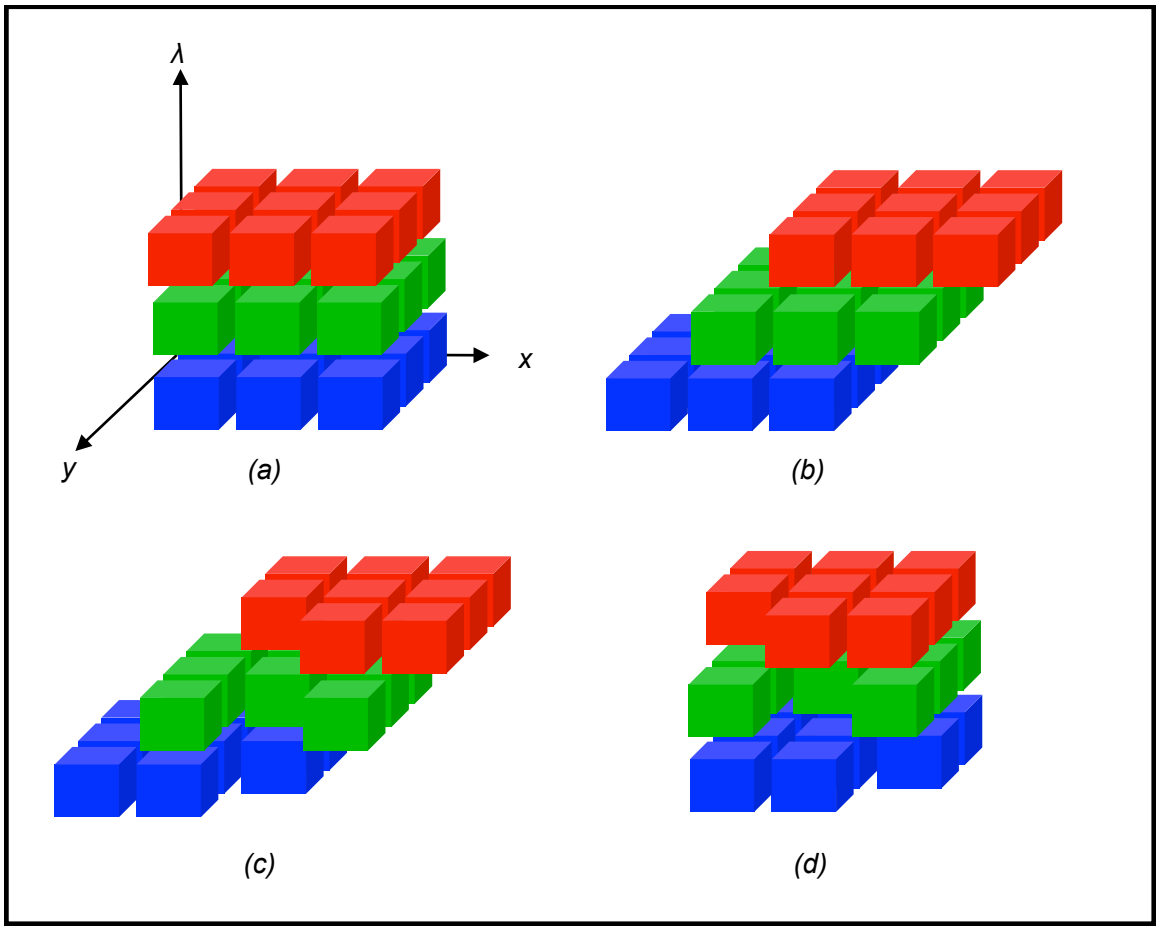

Figure 4.7: (a) The input cube is (b) sheared by the first grating, (c) encoded at the DMD, and finally (d) spatially re-registered by the second grating.

which is identical to the first, but with the reverse dispersion direction. This removes the shear in the encoded spectral datacube, seen in Figure 4.7(d). Lens 4 images the unsheared spectral datacube onto the FPA. The grayscale image of the FPA essentially integrates the spectral datacube over the spectral dimension.

We choose to use a dual-disperser architecture rather than the single-disperser architecture such as the one in the CASSI since the mirror patterns on the DMD acts as spectral filters for each spatial location. In the dual-disperser design, the spectral coding of each spatial location is more straightforward and elegant approach for parallel direct classification: the architectures allows us to write the image of each FPA pixel as an innner product of the spectrum at the corresponding location at the source with the spectral filter created by the DMD, allowing for parallel operation of the Bayesian inference approach. A single-disperser design would entangle both the spectral and spatial dimension of the spectral datacube, and would require solving

a single larger joint inference problem.

4.2.1 Forward Model

Imagine a continuous spectral datacube which we call the source spectral density of $D_0(x, y; \lambda)$ at the input aperture. To simplify the forward model analysis, assume unit magnification and ignore diffraction, aberrations, and vignetting. The spectral density just before reflection from the DMD is written as:

$$\begin{aligned} D_1(x, y; \lambda) &= \iint \delta(x' - [x + \alpha(\lambda - \lambda_c)])\delta(y' - y) D_0(x', y'; \lambda) dx' dy' \\ &= D_0(x + \alpha(\lambda - \lambda_c), y; \lambda) \end{aligned} \tag{4.1}$$

Notice that this can be thought of as a two-dimensional convolution of the spectral density with an wavelength dependant point-spread function (PSF) that shifts the spectral density by an amount $\alpha(\lambda - \lambda_c)$, where α is the dispersion and λ_c is the center wavelength. If there is no dispersion, when $\alpha = 0$, then one would get the original spectral density back. Note, that when $\lambda = \lambda_c$ there is no shift in the x direction.

After reflecting from the DMD the spectral density can then be written as:

$$D_2(x, y; \lambda) = T(x, y) D_1(x, y; \lambda) = T(x, y) D_0(x + \alpha(\lambda - \lambda_c), y; \lambda) \tag{4.2}$$

where $T(x, y)$ represents the reflection pattern of the DMD. If $T = 1$ the light is being reflected into the second arm and if $T = 0$ the light is being reflected towards the beam dump.

Propagating through the second arm has the effect of reversing the shear imposed on the spectral density:

$$\begin{aligned} D_3(x, y; \lambda) &= \iint \delta(x' - [x - \alpha(\lambda - \lambda_c)])\delta(y' - y) D_2(x, y; \lambda) dx' dy' \\ &= T(x - \alpha(\lambda - \lambda_c), y) D_0(x, y; \lambda) \\ &= H(x, y; \lambda) D_0(x, y; \lambda) \end{aligned} \tag{4.3}$$

Notice that the dispersion α has the opposite sign. This equation is represented in discrete form in Figure 4.7(d). Thus, propagating through the entire optical system represents multiplying the input spectral density with a spectral density filter function $H(x, y; \lambda)$.

Since the FPA image is grayscale, ignoring quantization effects, the image can be written as an integral over the wavelengths

$$I(x,y) = \int H(x,y;\lambda) D_0(x,y;\lambda) d\lambda \tag{4.4}$$

By taking into account the spatially pixelated detector array with pixel size Δ, then the discrete FPA image is

$$\Gamma_{nl} = \iiint \text{rect}\Big(\frac{x}{\Delta} - n, \frac{y}{\Delta} - l\Big) H(x,y;\lambda) D_0(x,y;\lambda) dx' dy' d\lambda \tag{4.5}$$

where Γ_{nl} is the image value from the n^{th} and l^{th} location. Now consider that the DMD pattern T is also pixelated with the same pixel size Δ as in the detector.

$$T(x,y) = \sum_{n',l'} T_{n',l'} \text{rect}\Big(\frac{x}{\Delta} - n', \frac{y}{\Delta} - l'\Big) \tag{4.6}$$

Inserting Equation (4.6) into Equation (4.3) and Equation (4.5) produces a single equation which describes how the signal-of-interest $D_0(x,y;\lambda)$ is measured by the AFSSI-C:

$$\begin{aligned}\Gamma_{nl} = \sum_{n'l'} \iiint &\text{rect}\left(\frac{x}{\Delta} - l, \frac{y}{\Delta} - n\right) \text{rect}\left(\frac{x - \alpha(\lambda - \lambda_c)}{\Delta} - l', \frac{y}{\Delta} - n'\right) \\ &\times T_{n'l'} D_0(x,y;\lambda)\, dx\, dy\, d\lambda.\end{aligned} \tag{4.7}$$

This equation maybe somewhat difficult to interpret, so to add some additional intuition I will go through an example of a monochromatic source at the center wavelength, $D_0(x,y,\lambda) = I_0(x,y)\,\delta(\lambda - \lambda_c)$, where I_0 is the intensity distribution of the monochromatic scene. In this case, Equation (4.7) simplifies to

$$\begin{aligned}\Gamma_{nl}(\lambda = \lambda_c) &= \sum_{n'l'} \iiint \text{rect}\left(\frac{x}{\Delta} - l, \frac{y}{\Delta} - n\right) \text{rect}\left(\frac{x}{\Delta} - l', \frac{y}{\Delta} - n'\right) \\ &\qquad \times T_{n'l'} I_0(x,y)\,\delta(\lambda - \lambda_c)\, dx\, dy\, d\lambda \\ &= \sum_{n'l'} T_{n'l'} \iint \text{rect}\left(\frac{x}{\Delta} - l, \frac{y}{\Delta} - n\right) \text{rect}\left(\frac{x}{\Delta} - l', \frac{y}{\Delta} - n'\right) \\ &\qquad \times I_0(x,y)\, dx\, dy \int \delta(\lambda - \lambda_c)\, d\lambda \\ &= \sum_{n'l'} \delta_{ll'} \delta_{nn'} T_{n'l'} I_{nl} \\ &= T_{nl} I_{nl},\end{aligned} \tag{4.8}$$

where I_{nl} is a spatially pixelated version of the monochromatic source with intensity distribution $I_0(x,y)$. One now sees that in the monochromatic case, the measurement is point-by-point multiplication of the DMD pattern with the discrete monochromatic image I_{nl}

In the next example, I consider the case where a monochromatic source is not at the center wavelength but is shifted in wavelength. The wavelength of the monochromatic source is shifted from the center wavelength by a single spectral channel $\lambda = \lambda_c + \Delta_\lambda$, where $\Delta_\lambda = \Delta/\alpha$. The spectral density is now written as $D_0(x,y,\lambda) = I_0(x,y)\,\delta(\lambda - (\lambda_c + \Delta_\lambda))$. Equation (4.7) simplifies to

$$\begin{aligned}
\Gamma_{nl}(\lambda = \lambda_c + \Delta\lambda) &= \sum_{n'l'} \iiint \mathrm{rect}\left(\frac{x}{\Delta} - l, \frac{y}{\Delta} - n\right) \mathrm{rect}\left(\frac{x}{\Delta} - (l'+1), \frac{y}{\Delta} - n'\right) \\
&\qquad \times T_{n'l'} I_0(x,y)\,\delta(\lambda - (\lambda_c + \Delta\lambda))\,dx\,dy\,d\lambda \\
&= \sum_{n'l'} T_{n'l'} \iint \mathrm{rect}\left(\frac{x}{\Delta} - l, \frac{y}{\Delta} - n\right) \mathrm{rect}\left(\frac{x}{\Delta} - (l'+1), \frac{y}{\Delta} - n'\right) \\
&\qquad \times I_0(x,y)\,dx\,dy \int \delta(\lambda - (\lambda_c + \Delta\lambda))\,d\lambda \\
&= \sum_{n'l'} \delta_{l(l'+1)} \delta_{nn'} T_{n'l'} I_{nl} \\
&= T_{n(l-1)} I_{nl} \qquad (4.9)
\end{aligned}$$

This shows that a shift by one spectral channel results in a shift of one pixel in of the DMD pattern.

Using the intuition from the last two examples, I will now extend this to a non-monochromatic case. Consider spectral channel index c out of N_λ total spectral channels.

We can also define a discretized source spectral datacube D_{nlc}, and then the detector signal Γ_{nl} as a result of mirror pattern T acting on the pixelated source is

$$\Gamma_{nl} = \sum_{c=0}^{N_\lambda - 1} T_{n(l+c)} D_{nlc}, \qquad (4.10)$$

which shows the measurement at each pixel being the inner product of the source spectrum and the spectral filter which results from the mirror pattern. If one inspects the adjacent pixel on the detector, $\Gamma_{n(l+1)}$ we find that

$$\Gamma_{n(l-1)} = \sum_{c=0}^{N_\lambda - 1} T_{n(l+1+c)} D_{n(l+1)c}. \qquad (4.11)$$

The spatial location (n, l) sees the effect of the pattern in mirror locations T_{nl} to $T_{n(l+N_\lambda-1)}$, while the neighboring spatial location at $n, (l+1)$ is encoded by the mirror pattern from $T_{n(l+1)}$ to $T_{n(l+N_\lambda)}$. Notice that given two neighboring spatial pixels in the input aperture, l and $l+1$, mirror pixels $T_{n(l+1)}$ to $T_{n(l+N_\lambda-1)}$ are common to both locations.

Now the design constraint of the AFSSI-C architecture is clear, while a naively parallel set of AFSS would have been costly in terms of SWAP-C, using a common entrance pupil prevents independent spectral filters for each spatial location. The spectral filters imposed by the DMD pattern maybe unique for every spatial location, but *cannot* be implemented independently, requiring the features to be designed *jointly* for spatial locations along a row (the l direction).

4.3 Adaptive Classification Algorithm

As mentioned earlier, the AFSSI-C uses adaptive features to directly classify the spectrum at each spatial location. At each measurement step m, the classification algorithm computes the probability of each hypothesis spectra in a spectral library. Building on the discussion of Principal Component Analysis (PCA) and Bayesian statistics in Chapter 2, I will discuss how the classification algorithm of the AFSSI-C works.

Since the AFSSI-C feature vectors are not independent for each spatial location, it is more intuitive to discuss the algorithm when only a single spatial location exists in the spectral datacube. The reader can then expand the intuition to the entire scene. For a single pixel architecture, one can write the discrete forward model as

$$g_m = \mathbf{t}_m \cdot \mathbf{f} \tag{4.12}$$

where g_m is the m^{th} measurement of DMD mirror pattern $\mathbf{t}_m$ and $\mathbf{f}$ is the ground truth spectrum of the source.

The measurement from the detector is compared to the spectral library, subject to the spectral filter implemented at that spatial location. A probability is assigned to each hypothesis in the spectral library using Bayes' theorem. The conditional probability of the hypothesis, which is spectrum $\mathbf{s}_i$ is present, given the measurement

history up to the m^{th} measurement $\{g\}_m$, can be written as:

$$\mathrm{P}\left(h_i|\{g\}_m\right) = \frac{\mathrm{P}\left(\{g\}_m|\, h_i\right)\ \mathrm{P}\left(h_i\right)}{\mathrm{P}\left(\{g\}_m\right)}. \tag{4.13}$$

Remember from Chapter 2, that PCA provides the optimal basis to distinguish between data vectors in a vector space by diagonalizing the covariance matrix. The covariance matrix (or scatter matrix) is simply the matrix multiplication of the spectral library (with the mean spectrum subtracted) with the transpose of itself.

$$\begin{aligned}\mathbf{Q} &= \sum_{r\,=1}^{N_R} \left(\mathbf{s}_r - \bar{\mathbf{s}}\right)\left(\mathbf{s}_r - \bar{\mathbf{s}}\right)^T \\ &= \mathbf{X}\,\mathbf{X}^T.\end{aligned} \tag{4.14}$$

However, since the spectral library is constant with respect to the measurements, the set of principal components will also remain constant. Intuitively, one can use the probability of each spectrum that was computed from the last measurement to weight each of the spectra in the library. Geometrically, one can imagine that the direction of largest variation is now biased to align in the direction with higher probability spectra. The geometric depiction of the probabilistically weighted Principal Component Analysis (PCA) which we call pPCA is shown in Figure 4.8.

I will now go over the formalism for probabilistically weighted Principal Component Analysis (pPCA). At each spatial location, we create a covariance matrix with N_R spectral hypotheses, with individual spectrum $\mathbf{s}_r$, weighted in relation to the prior probability associated with each hypothesis given the measurement history. The individual covariance matrices $\mathbf{Q}(m)$ at each spatial location with spectral library element hypothesis h_r take the form

$$\begin{aligned}\mathbf{Q}(m) &= \sum_{r\,=1}^{N_\lambda} \mathrm{P}\left(h_r|\{g\}_m\right)\left(\mathbf{s}_r - \bar{\mathbf{s}}\right)\left(\mathbf{s}_r - \bar{\mathbf{s}}\right)^T \\ &= \mathbf{X}(m)\,\mathbf{X}^T(m).\end{aligned} \tag{4.15}$$

Here $\mathbf{X}(m)$ is the matrix of weighted spectral elements, where the row index i is from 1 to the number of spectral channels N_λ; the columns r from 1 to N_R as follows,

$$\mathbf{x}_r(m) = \sqrt{\mathrm{P}\left(h_r|\{g\}_m\right)}\left(\mathbf{s}_r - \bar{\mathbf{s}}\right), \tag{4.16}$$

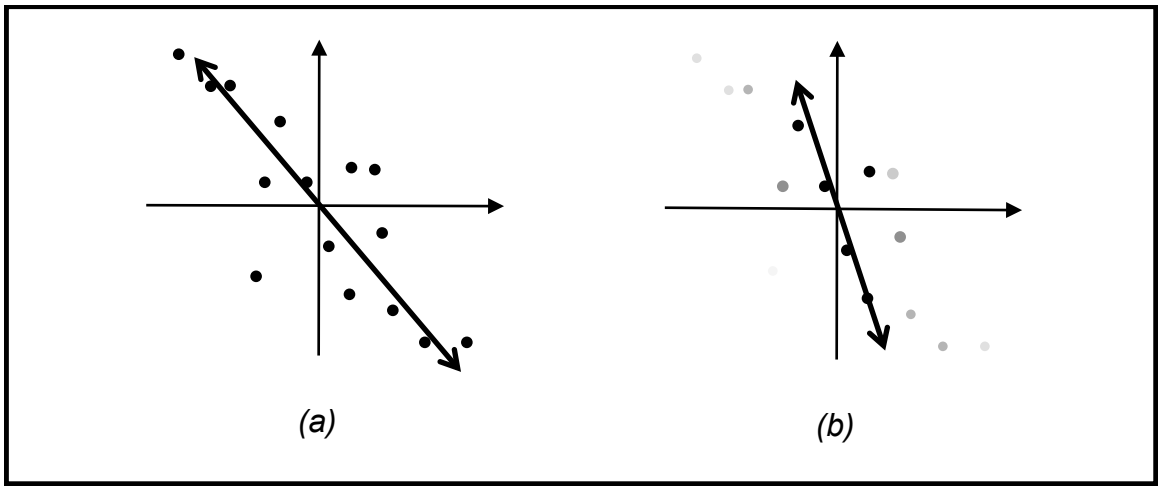

Figure 4.8: Depiction of the pPCA (simple 2D example). (a) First principal component before a measurement has been made. All of the hypotheses are equiprobable, depicted here as all points having the same grayscale value. (b) After a measurement has been made: the darker points are more probable hypotheses, with the less probable hypotheses taking on lighter shades of gray. The first principal component has now been shifted to the direction of greatest variation in the weighted data.

and $\bar{\mathbf{s}}$ is the probabilistically-weighted sum of the spectral library:

$$\bar{\mathbf{s}} = \sum_{r=1}^{N_R} \mathrm{P}\left(h_r|\{g\}_m\right)\mathbf{s}_r. \tag{4.17}$$

We can then compute the first eigenvector of $\mathbf{Q}(k)$, analogous to PCA. We call this probabilistic-PCA or pPCA, and in the case of the AFSS this eigenvector becomes the feature for the next measurement. The scatter matrix $\mathbf{Q}(k)$ is then updated with every measurement.

4.3.1 Updating Probabilities

In practice we do not directly compute the posterior probabilities for each hypothesis using Bayes' theorem, Equation (4.13), since we have no meaningful way of computing the probability of the measurement history $P(\{g\}_m)$. Thankfully, the method of Maximum A Posteriori (MAP) and taking ratios of posterior probabilities that I discussed in Chapter 2 allows one to compute relative probabilities which will be used to weight the spectral library after each measurement. For now I will define the ratio of the posterior probabilities

$$L_{ij}^{\{g\}_m} := \frac{\mathrm{P}(h_i|\{g\}_m)}{\mathrm{P}(h_j|\{g\}_m)} = \frac{\mathrm{P}(\{g\}_m|h_i)}{\mathrm{P}(\{g\}_m|h_j)}\frac{\mathrm{P}(h_i)}{\mathrm{P}(h_j)} \tag{4.18}$$

where $\{g\}_m$ is the set of all measurements, including the current measurement number index m.

Note that if we assume that the likelihood probabilities are independent, then we can write the joint probability as a product of the likelihood of the current measurement with the likelihoods of all past measurements

$$\begin{aligned} P(\{g\}_m|h_i) &= P(g_m|h_i)P(\{g\}_{m-1}|h_i) \\ &= P(g_m|h_i)\prod_{m'=1}^{m-1} P(g_{m'}|h_i) \end{aligned} \tag{4.19}$$

So we can expand Equation (4.18)

$$L_{ij}^{\{g\}_m} = \frac{P(g_m|h_i)\prod_{m'=1}^{m-1} P(g_{m'}|h_i)}{P(g_m|h_i)\prod_{m'=1}^{m-1} P(g_{m'}|h_j)}\frac{\mathrm{P}(h_i)}{\mathrm{P}(h_j)} \tag{4.20}$$

If I define

$$L_{ij}^{\{g\}_{m-1}} := \frac{\prod_{m'=1}^{m-1} P(g_{m'}|h_i)}{\prod_{m'=1}^{m-1} P(g_{m'}|h_j)}\frac{\mathrm{P}(h_i)}{\mathrm{P}(h_j)} \tag{4.21}$$

then Equation (4.20) simplifies into

$$L_{ij}^{\{g\}_m} = \frac{P(g_m|h_i)}{P(g_m|h_j)}L_{ij}^{\{g\}_{m-1}} \tag{4.22}$$

This equation provides a convenient way to update the ratio of posterior probabilities after each measurement.

The question now turns to how does one actually compute the likelihoods. For this step we assumed that the likelihood takes on a Gaussian noise model with standard deviation of the noise σ. Thus we say the probability of the measurement g_m given that hypothesis h_i is true is

$$P(g_m|h_i) = \frac{1}{\sqrt{2\pi\sigma^2}}\exp\left[-\frac{(g_m - \mathbf{t}_m \cdot \mathbf{s}_i)}{2\sigma^2}\right] \tag{4.23}$$

where $\mathbf{t}_m$ is the spectral filter at the measurement step m. This equation means that when the measurement value g_m is close to the inner product of the spectral filter and the hypothesis spectrum, the probability is large. As the measurement value deviates from $\mathbf{t}_m \cdot \mathbf{s}_i$ the probability decreases.

After each measurement we compute Equation (4.22) for all pairs of spectra i, j using the noise model Equation (4.23). In practice, the exponentials of numbers of moderate size cause numerical issues. To avoid this, we compute the logarithm of the likelihood (log-likelihood) ratios to eliminate the exponentials. In Appendix E, I will discuss the update rule using log-likelihood ratios.

4.3.2 Extension to Spectral Imaging

In the case of spectral classification for only a single spatial location, like the AFSS, the number of DMD mirrors (that are used) is equal to the number of spectral channels.

$$N_d = N_\lambda \tag{4.24}$$

As we saw before the dimensionality of the feature vectors, produced by taking the eigenvectors of the covariance matrix, is equal to the number of spectral channels.

In the AFSSI-C architecture, light from spatial pixel l is dispersed onto DMD pixels l to $l + N_\lambda - 1$. While light from the next spatial pixel $l + 1$, is dispersed onto DMD pixels $l + 1$ to $(l + N_\lambda)$. Therefore adjacent spatial pixels have $N_\lambda - 1$ DMD pixels in common. Light is not dispersed between rows. This means that one can treat the design of the features between different rows separately. However, within a particular row, this dispersion prevents independent features at each spatial pixel location on the row. This is one of the major design constraints in the AFSSI-C.

It turns out, one can still use pPCA by constructing a very large data matrix $\tilde{\mathbf{X}}$. The individual probabilistically weighted spectral library matrices $\mathbf{X}_l$ for each spatial pixel l is placed inside $\tilde{\mathbf{X}}$. The adjacent spatial location will have $\mathbf{X}_{l+1}$ placed next to it but one row down. The first row is padded with zeros.

For example, imagine two adjacent spatial pixels $l = 1$ and $l = 2$ with five spectral channels $N_\lambda = 5$ and three spectra in the spectral library $N_R = 3$.

$$\tilde{\mathbf{X}} = \begin{bmatrix} x_{1,1,1} & x_{1,2,1} & x_{1,3,1} & 0 & 0 & 0 \\ x_{2,1,1} & x_{2,2,1} & x_{2,3,1} & x_{1,1,2} & x_{1,2,2} & x_{1,3,2} \\ x_{3,1,1} & x_{3,2,1} & x_{3,3,1} & x_{2,1,2} & x_{2,2,2} & x_{2,3,2} \\ x_{4,1,1} & x_{4,2,1} & x_{4,3,1} & x_{3,1,2} & x_{3,2,2} & x_{3,3,2} \\ x_{5,1,1} & x_{5,2,1} & x_{5,3,1} & x_{4,1,2} & x_{4,2,2} & x_{4,3,2} \\ 0 & 0 & 0 & s_{5,1,2} & s_{5,2,2} & s_{5,3,2} \end{bmatrix} \tag{4.25}$$

The subscripts of element $x_{c,r,l}$ refer to c the spectral channel, the r^{th} spectrum in the spectral library and l the spatial pixel index. Therefore we see that the first three columns are just $\mathbf{X}_1$, the modified spectral library for spatial pixel $l = 1$ and the last three columns is the modified spectral library $\mathbf{X}_2$ for spatial pixel $l = 2$.

Notice that the rows of this matrix have physical meaning. The first row corresponds to the first DMD pixel, since only spectral channel one from all three possible

spectra from location $l = 1$ can be imaged onto the first DMD pixel (mirror). The second row corresponds to the second spectral channel of the three possible spectra from location $l = 1$ and the first spectral channel of the three possible spectra from location $l = 2$.

The revised covariance matrix is now

$$\tilde{\mathbf{Q}} = \tilde{\mathbf{X}}\tilde{\mathbf{X}}^T \tag{4.26}$$

Where $\tilde{\mathbf{X}}^T$ is the transpose of the matrix $\tilde{\mathbf{X}}$. Thus the columns of $\tilde{\mathbf{X}}^T$ have the same physical interpretation as the rows of $\tilde{\mathbf{X}}$. Each element of $\tilde{\mathbf{Q}}$ is therefore the covariance of each of the possible spectral channel from each location onto each DMD mirror. Thus pPCA is computing the eigenvector of the covariances in the "DMD mirror space". This is what we call the joint-pPCA, basically the extension of pPCA to the imaging problem.

4.4 Experiments

A systems level flowchart is shown in Figure 4.9 which shows each of the major steps in running the AFSSI-C experiment. In this section, I will discuss each step, with the goal of continuing to concentrate on the practical considerations of the AFSSI-C. The experimental results in this dissertation are for a 4-class problem, with an input spectral datacube of 64 $\times$ 64 spatial pixels and 38 spectral channels. Figure 4.10 and Figure 4.11 are depictions of the associated 4-class spectral library, with the spectral source displayed on the LED monitor. While this spectral datacube is small compared to those used in remote sensing, the implemented size was driven by practical considerations regarding the source, DMD and detector on hand. The dual-disperser architecture, in general, places no significant limitation on possible datacube sizes. Similarly, the processing involved in the Bayesian inference and feature design are computationally lightweight and do not create a computational limit on datacube size.

Classification difficulty is quantified as the Task Signal-To-Noise Ratio (TSNR) which considers the noise in the system relative to the separation distance between hypotheses spectra. We define Task Signal-To-Noise Ratio (TSNR) as

$$\text{TSNR} = 10 \log_{10}\left(\frac{d_{min}}{\sigma}\right) \tag{4.27}$$

For our experiment, the noise is approximately Additive white Gaussian noise (AWGN) distributed with standard deviation σ. The minimum Euclidean distance between the hypotheses in the spectral library d_{min}. When using this definition for TSNR, a value of 0 dB TSNR is the point where the noise is equal to the minimum distance between the library elements.

4.4.1 Hardware

I will now discuss the hardware used in the AFSSI-C experiment, shown in Figure 4.12. Specifically I will describe some of the design decisions and consequences of those decisions in the AFSSI-C experiment. To avoid expensive and time consuming optical design and fabrication, the lenses were restricted to commercial off-the-shelf achromatic doublets. A compact, fixed focal length, lens (Edmund Optics

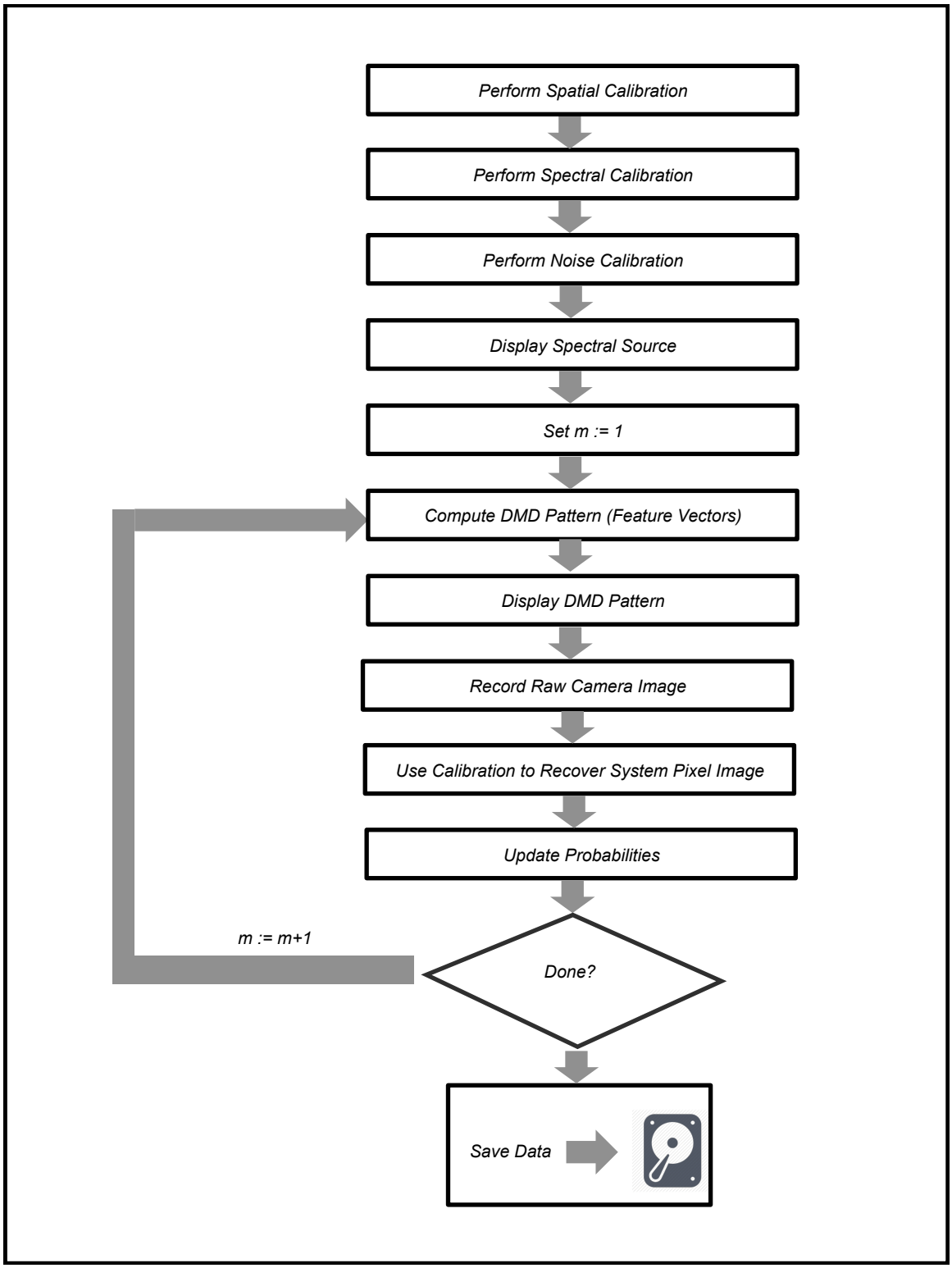

Figure 4.9: Systems Level Flowchart of the AFSSI-C Experiment

Stock No. 58-001, f = 12mm) is used as an objective lens, relaying the object onto an intermediate image plane. A fast entrance optic is needed to meet the layout

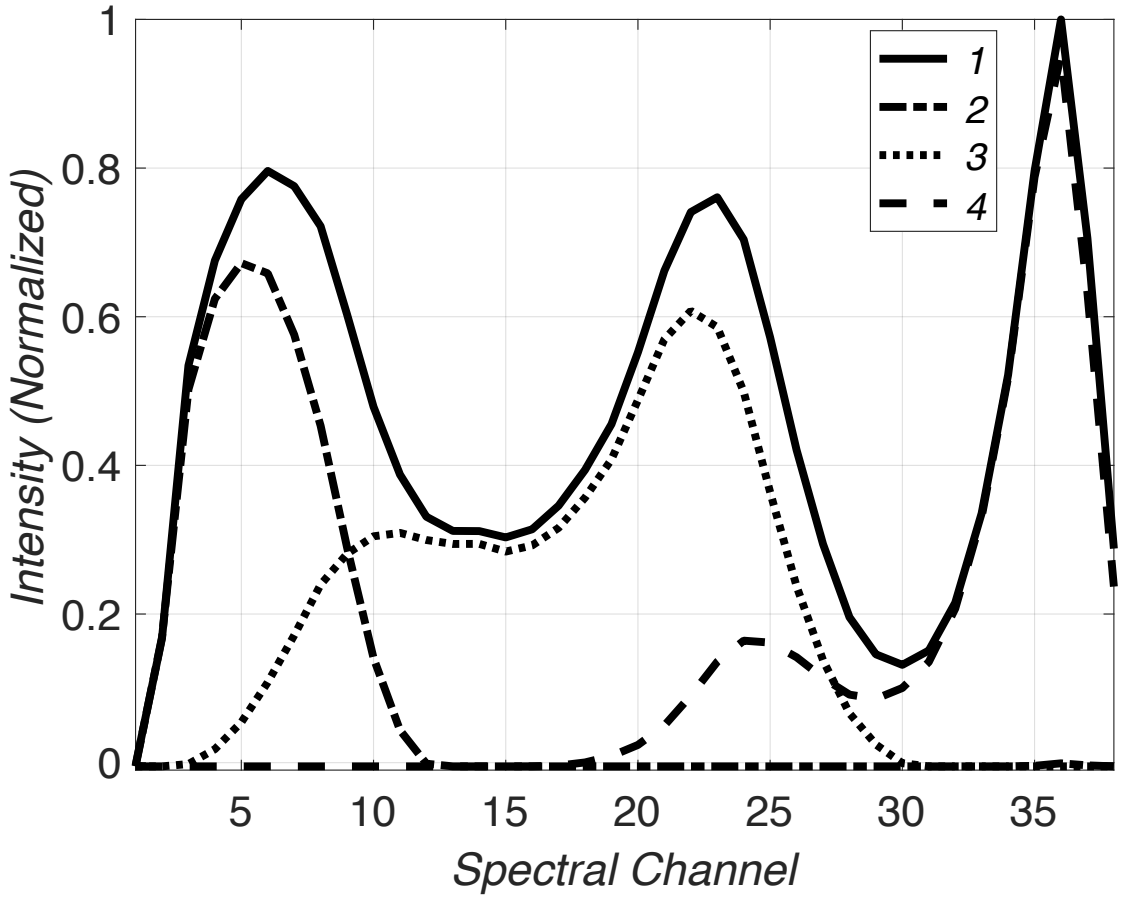

Figure 4.10: The spectral library consists of four spectral classes: the white, red, green, and blue of the LED monitor.

and magnification requirements. Referring to Figure 4.5, lenses 1 and 4 are 75 mm focal length achromatic doublets with a 50 mm diameter (Edmund Optics Stock No. 49-291), and lenses 2 and 3 are 150 mm focal length achromatic doublets with 25 mm diameter (Edmund Optics Stock No. 47-643).

For a repeatable, programmable synthetic datacube source, we used an LED display (Dell P2311H LED monitor with 248 µm pixel pitch). Though this limits the spectra we were able to generate, it still allows for programmable spectra at every spatial location. The center 1080 $\times$ 1080 pixels are used to generate the source spectral datacubes. Spectra consist of combinations of the RGB monitor colors.

The AFSSI-C operated at a spectral range of 425-625 nm, which is approximately the visible wavelength range. We placed thin-film spectral filters in the collimated space to attenuate light outside of this range. The system is designed for $N_\lambda = 38$

Figure 4.11: Four class spectral source used for the AFSSI-C experiment.

spectral channels, resulting in roughly 5 nm/spectral channel. The system magnification to the DMD, which then dictates the lateral spread allowed per spectral channel, requires custom 0.10 lines µm holographic blaze gratings as the dispersive elements, fabricated by Wasatch Photonics. The gratings are designed to minimize diffraction in all the orders except the first order.

The DMD used in our experimental prototype was manufactured by Texas Instruments, it is a DLP Discovery 4100 DMD development kit, with the DLP9500 0.95" 1080p DMD chipset. The DMD has 1080 × 1920 10.8 µm mirrors on the array, and the mirrors pivot 12 degrees on a diagonal axis. The DMD is oriented such that the bottom of the array is parallel to the optical table, which forces the second arm of the AFSSI-C to rise off the optical table at an angle of 17.5 degrees; this orientation was chosen so that the mirrors would be oriented as squares rather than diamonds. While the periodic structure of the DMD can produce strong diffraction for certain types of illumination, no diffraction effects are observed in the AFSSI-C as the light incident on the DMD is incoherent.

A Santa Barbara Instrument Group (SBIG) ST-10XME detector with a 2184 ×

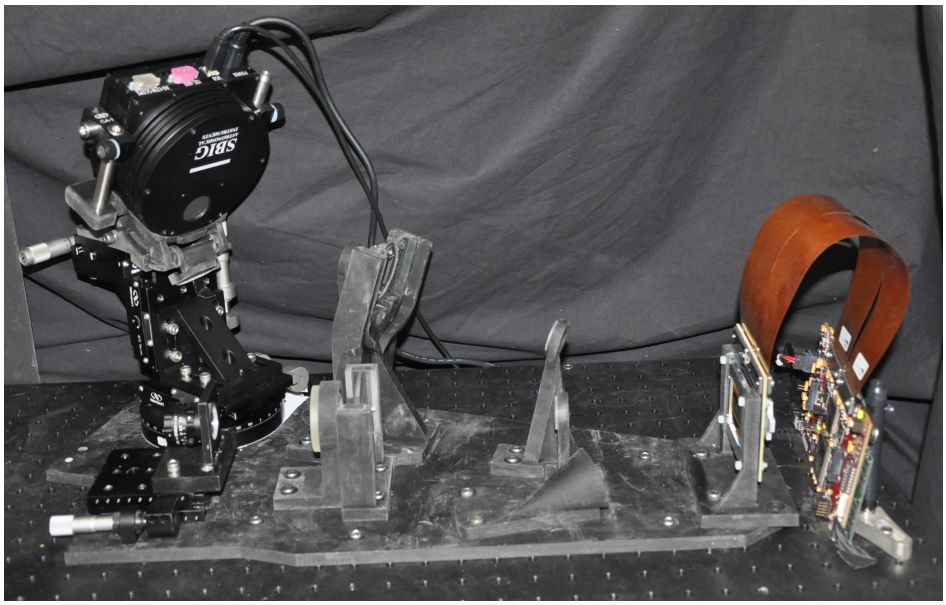

Figure 4.12: Photo of AFSSI-C.

1472, 6.8 µm pitch CCD as the camera. Due to the angle of the optical axis of the second arm of the AFSSI-C, a goniometer was constructed using a rapid prototyping 3D printer to facilitate easy alignment of the SBIG camera [95].

Once the optical layout was finalized, the physical system was designed in SolidWorks. The lens mounts, detector mount, DMD mount, beam dump, and locating plates were then fabricated on a Eden 350 rapid prototyping 3D printer.

Light baffling is utilized to shield the system from ambient light as well as stray light within the system. A baffling structure was designed and printed to surround the detector, with smaller baffling elements to attenuate stray light around the optical paths.

4.4.2 Implementing Codes

One of the practical considerations of the coding scheme in the AFSSI-C is that the pPCA algorithm generates both positive and negative elements in the feature vectors. These are analogous to having positive and negative weights in the weighing problem. Unfortunately, with incoherent light, one is unable to make simultaneous positive and negative measurements because we cannot easily manipulate the phase of the electric field. Therefore, in the AFSSI-C we are forced to take two camera exposures for each measurement step and subtract the negative set of measurements

from the positive set of measurements. This means that an additional noise term is added to each measurement step. For the rest of this chapter, when I say measurement or measurement step, I really mean one measurement step which records two camera exposures.

Another practical issue is that the DMD we used only allows a binary mode of operation. Either the mirror reflected the light towards the second arm or it reflected light towards the beam dump. In practice any positive element is set to 1 and any negative element is set to -1.

4.4.3 Calibration

One reoccurring topic in this dissertation is calibration. In computational sensing the measurement may not look anything like the signal-of-interest. Just like in the Static Computational Optical Undersampled Tracker (SCOUT), the AFSSI-C relies heavily on several calibration procedures which involves spatial, spectral, and noise measurements for optimal classification performance.

4.4.3.1 Spatial Calibration

One of the major opto-mechanical issues in the AFSSI-C is caused by the geometry of the DMD with respect to the second arm. The DMD plane is normal to the optical axis of the first arm, but it is at an angle (equal to twice the tilt of the mirrors on the diagonal) to the second arm. One can think of the DMD plane as the object plane for the second arm of the system. Since the DMD is tilted, this creates a Scheimpflug distortion: a tilted object plane images to tilted image plane [96].

Because of Scheimpflug distortion, the image of the monitor physical pixels are not aligned to the rectangular grid of pixels on the FPA, see Figure 4.13. Even worse there is not a one-to-one relationship of FPA pixels to object scene pixels since the magnification is not constant over the FOV. In order to produce optimal performance we need to know the mapping of monitor to FPA pixel locations. The goal of spatial calibration is to make a set of measurements in which we can infer a parameterized affine transform from FPA physical pixel coordinates to monitor physical pixel coordinates.

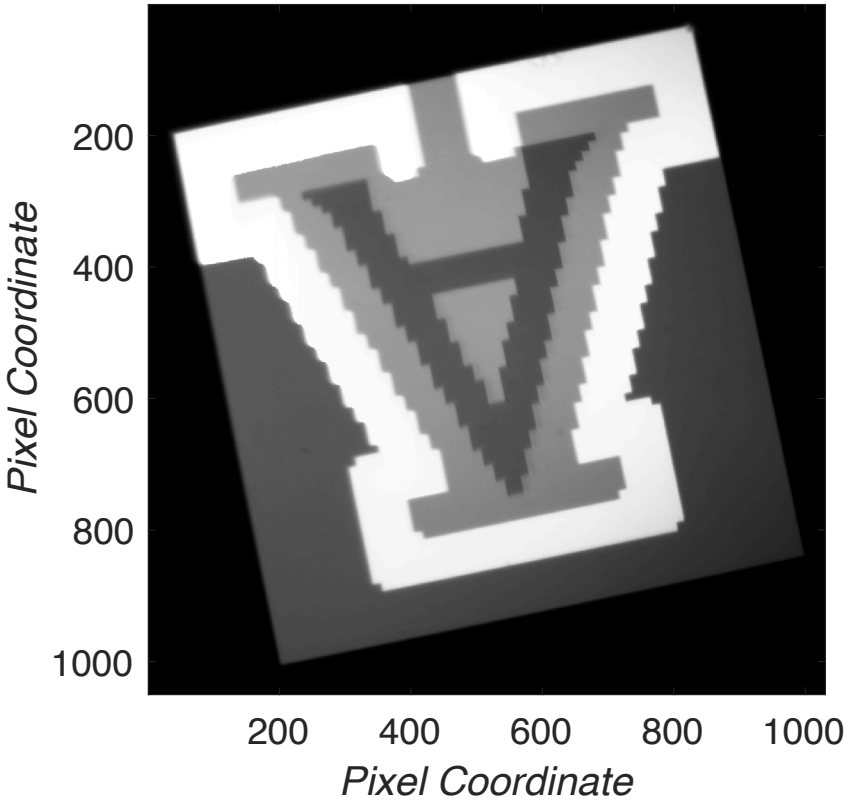

Figure 4.13: The image produced by the camera without any post-processing.

Due to spatial resolution constraints and other practical issues, we grouped several physical pixels together in a system pixel (SP). For example, in our experiment we have an effective resolution of 64 × 64 system pixels. However, on the monitor a system pixel (SP) actually consists of 16 × 16 physical pixels on the source. Similarly, on the DMD a system pixel (SP) consists of 8 × 8 physical pixels (mirrors) and on the detector a SP consists of 12 × 12 physical pixels.

A straightforward approach to inferring the relationship between FPA and object scene pixels would be to energize each physical pixel on the Region of Interest (ROI) one at a time and record the image on at the physical pixel. We could then create a look-up table for each FPA pixel to infer where from the LED this came from. However, energizing each of the 1024 × 1024 physical pixels at 5 seconds per exposure will take approximately 61 days. We are therefore forced to infer the spatial transform from a small number of physical pixels images.

At this point we are only concerned with how the the LED monitors appear on the FPA image. The alignment of the DMD does affect the image quality and

location, but the pattern of the DMD only affects the radiance of an image point. Therefore, we set the entire DMD mirror pattern to reflect towards the second arm. In this state, the DMD acts like a normal mirror.

First the monitor displays a grid of white dots, Figure 4.14. The number of dots can be then be increased for better accuracy but increases the calibration time. The list of monitor physical pixel coordinates where the center of each dot appeared on the monitor is then saved.

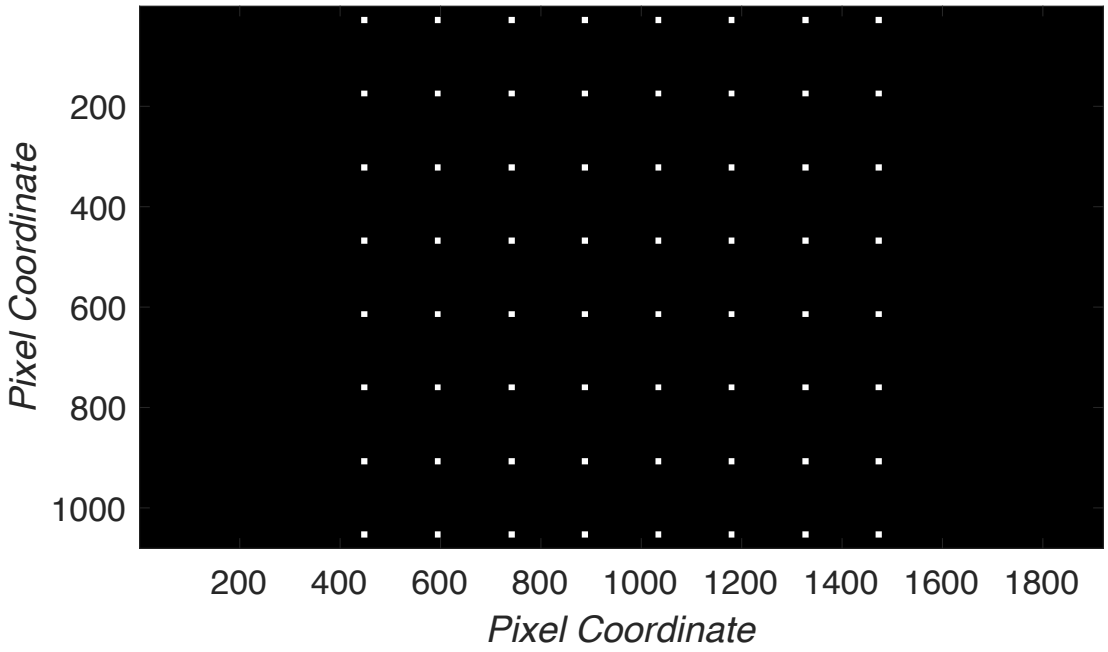

Figure 4.14: An array of white dots is displayed on the monitor as part of the spatial calibration. The image of this grid is recorded, the centroids are estimated and a spatial transform is inferred between the physical pixel coordinates of the LED monitor and the FPA

A raw camera image is captured and stored, see Figure 4.15. To account for hot pixels and systematic error in the system the camera image of the all black monitor is subtracted from the camera image of the array of white dots which gives us the dark subtracted image of white dots. The dark subtracted detector image is then thresholded to prevent noisy pixels from being counted as one of the images of the white dots from the monitor. All values above the threshold is set to 1 while all values below the threshold is set to 0.

Using a built-in MATLAB `regionprops` function with the optional `centroid` argument, we can infer the center pixel, centriod, for each of the white dot images

in the thresholded image. This function generates a list of pixel coordinates from the thresholded image. Each pixel coordinate is associated with the center of each white dot in the image. Often we get a list of coordinates with a length that does not match the number of white dots displayed onto the monitor. At this point, we may need to change the threshold value to return the same number of coordinate pairs as dots displayed on the monitor. For example if we have an 8×8 grid of dots we expect a list of 64 detector pixel coordinate pairs. As a sanity check, we can create an image of the where the inferred centriods are and overlay it on thresholded image of the grid of white dots. The image of the centriods should be approximately on top of the image of each white dot, see Figure 4.16.

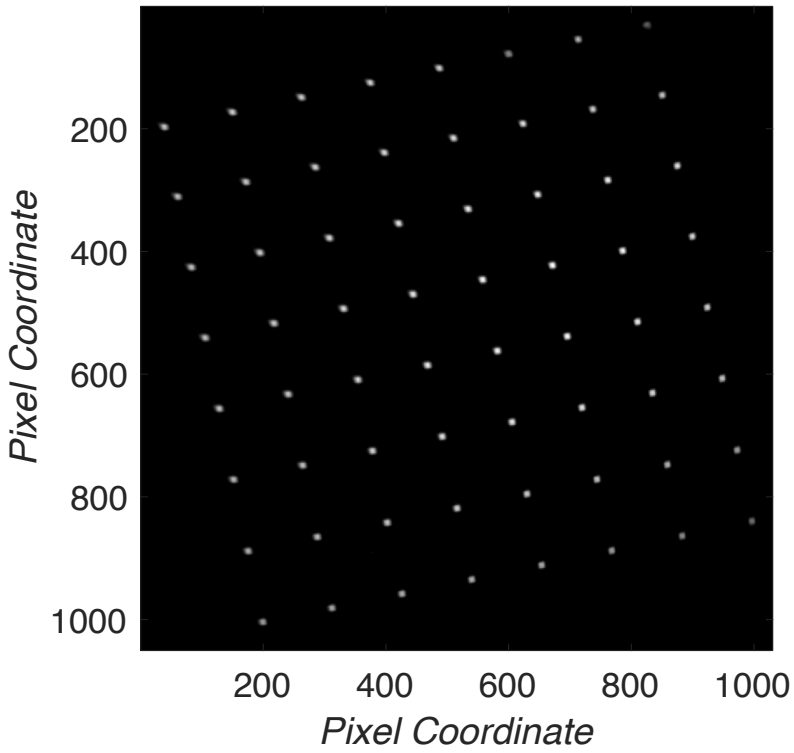

Figure 4.15: The detector image of the array of white dots as part of the spatial calibration

Each dot is added to the monitor scene one at a time to prevent accidentally associating the wrong dot on the image. Doing this each time the AFSSI-C is calibrated is time consuming, since we need at least as many FPA exposures as there are dots in the object scene. In practice, we use MATLAB to look up the locations from the last spatial calibration and if we believe the instrument has not been misaligned since then, we look in a neighborhood of pixels around the last dot

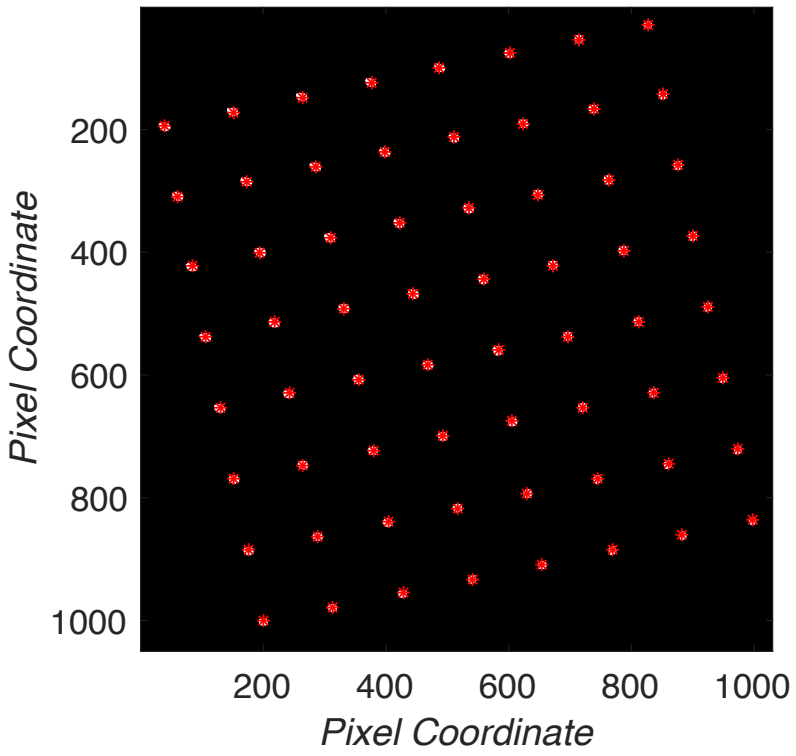

Figure 4.16: The detector image of the array of white dots as part of the spatial calibration

locations and compute the centroids of the dots, see Figure 4.17.

Now that we have a list of dot centers in the monitor pixel coordinates and a list of dot centers in the detector pixel coordinates, we use the MATLAB function `cp2tform`, which takes the list of control points and uses them to infer the spatial transform. This transform is a parameterized relationship between every single physical pixel coordinate on the monitor and physical pixel coordinate on the detector.

Each SP contains 16×16 physical pixels, using the spatial transform, we known which detector pixels these physical pixels will image to. The average value of all the physical pixels from the FPA image correspond to the SP value. Thus we can reverse the effects for any image rotation or translation at the FPA plane.

As part of the calibration procedure, an image is saved by turning on the entire DMD and measuring the intensity at each spatial location with the monitor set to white at full intensity. One can think of this image as an intensity map, providing information as to which locations have more throughput than others in the FOV. During the classification operation, each camera image is normalized by this intensity

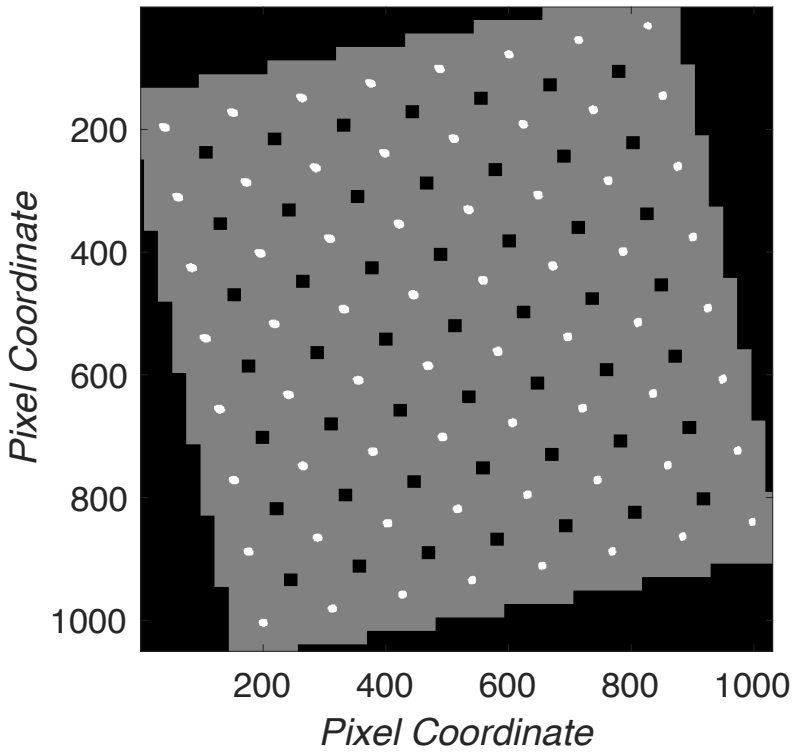

Figure 4.17: Instead of displaying the dots one at a time, we use the location of the dot images from the last spatial calibration and search in a rectangular region around those coordinates for the new dots. This greatly speeds up the spatial calibration since each exposure is approximately 3-5 seconds.

map, while the spectral library at each location is normalized by the integration of the white spectrum (which has the system spectral response information folded into it), allowing the classification decisions to be made.

4.4.3.2 Spectral Calibration

Unfortunately, due to vignetting and imperfections in the gratings the spectral response of the AFSSI-C, the combined optical system produces a shift-invariant spectral response. To correct for this we developed a calibration procedure to measure how the spectral response of the optical system varies over the region-of-interest in the FOV.

A flowchart of the spectral calibration is shown in Figure 4.19. Remember that in the AFSSI-C architecture, the first column of mirrors on the DMD reflects light corresponding to the first spectral channel of the first spatial column. The second column of mirrors reflects light corresponding to the first spectral channel of the second spatial column and also reflects the light in spectral channel 2 of the first

spatial column, and so on, see Figure 4.18. Also, the dual-disperser architecture of the AFSSI-C means that light from a single spatial pixel images to a single spatial pixel on the FPA, regardless on what pattern is being displayed on the DMD. In other words, in a single camera image we can measure the intensity of different spectral channels of adjacent spatial locations along a row by turning on only one DMD mirror. Therefore, we can turn on an entire row of pixels in the monitor and realize that different FPA images correspond to different spectral channels for each spatial location.

Also remember that the dispersion is only in the horizontal direction, along the columns of the DMD. Therefore, we can set the entire ROI to display the same spectrum. Then sweep the entire column of the DMD. In our experiment, the number of effective DMD SP columns is equal to the number of monitor SP in a row plus the number of spectral channels minus one

$$N_{d_x} = R_x + N_\lambda - 1 \tag{4.28}$$

Therefore, we had 101 DMD SP columns. So the first 38 camera images corresponds to the 38 spectral channels in the first spatial SP column. Then images 2 to 39 correspond to the first 38 spectral channels in the second spatial SP column.

Another way to reduce the spectral calibration time is by realizing that we can turn on two DMD SP columns at a time. This is because we only have 38 spectral channels and 101 DMD SP columns. So in our calibration procedure we turn on two DMD SP columns at a time each 50 SP columns apart. Then for each spectrum in spectral library we only need 51 exposures. If we assume a 5 second exposure this only takes about 20 minutes. In practice we use a much longer exposure time to increase the SNR of each FPA exposure. The total time for a four class spectral library is approximately 4 hours.

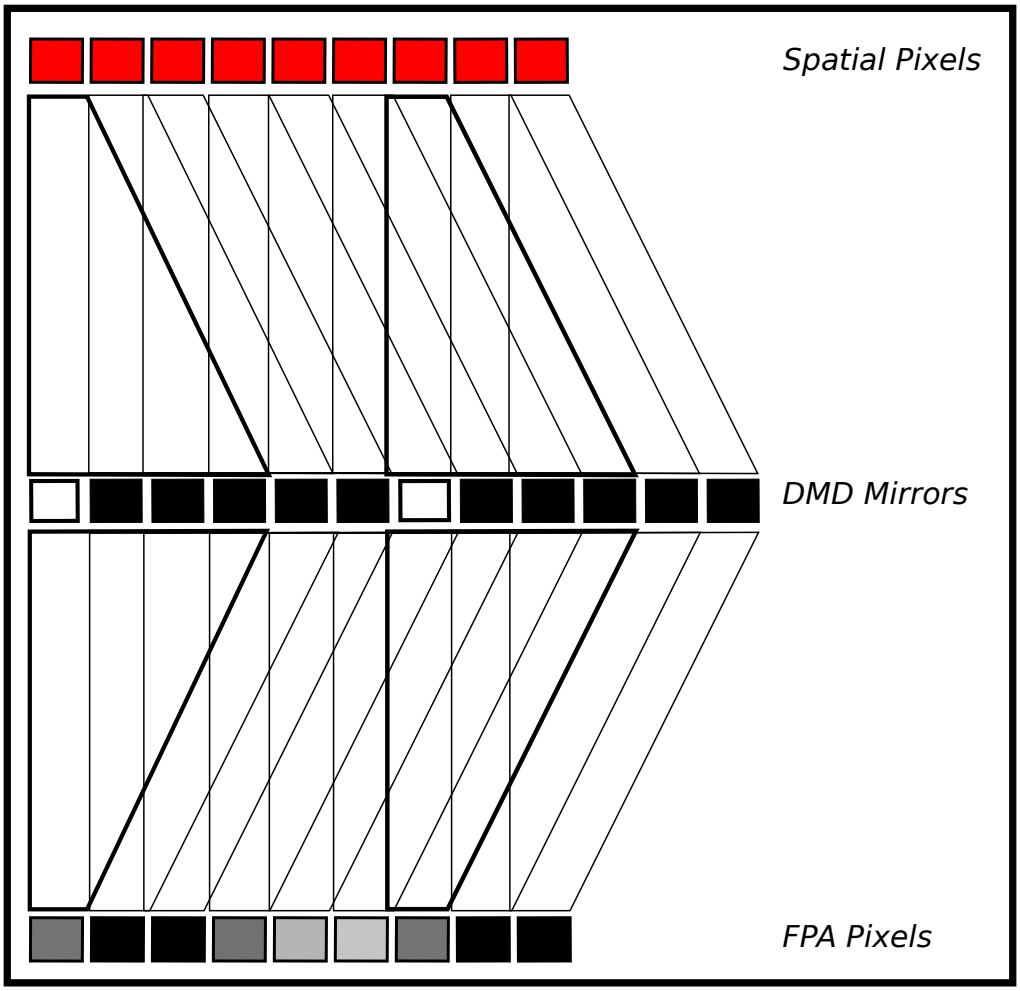

Figure 4.18: In order to speed up the spectral calibration we turn the entire monitor ROI to the same spectrum and sweep two DMD columns at a time.

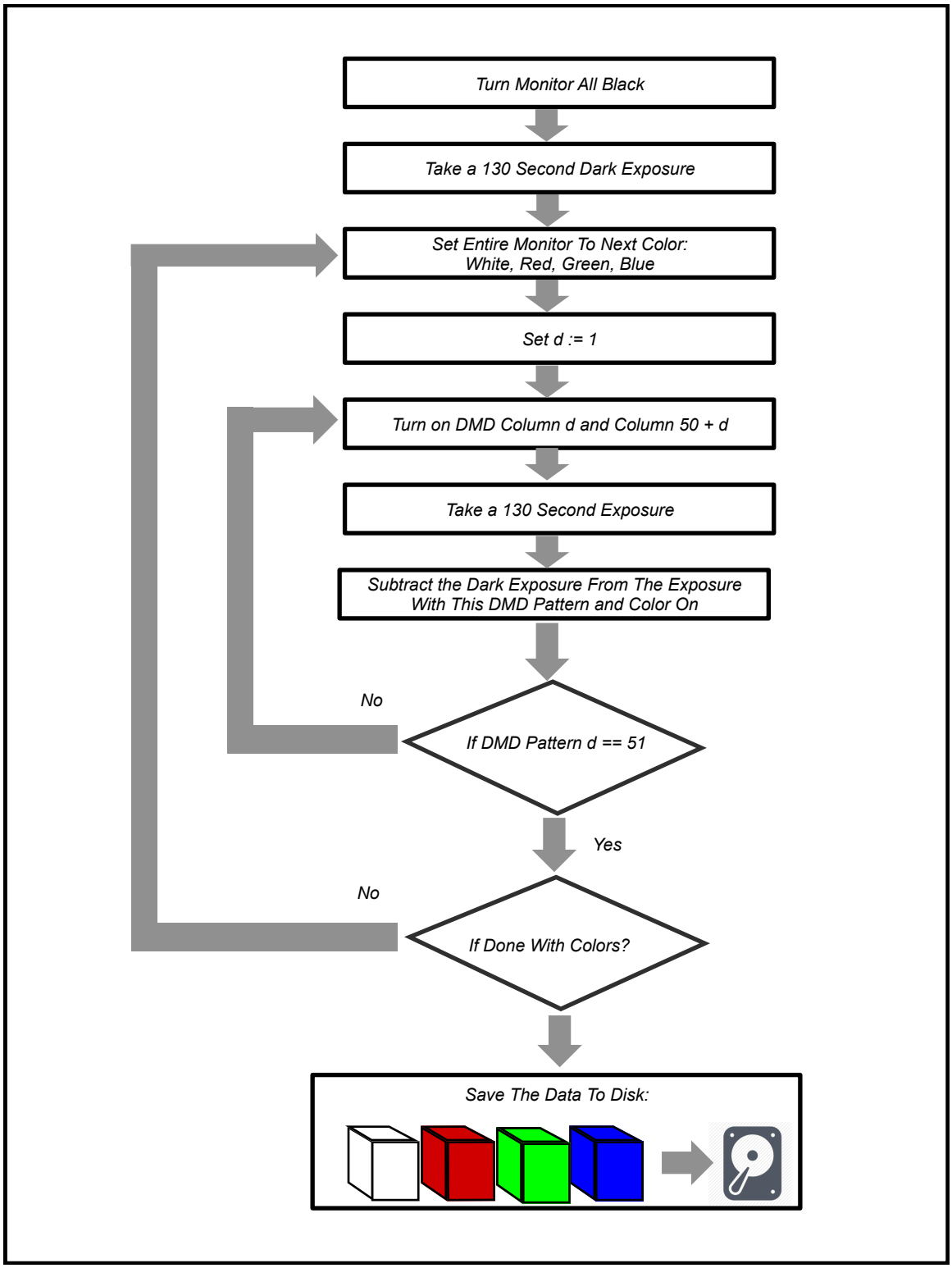

Figure 4.19: In order to speed up the spectral calibration we turn the entire monitor ROI to the same spectrum and sweep two DMD columns at a time.

4.4.3.3 Noise Model Calibration

As I discussed in Section 4.3.1, the adaptive algorithm we used to make classification decisions compares the ratio of posterior probabilities by updating ratios of the likelihoods after each measurement. The likelihood is defined as the probability of the measurement given that the hypothesis is true. This probability is given by the noise model using Equation (4.23). Accurate estimation of the noise standard deviation, $\hat{\sigma}$, is required for optimal classification rates.

If $\hat{\sigma}$ is much larger than the actual value, σ, then the probability for different spectral classes will tend to be similar. However, as I discussed, the pPCA algorithm depends on weighting the probability of some classes higher than others to outperform non-adaptive systems. This may lead to longer convergence times to the correct class. If the $\hat{\sigma}$ is too small then the algorithm may converge to the wrong class.

Fortunately estimating the noise of the system is a relatively straightforward. At each spatial SP location, we randomly choose one the spectra from the spectral library and display it, then the we display a random pattern on the entire DMD and record the camera image. We then compare the camera image with the measured intensity value at each spatial location to what we expected from the simulated camera image. This is repeated several times until we have sufficient statistics to plot the distribution of the differences between measured and expected intensities, see Figure 4.20. Using the MATLAB `fit` function with the one-dimensional Gaussian option `gauss1`, we fit a Gaussian to the data to estimate the standard deviation of the noise, $\hat{\sigma}$.

The value of σ is the intrinsic amount of noise in the instrument. It takes into account all possible sources of noise in the AFSSI-C. However, we would like to vary the total amount of noise to see how the experimental classification performs compared to simulation and how the it performs compared to various other imaging spectrometers. This is done in post-processing by adding normally distributed noise with standard deviation σ_{added} to the measurements. Since the intrinsic system noise is approximately Gaussian, the total variance is thus

$$\sigma_{total}^2 = \sigma^2 + \sigma_{added}^2 \tag{4.29}$$

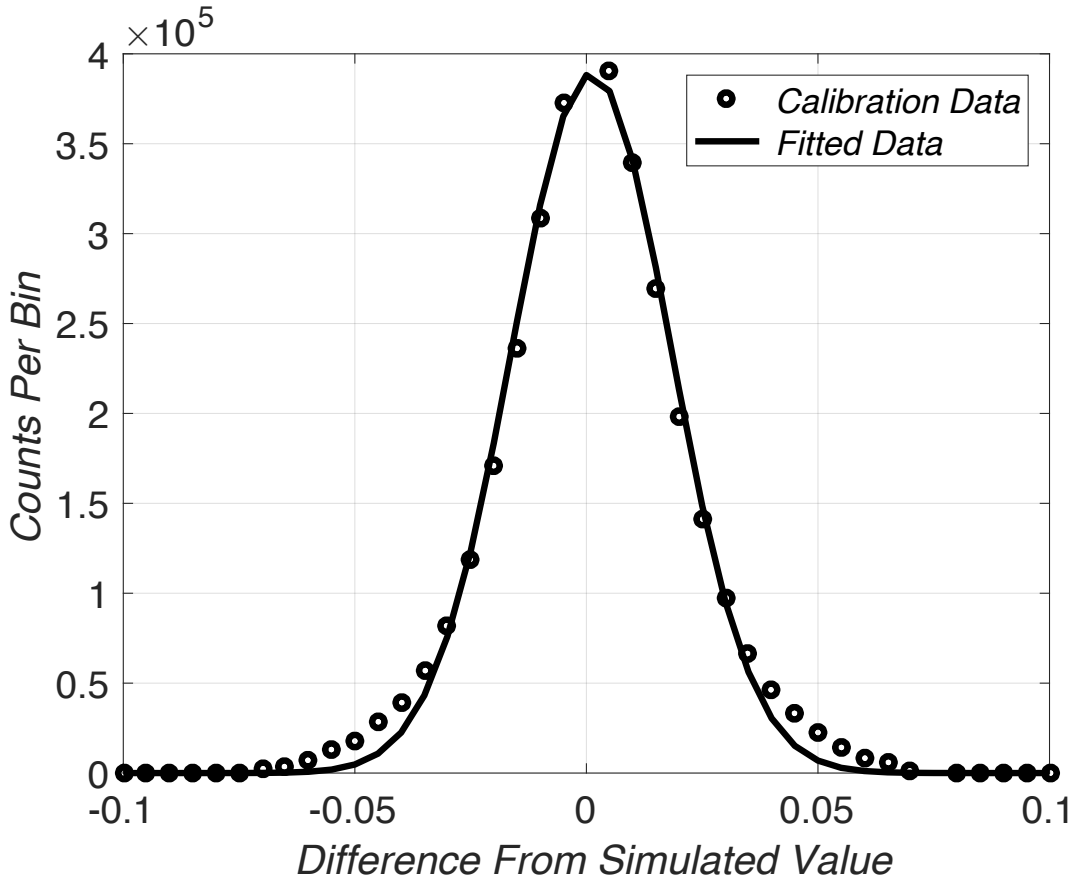

Figure 4.20: Estimating the System Noise: The circles represent the distribution of difference between the measurement and expected values of randomly chosen spectra with random spectral filters. The solid line represents a Gaussian fit, which allows us to estimate the standard deviation of the system noise.

4.5 Experimental Results

Figure 4.21 presents experimental classification results at 0, -3, and -6 dB TSNR after measurements. Measurements 1 through 30 are presented in Appendix F. As the classification difficulty increases (lower TSNR), the adaptive algorithm takes more measurement steps to correctly classify.

The experimental classification rates that we obtained agree well with simulation. Figure 4.22 shows the performance of the AFSSI-C experimental results and the simulated values. The simulated and experimental data are the average of 30 simulations and 10 experiments, for each of the four TSNR values representing several levels of increasing classification difficulty (0, -3, -6, and -9dB TSNR). The agreement between simulation and experimental results allows us to compare the

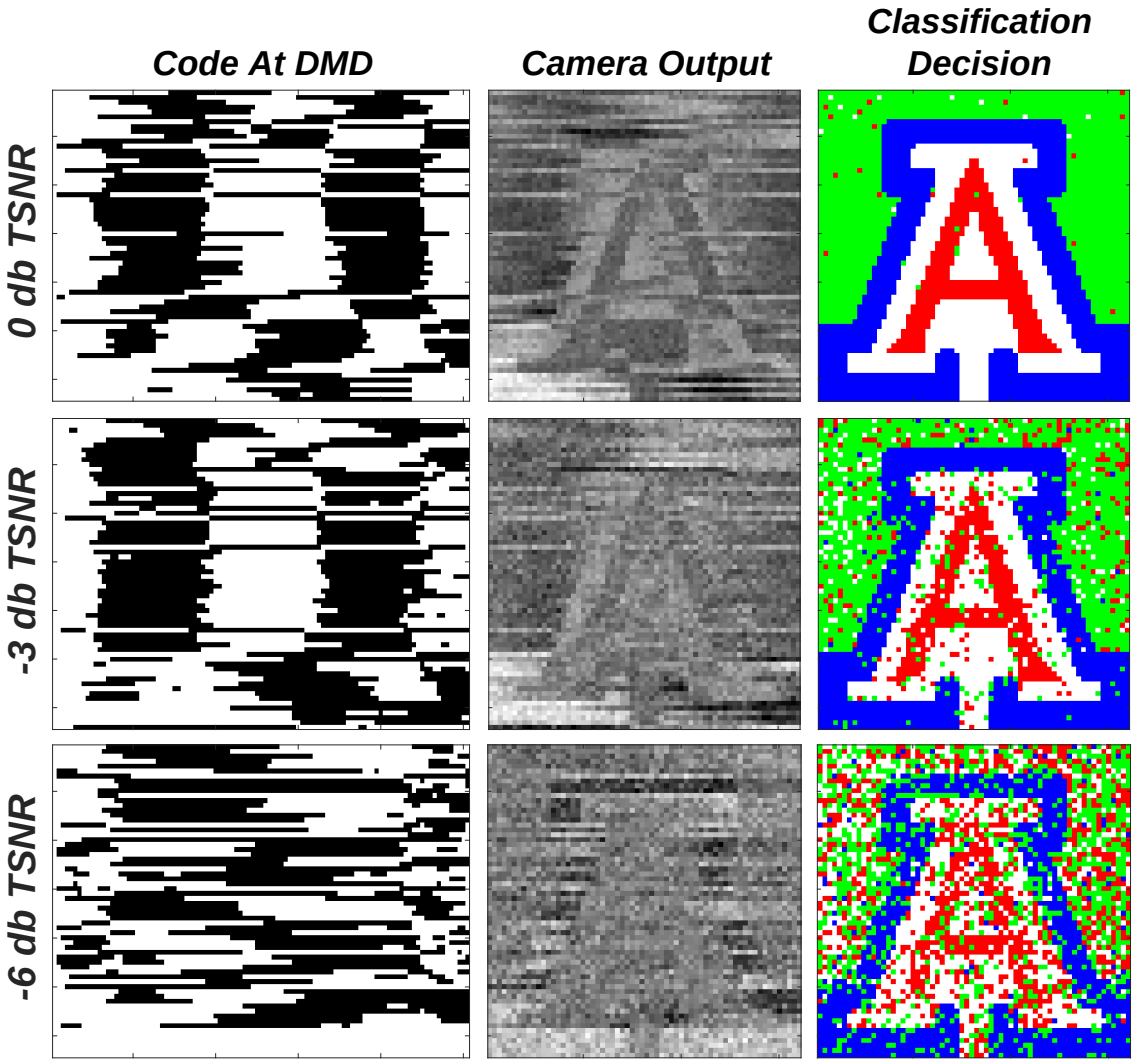

Figure 4.21: Data from after the third measurement step at 0, -3, and -6 dB TSNR. The left column is a depiction of the DMD code, center is the output from the detector, and the right is the classification decision at the current measurement.

performance of the AFSSI-C with traditional spectral imaging systems and non-adaptive computational sensors via simulation, see Figure 4.23. It also compares different feature design modalities. To test feature designs, the adaptive, joint-pPCA designed features are compared to static, pseudo-random features which is what is found many other computational sensing spectral imagers [32]. The traditional systems that were investigated as simulations are the pushbroom, whiskbroom, and tunable filter systems. The performance of the simulated traditional systems follows intuition: a whiskbroom system has to sweep over all 4096 spatial locations, while the pushbroom and tunable filter system only make 64 and 38 scanning steps,

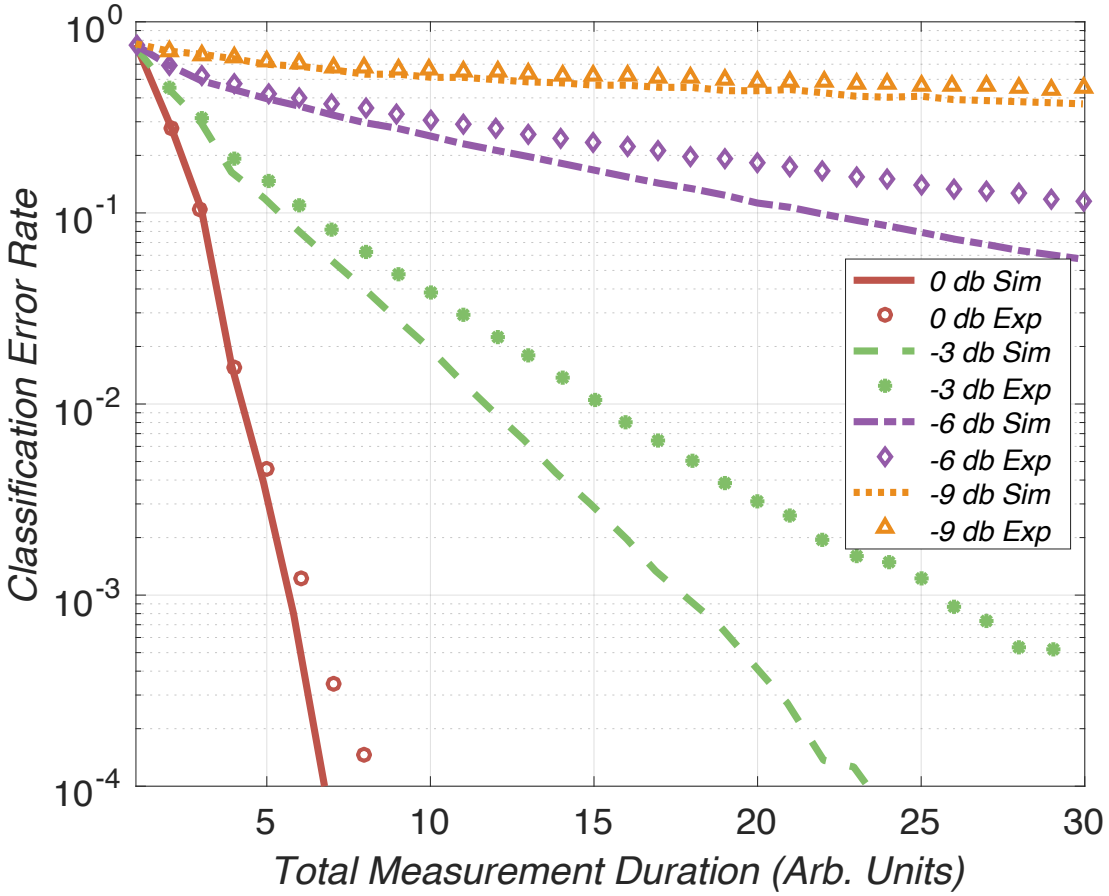

Figure 4.22: Comparison of the AFSSI-C experimental system results to the simulation results for multiple TSNR levels by plotting the classification error versus measurement. Shown are repeated experiments of a $64 \times 64 \times 38$ spectral datacube and a 4-class library.

respectively. This explains the greater TSNR and hence greater classification accuracy of the tunable filter system, with the whiskbroom system being least accurate of the three. Note that the sequential nature of the measurements in the AFSSI-C limits its applicability to those scenes which are slowly varying with respect to the time needed to achieve a desired classification accuracy.

There are a number of phenomena which give the AFSSI-C an advantage over traditional systems. When comparing the AFSSI-C system with joint-pPCA designed features to traditional systems at the 5^{th} measurement in Figure 4.23, we see that the classification accuracy improves by $250\times$. This improvement in performance is attributed to a combination of factors such as the open aperture architecture of the AFSSI-C design, lack of scanning, and adaptivity.

Figure 4.23 also demonstrates the advantage joint-pPCA. By inspecting the 5th

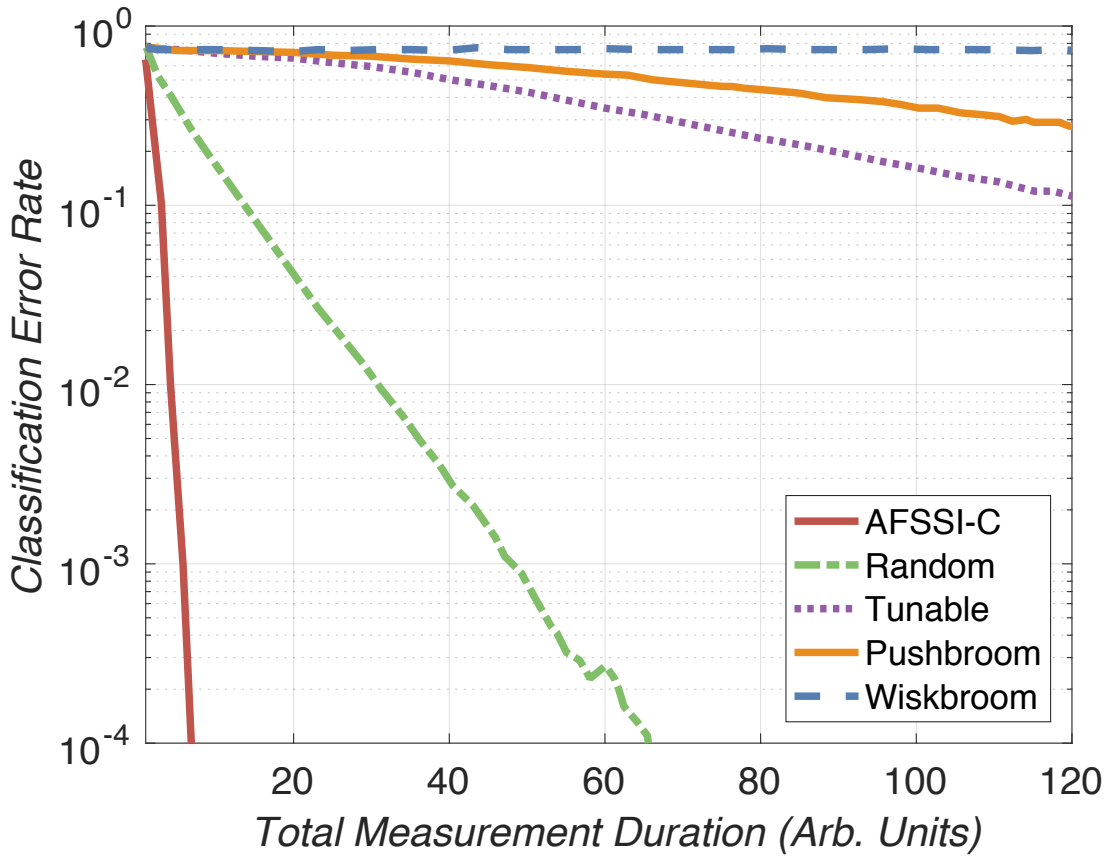

Figure 4.23: Simulation comparing the classification performance for different measurements at TSNR = 0 for different systems: the AFSSI-C with designed features (joint pPCA), the AFSSI-C with random features, the traditional pushbroom imager, the traditional tunable filter imager, and the traditional whiskbroom imager. The input is a $64 \times 64 \times 38$ spectral datacube with a 4-class library.

measurement, one can observe a $100\times$ performance gain relative to static, pseudo-random coding. The performance curve for the random coding case is generated by directly classifying from the acquired measurements using an identical Bayesian inference framework. The information processing inequality of information theory [4] guarantees us that the alternative approach of classifying after datacube reconstructions can do no better than direct classification. Thus, the difference of these two curves represents the actual classification performance improvement due to adaptivity, while the separation between the static-coded curve and the traditional systems represents the performance improvement arising from the increased open aperture. The simulated results show that the AFSSI-C system to outperform traditional systems where classification is the desired result of the analysis.

4.6 Conclusion

In this chapter, I discussed the Adaptive Feature Specific Spectral Imaging-Classifier (AFSSI-C), a spectral imager classifier that utilizes adaptive spectral features in a low size, weight and power-cost (SWAP-C) configuration. My collaborators and I were able to show multiple order-of-magnitude improvement in classification accuracy compared to traditional spectral imaging systems when the noise in the system is equal to the minimum separation between the library spectra, by employing a system simulation corroborated with experimental results. By taking advantage of its adaptiveness, the AFSSI-C performance with designed features also achieves multiple order-of-magnitude improvement over a random feature implementation. These adaptive features are designed via Bayesian inference and a novel joint-probabilistic PCA approach, which drives the measurement decision evolution to boost the discrimination ability between spectral candidates at every spatial location.

Direct classification is a useful modality for many of the applications of spectral imaging. By making measurements of an encoded datacube instead of explicit measurement of every element, huge performance gains are realized. The AFSSI-C system can greatly benefit a number of industries that rely on *in situ* material classification.

Chapter 5

Computational Spectral Unmixing

5.1 Introduction

In Chapter 4, I argued that spectral classification is the goal of spectral imaging in most cases. However, in certain situations the analyst is interested in quantifying the presence of several materials in a single spatial location from a *mixed spectrum*. Mixed spectra can occur when the sensor is part of a high-altitude platform such as an unmanned aerial vehicle (UAV) or satellite. The large standoff distances between the ground and the instrument result in large spatial resolutions. For example, in the Hyperion Imaging Spectrometer, which is satellite based, the spatial resolution is 30 meters [97]. One cannot reasonably expect a single material to always occupy that large of an area. *Spectral unmixing* is any procedure which attempts to take the measured spectrum of a mixed pixel and decompose it into a set of constituent spectra called the *endmembers* and a set of corresponding fractions called the *fractional abundances*.

Traditional spectral unmixing requires several seperate steps to quantify the fractional abunances. The isomorphic sensor must spatially or spectrally scan the object scene, building the spectral datacube piece by piece. At each step, the restricted aperture or wavelength range rejects a significant fraction of the availiable light. A post-processing step is then used to reduce the size of the data to reduce the computational load. Finally an inversion step is used to compute the fractional abundances. Intuitively, the advantages of computational sensing discussed throughout this dissertation should be able to ameliorate some of the design trade-offs in

traditional spectral unmixing.

In this chapter, I will talk about my efforts to apply the techniques of computational sensing to directly estimate the fractional abundance without the need to reconstruct the spectral datacube. I will introduce a computational spectral imaging architecture called the LCOS Computational Spectral Imager (LCSI). I will then provide simulation results that demonstate the advantage of computational spectral unmixing using the AFSS and the LCSI over traditional architectures by leveraging the Fellgett and Jacquinot advantage with compressive sensing algorithms which promote sparse solutions. Results from a proof-of-principle experiment demonstrate the promise of computational spectral unmixing.

5.2 The Linear Mixing Model

There are two main reasons why mixed spectra occur [98, 99]. First, if the spatial resolution of the sensor is low enough, separate materials can jointly occupy the instantaneous field-of-view (FOV) of the pixel, the resulting spectral measurement is a combination of the constituent spectra, see Figure 5.1(a). In this case, one can imagine the object scene as a *checkerboard* mixture: light from the illumination source scatters or reflects from only one of the materials before being observed by the sensor, multiple scattering between materials are ignored. The second reason for mixed pixels occurs when different materials are combined into a homogeneous mixture, see Figure 5.1(b). In this case, mixed spectra are not caused by poor spatial resolution, they are inherent to the nature of the scene. For the purposes of this chapter, I will focus on the first case, mixed spectra caused by the spatial resolution limitations of the sensor.

When mixed spectra occur due to the spatial resolution limitation of the instrument, the fractional abundance is linearly proportional to the relative area of each material. This is called the Linear Mixing Model (LMM), where the mixed spectrum can be written as

$$\mathbf{f} = \sum_{r=1}^{N_R} x_r \mathbf{s}_r + \mathbf{e} = \mathbf{S}\mathbf{x} + \mathbf{e} \tag{5.1}$$

where $\mathbf{f}$ is the mixed spectrum, $\mathbf{s}_r$ is the r^{th} endmember spectrum, $\mathbf{S}$ is a matrix in which the columns are the endmember spectra, $\mathbf{x}$ is the fractional abundance vector,

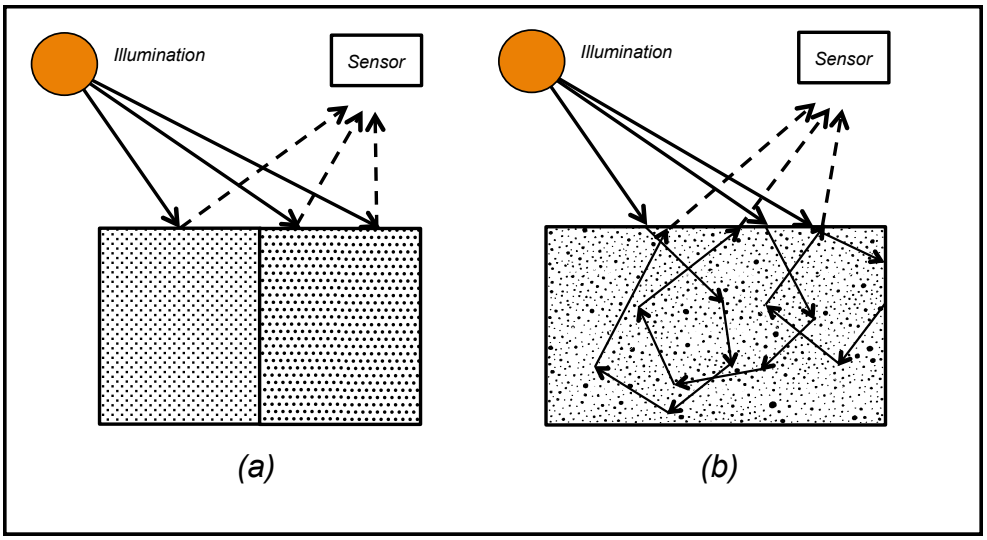

Figure 5.1: (a) Illustration of linear mixing where incident solar radiation reflects from a surface and the surface consists of distinct materials. (b) Illustration of nonlinear mixing where incident solar radiation encounters an intimate mixture of materials, reflecting or scattering multiple times before being reflected toward the spectral image sensor.

and N_R is the number of endmembers in the endmember library, each spectra has N_λ spectral channels. In the Linear Mixing Model (LMM), the interactions between distinct endmembers are assumed to be neglible [100].

There are two contraints imposed by the physics of the situation. Intuitively we should expect that the fractional abundance should be equal to or larger than zero. This is the *nonnegativity* constraint:

$$x_r \geq 0. \tag{5.2}$$

We should also expect that if energy is conserved, i.e. there is no absorption of light, then the fractional abundance should sum to one. This is the *additivty* constraint:

$$\sum_{r=1}^{N_R} x_r = 1 \tag{5.3}$$

5.2.1 Unmixing in Traditional Spectral Imaging

In traditional spectral imaging, the spectral datacube is first isomorphically acquired by the instrument before any unmixing step is performed. Often a *dimensionality reduction* step is used to reduce the computational burden of processing the spectral datacube [98, 99]. If the endmembers are unknown, an *endmember determination*

step is executed. Finally, the *inversion* step is used to estimate the fractional abundances.

Notable data reduction algorithms include Principal Component Analysis (PCA) and Maximum Noise Fraction (MNF). As described in Chapter 2, PCA is applied to the measured data and finds the basis which decorrelates the data. In PCA one typically observes a steadily decreasing signal-to-noise ratio as the principal compenent number increases [101]. However, this is not always the case, since it equates variance with information and is based on the assumption that the data structure can be described by a multi-dimensional normal distribution [102]. In Maximum Noise Fraction (MNF), the algorithm attempts to order the components in terms of SNR which consists of two seperate PCA rotations and a noise whitening step. MNF requires estimation of the noise covariance matrix in addition to the covariance of the data.

The inversion step actually estimates the fractional abundance. There are a variety of inversion techniques which actually attempt to estimate the fractional abundance vector. Many are based on minimizing the squared error and attempt to enforce additivity or non-negativity [99, 103]. There are various algorithms based on Maximum A Posteriori (MAP), Maximum Likelihood Estimation (MLE), and clustering which can be used for spectral unmxing. Unfortunately, we cannot explore each inversion technique, however due to their prevalence, I will use least-squares based inversion technique when comparing computational spectral unmixing techniques to tradiational unmixing techniques.

5.3 Architecture

In this research, two seperate architectures are used for spectral unmixing. The first architecture is the AFSS, which is the single pixel version of the AFSSI-C described in depth in Chapter 4. The second architecture is a Liquid Crystal on Silicon (LCOS) based spectral imager, called the LCSI, which allows for an extremely compact instrument, see Figure 5.2. The system provides a programmable spectral filter, which can be independently addressed at each physical pixel of the SLM. The device consists of an array of micro cells of liquid crystal on a reflecting layer [104]. Each layer of liquid crystal can be modeled as a thin retarder plate. Since

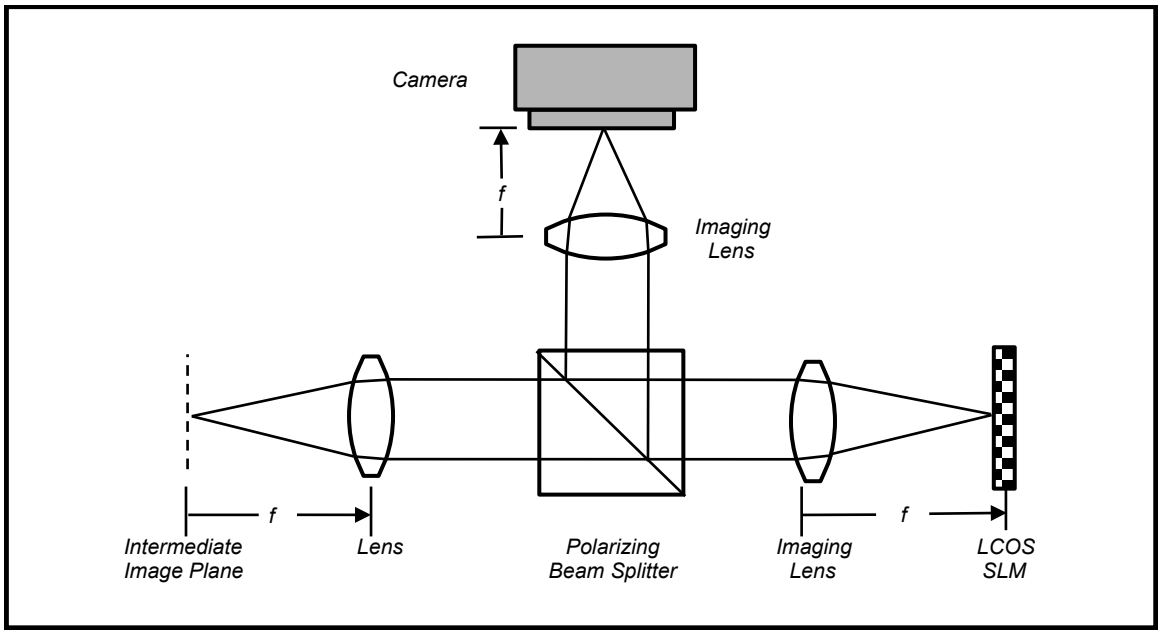

Figure 5.2: The LCOS Spectral Imager. Light from the intermediate image plane is collimated. A polarizing beam splitter passes p-polarized light and rejects s-polarized light. Upon reflection of the LCOS SLM, the polarization state is changed to some elliptical polarization state. Only the s-polarized portion of the elliptical polarization is reflected toward the upper part where it is imaged onto a scientific camera. The intensity of the light that is passed depends on birefringence created by the programmable LCOS.

the birefringent phase retarder is sensitive to wavelength, the LCOS combined with a polarizing beam splitter or a linear polarizer produces a wavelength dependent transmission pattern, a spectral filter, which modulates the input spectra [56].

Unfortunately, a full discussion of polarization is not appropriate for this dissertation. The important point is that for light of a single wavelength, the LCOS produces polarization which in general is not linearly polarized. Placing a linear polarizer after the light has been reflected from the LCOS will then force the transmitted light to be linearly polarized but at an intensity that depends on the projection of the output polarization from the LCOS onto the transmission direction of the polarizer. Passing non-monochromatic light to an LCOS and then a linear polarizer will impart a wavelength dependent intensity. In short, the SLM provides polarization and wavelength dependent transmission patterns to encoded the spectral datacube [105].

5.3.1 Forward Model

The forward model for the LCSI is similar to the forward model for the AFSSI-C presented in Section 4.2.1, except we do not need to account for the dispersion when imaging from the input plane to the LCOS and from the LCOS to the FPA of the camera. I will thus skip the derivation of the forward model and simply present the final equation for the measurement value at pixel n and l from the camera:

$$\Gamma_{nl} = \sum_{n'l'} \iiint \text{rect}\left(\frac{x}{\Delta} - l, \frac{y}{\Delta} - n\right) \text{rect}\left(\frac{x}{\Delta} - l', \frac{y}{\Delta} - n'\right) \times T_{n'l'}(\lambda) D_0(x, y; \lambda)\, dx\, dy\, d\lambda. \tag{5.4}$$

Notice that there is no dispersion constraint like the one created in the AFSSI-C or a joint spatial-spectral constraints like in the CASSI. The LCOS creates dispersion by the very nature of wavelength dependent birefringence. If one so choose to, one can simply treat each pixel in the image as completely independent from neighboring pixels. This greatly simplifies the analysis.

Similar to the AFSSI-C we can further simplify this by imagining a discrete spectral density, the spectral datacube. The discretized source spectral datacube is D, and then the detector signal Γ is a result of spectral filter created by the Polarizing Beam Splitter (PBS) and LCOS combination where spectral filter T acting on the pixelated source is

$$\Gamma_{n,l} = \sum_{c=0}^{N_\lambda - 1} T_{n,l,c} D_{n,l,c} \tag{5.5}$$

which shows the measurement at each pixel being the inner product of the source spectrum and the spectral filter created by the PBS-LCOS combination. In a single spatial location, this reduces to a simple inner product

$$g_m = \mathbf{h}_m^T \mathbf{f} \tag{5.6}$$

where the subscript m represents the m^{th} measurement step and $\mathbf{f}$ the true spectrum at that pixel. For a sequence of measurements this simplies to

$$\mathbf{g} = \mathbf{H}\mathbf{f} \tag{5.7}$$

where the m^{th} row of $\mathbf{H}$ is $\mathbf{h}_m^T$ and $\mathbf{g}$ is an $N_m \times 1$ vector, $\mathbf{f}$ is the ground truth mixed spectrum

$$\mathbf{f} = \mathbf{S}\mathbf{x} \tag{5.8}$$

Thus for a sequence of noisy measurements at a single pixel

$$\mathbf{g} = \mathbf{HSx} + \mathbf{e} = \mathbf{Ax} + \mathbf{e} \tag{5.9}$$

where $\mathbf{H}$ is an $N_m \times N_\lambda$ matrix, $\mathbf{S}$ is the endmember library which is an $N_\lambda \times N_R$ matrix, $\mathbf{x}$ is the fractional abundance vector which an $N_R \times 1$ vector, $\mathbf{e}$ is the additive noise which is a $N_m \times 1$ vector. If I define

$$\mathbf{A} = \mathbf{HS}, \tag{5.10}$$

one can think of $\mathbf{H}$ as the sensing matrix and $\mathbf{S}$ as the representation matrix as discussed in Section 2.5.

5.4 Solving the Inverse Problem

For this work I chose to use the least-squares estimator (LSE)

$$\hat{\mathbf{x}} = \left(\mathbf{A}^T\mathbf{A}\right)^{-1}\mathbf{A}^T\mathbf{g} \tag{5.11}$$

to demonstrate the advantage provided by multiplexing without using compressive sensing based algorithms. To demonstrate compressive sensing approaches, I will used the built-in MATLAB `lasso` function which attempts to minimize the ℓ_1 regularized least-squares objective function

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}} \ \|\mathbf{Ax} - \mathbf{g}\|_2^2 + \tau\|\mathbf{x}\|_1 \tag{5.12}$$

in order to find fractional abundances that are sparse.

In the traditional spectral imager, the fractional abundance is estimated after the spectral datacube is acquired. In our work, the fractional abundance can be estimated after each measurement step. However, since we need a way to compare our results to traditional spectral imaging, we can use the tunable filter architectures for a baseline comparision. The tunable filter acquires a single wavelength over the entire field-of-view and thus allows us to obtain measurements in a time sequential manner. This means that we can use the LSE to get an idea of how a traditional spectral imaging architecture would perform.

As I mentioned earlier, in remote sensing the fractional abundance vector $\mathbf{x}$ tends to be sparse: the number of endmembers in the library is much larger than

number of endmembers that are actually present in the mixed spectrum $N_R > \|\mathbf{x}\|_0$. Therefore, one can invoke the techniques designed for compressive sensing to find solutions that are sparse.

5.5 Prior work

5.5.1 Prior Efforts in Computational Spectral Unmixing

Several researchers have shown promising results in applying compressive sensing to spectral unmixing using a modified single-pixel camera architecture [106]. They demonstrated the ability to reconstruct the fractional abundance planes without the need to explicitly reconstruct the spectral datacube. In this approach, the object scene is imaged onto a DMD and then a condensor lens focuses the reflected light into a whiskbroom spectrometer. One can think of this architecture as a set of parallel single-pixel cameras each operating at a different spectral channel, with the constraint that each DMD must display the same pattern. This architecture does not code the spectral dimension of the spectral datacube. The researchers demonstrated compressive unmixing by minimizing the total variation (TV) of the endmember images while enforcing the nonnegativity constraint.

In another effort, researchers use the Coded Aperture Snapshot Spectral Imaging (CASSI) architecture to perform compressive sensing on the spectral datacube and solve the ℓ_1-regularized least squares problem (lasso in regression) to promote sparsity in the fractional abundances [107]. Due to the nature of the single-disperser CASSI architecture, the researchers are forced to solve a larger joint-inference problem to preform spectral unmixing.

5.5.2 Prior efforts using LCOS Computational Spectral Imaging

Several groups have previously demonstrated computational spectroscopy and spectral imaging results using variations of liquid crystal technology. The first instance, in 2012, used a single-pixel liquid crystal device to demonstrate compressive spectroscopy and exhibited a 10× reduction in the number of measurements compared

to a traditional ismorphic spectrometer [108]. Shortly after in 2013, a demonstration of a compressive spectral imager using an LCOS SLM was published which jointly coded spatial and spectral features [109]. In 2015, an LCOS based hyperspectral imaging sensor demonstrated blind compressive sensing, using a Bayesian approach to dictionary learning *in situ* [56]. That same year, a miniture ultraspectral imaging system based on a custom built liquid crystal cell, which applies the same spectral filter to each spatial location, demonstrated the ability to reconstruct gigapixel spectral datacubes with an order of magnitude reduction in measurement steps compared to isomorphic systems [110]. However, to my knowledge, no one has ever attempted to perform direct spectral unmixing using an LCOS based device.

5.6 Design and Selection of Spectral Filters for Unmixing

5.6.1 Adaptive Unmixing Algorithm For the AFSSI-C

The Adaptive Feature Specific Spectrometer (AFSS) has the ability to display psuedo-arbitrary spectral filters with the restriction of using binary codes $\{-1, +1\}$ using the DMD. In the AFSS, one can emulate a tunable filter spectrometer by measuring one spectral channel at time, i.e. turn on one mirror per measurement step. In this case, the measurement matrix is equal to the identity matrix $\mathbf{H} = \mathbf{I}$. Combining the tunable filter approach with the LSE produces what one should expect from a traditional isomorphic spectrometer to conduct spectral unmixing.

We can also use random binary codes to achieve a multiplexed measurement to improve the throughput of each measurement to reduce the unmixing error. The spectrum can then be estimated by processing the measurement with the LSE to demonstrate the performance gained by collecting more light per measurement step. However, estimating the fractional abundance using a compressive sensing based algorithm such as the MATLAB `lasso` function can produce even better unmixing results by enforcing sparsity.

As shown in the AFSSI-C, adaptive coding can significantly improve performance compared to multiplexing alone. We developed an algorithm for adaptively creating spectral filters which uses a modified version of pPCA to create adaptive spectral

filters. This algorithm begins with the spectral library which consist of the each endmember spectra $\mathbf{S}$. Initially, before any measurements are made, the estimated fractional abundance of each endmember is assumed to be the same:

$$\hat{\mathbf{a}}_{m=0} = \begin{bmatrix} \frac{1}{N_R} \\ \frac{1}{N_R} \\ \vdots \\ \frac{1}{N_R} \end{bmatrix} \tag{5.13}$$

where N_R is the number of endmembers. The subscript denotes the m^{th} measurement step. Before any measurement is made, $m = 0$. Then each endmember in the spectral library is weighted by the square of their respective estimated fractional abundance

$$\mathbf{S}_w = \begin{bmatrix} \hat{a}_1^2\mathbf{s}_1 & \hat{a}_2^2\mathbf{s}_2 & \dots & \hat{a}_{N_R}^2\mathbf{s}_{N_R} \end{bmatrix} \tag{5.14}$$

this is called the weighted spectral library. Then the eigenvectors of the unnormalized covariance matrix of the weighted spectral library are computed to obtain the principal components

$$\mathbf{X}_m = \mathbf{S}_w\mathbf{S}_w^T. \tag{5.15}$$

Where the first principal component corresponds to the direction of largest variance and so on. Initially, for the first measurement, the algorithm chooses the first principal component for the spectral filter $\mathbf{h}_{m=1} = \mathbf{p}_1$. The measurement is then recorded

$$g_{m=1} = \mathbf{h}_{m=1}^T\mathbf{f} + e_m \tag{5.16}$$

A simulated "guess" measurement based on the current estimated fractional abundance $\hat{\mathbf{a}}_{m-1}$ is also computed:

$$\gamma_{m=1} = \mathbf{h}_{m=1}^T\mathbf{S}\hat{\mathbf{a}}_{m=0} \tag{5.17}$$

notice that the guess measurement does not include any simulated noise. Remember $\mathbf{S}$ denotes the original spectral library matrix, not the weighted spectral library matrix. After the measurement is recorded, the ℓ_2 norm of the difference between of the guess measurement and the actual measurement is recorded:

$$\xi_m = \|\gamma_m - g_m\|_2 \tag{5.18}$$

The fractional abundance is then estimated using the MATLAB `lasso` function. In practice, the LSE is used for the first measurement step since the function `lasso` requires atleast two rows for $\mathbf{A}$ and therefore two measurements, to run.

The spectral library is then reweighted using Equation (5.14). Again the principal components are recomputed. For the m^{th} measurement step, the m^{th} principal component is used for the spectral filter. After the measurement has been recorded, the ℓ_2 norm of the difference between the guess measurement and the actual measurement is recomputed ξ_m.

This continues after each measurement until either two things happen:

1. If the current ℓ_2 norm of the difference between the guess and the actual measurement exceeds the last ℓ_2 norm of the difference between guess and actual measurement by
$$\xi_m > \xi_{m-1} - \frac{\sigma}{2} \tag{5.19}$$
where σ is the standard deviation of the system noise, which is assumed to be AWGN.

2. Used the sixth principal component.

when either of these two conditions are met, the loop resets to using the first principal component again. I constrained the algorithm to only use the first six principal components, because I noticed that the unmixing preformance is optimized when limited to only the first six principal components. Intuitively, this may be occuring because higher principal components tend exhibit lower SNR or because the spectra in the library do not exhibit significant high frequency features. This algorithm is called Switch Weighted Principal Component Analysis (SWPCA). It is important to note that as of yet, this algorithm does not consider the dispersion constraint of the AFSSI-C, and therefore is more appropriate for the single pixel version, the AFSS. The MATLAB code which simulates Switch Weighted Principal Component Analysis (SWPCA) is found in Appendix G.

5.6.2 Hybrid Spectral Filters for the LCSI

The spectral filters produced by the LCSI architecture are constrained by the physics of the birefringence dispersion created by the LCOS. For our particular model, the

Holoeye PLUTO LCOS SLM, the spectral filter is changed by sending a different grayscale value to the green channel of the video input (Holoeye allows their SLM to be connected like a second monitor). Since there are 256 grayscale values (0-255), there are 256 different spectral filters, see Figure 5.3.

Since one is not able to design the spectral filters individually, the next best thing is to choose which spectral filters should be used at each measurement step. Intuitively, one can imagine that using the same spectral filter over and over again will lead to a measurement matrix $\mathbf{H}$ where the rows are linearly dependent. Instead, one should strive to construct a measurement that is highly incoherent to satisfy the restricted isometry property (RIP).

For my experiment, I will compare several methods of choosing the spectral filters. The first method is simply selecting them at random from a discrete uniform distribution between 1 and 256, while constraining the selections to prevent using the same spectral filter from being used more than once. This technique however does not attempt to incorporate any prior knowledge about the statistics of the endmembers, besides sparsity.

One attempt to incorporate the statistics of the endmembers is to use PCA to select the spectral filters. One could imagine an algorithm which computes the principal components of the endmembers (spectral library) and then selects the spectral filters which most closely resemble the principal component vector $\mathbf{p}$ based on their angle:

$$\mathbf{h} = \arg\max_{\mathbf{h}} \left\{ \frac{\mathbf{p}}{\|\mathbf{p}\|} \cdot \frac{\mathbf{h}}{\|\mathbf{h}\|} \right\}. \tag{5.20}$$

For the m^{th} measurement, the spectral filter whose dot product is largest with the m^{th} principal component of the endmember matrix is selected. However, using all the principal components actually increases the unmixing error after a certain amount of measurement steps, as compared to purely random selections. This occurs for several reasons: As the PC number increases, they actually become less informative. Second, principal components tend become more rapidly varying as the PC number increases, the spectral filters generated by the LCOS are in general smoothly varying and become more difficult to match with the higher principal components.

Thus, we created a hybrid spectral filter selection technique that uses PCA to select the first several spectral filters and then uses psuedo-random selections af-

ter a certain number of measurement steps. This balances the ability of PCA to choose spectral filters that improve the unmixing error at the initial measurement steps while using random selections to ensure the measurement matrix is incoherent enough to satisfy the RIP.

The hybrid approach to selecting the spectral filters is further extended to use a more advanced version of PCA called Maximum Noise Fraction (MNF) transform. MNF was originally developed to remove noise from multispectral satellite images [101] and attempts to select a basis which orders the signal-to-noise ratio of the MNF components. However, MNF is also used for Blind Signal Seperation (BSS), which is the seperation of a set of source signals from a set of mixed signals [111]. Instead of computing the principal components, I compute the maximum noise fraction components of the spectral library. I specifically used the noise adjusted principal component analysis (NAPCA) algorithm to compute the MNF of the spectra library which is found in [111]. The algorithm associates rapidly varying parts of the spectral library with noise and assumes the signal is contained in the smoothly varying parts of the spectral library.

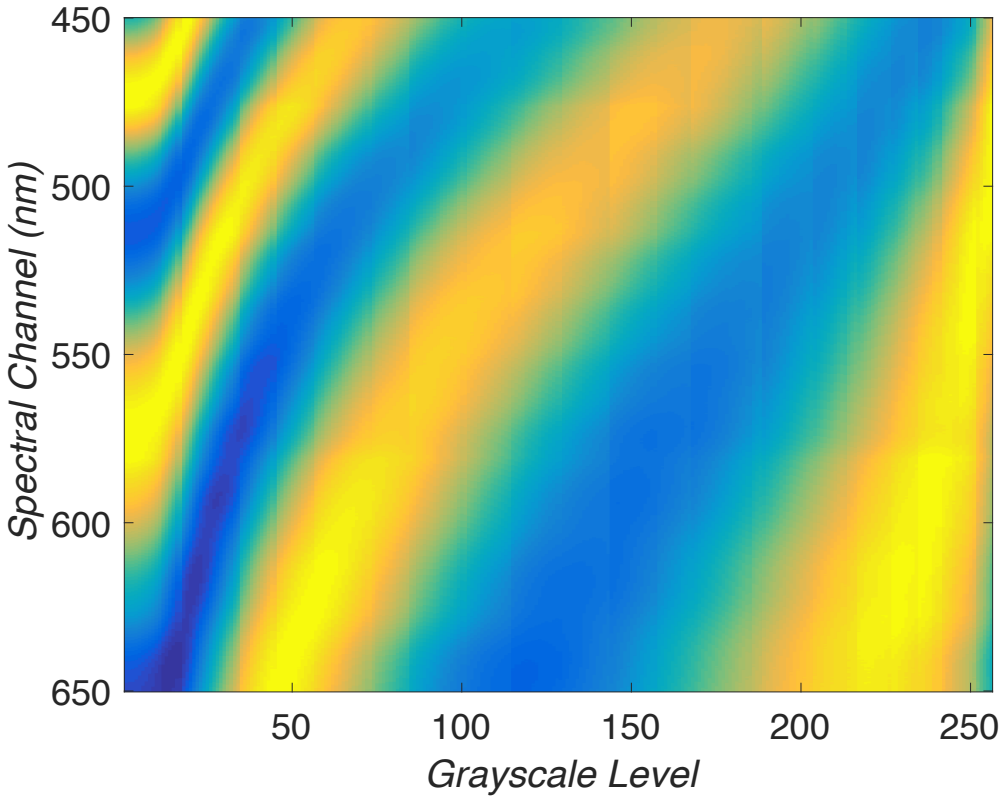

Figure 5.3: The spectral filters created by the Holoeye PLUTO SLM and polarizing beam splitter. Each column is a spectral filter at a particular grayscale level (0-255) sent to the PLUTO SLM by a custom MATLAB program.

5.7 Results

5.7.1 Simulation Results For the AFSS

Now that I have discussed various coding strategies for spectral unmixing using the AFSS, it is important quantify them. I performed simulations over five SNR levels from 10^{-2} to 10^{2}, where the SNR is defined as

$$\text{SNR} = \frac{\text{E}\left[\text{Var}_{N_\lambda(\mathbf{s})}\right]}{\sigma^2}, \tag{5.21}$$

which is the ratio of the average variance of the endmembers over the spectral channels to the noise variance. The results after 40 measurement steps are shown in Figure 5.4, which show the average Root Mean Squared Error (RMSE) of the estimated fractional abundance versus SNR. Root Mean Squared Error (RMSE) is

defined as

$$\text{RMSE} = \left[\frac{1}{N_R} \sum_{r=1}^{N_R} (\hat{a}_r - a_r)^2 \right]^{\frac{1}{2}} \tag{5.22}$$

The purple dotted line represents the performance of a tunable filter architecture combined with the least-squares estimator (LSE). With random binary patterns, one can obtain an improvement of approximately 3× which demonstrates the advantage of multiplexing, as shown in the mustard dash-dotted line. Further improvement is also achieved by incorporating the prior knowledge of sparsity of the fractional abundance, as shown in the orange dashed line. Finally, the blue solid line represents how adaptively designing the spectral filters and using compressive sensing provides an even better result than random coding alone.

Figure 5.5 shows the average RMSE as a function of measurement number for SNR = 0.01, 1, and 100. The blue line represents pseudo-random spectral filters, the error decreases monotonically with measurement number as one would expect. The green lines represent what happens if the adaptive algorithm never went back to the first principal component once either of the two conditions in Section 5.6.1 are met. Simply using higher principal components does not reduce the RMSE after a certain point, even when the spectral library is weighted by the fractional abundance. The red line represents the adaptive algorithm as discussed. This demonstrates the importance of going back to the first principal component whenever the two conditions are met in the adaptive algorithm: it breaks the performance plateau and significantly reduces the unmixing error compared to random coding alone.

5.7.2 Initial Experimental Results of Compressive Unmixing Using the AFSSI-C

I performed an initial proof-of-principle experiment using the AFSSI-C architecture. Due to the dispersion constraint of the AFSSI-C, I was unable to test the adaptive algorithm, however the random coding and the use of sparsity promoting algorithms can still be demonstrated.

For this experiment I had to upgrade the AFSSI-C prototype by adding a second display, see Figure 5.7. This is because the original prototype used only one LED computer monitor, so it can only display combinations of the same red, green, and

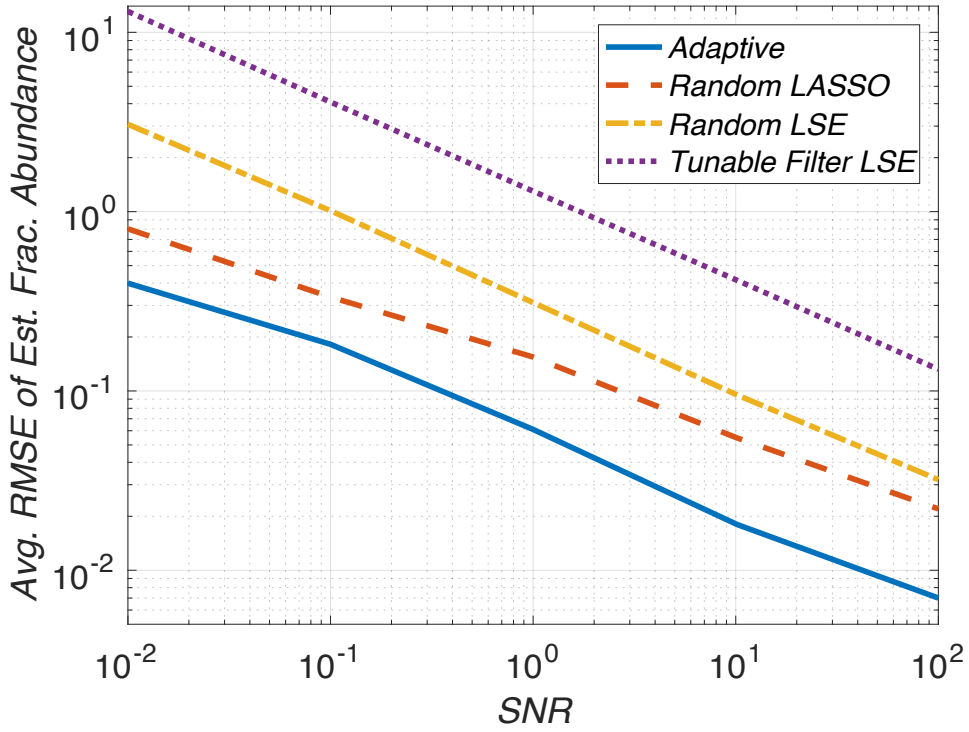

Figure 5.4: A comparison of spectral unmixing techniques for the AFSSI-C at five different SNR levels. The y-axis represents the average RMSE of the estimate fractional abundance. The x-axis represents the signal-to-noise in the measurements. The tunable filter LSE refers to the average unmixing performance if one were to use a tunable filter spectral imaging architectures with least squares estimation to estimate the fractional abundance. The random LSE line refers to the improvement in spectral unmixing This shows the improvement that multiplexing provides over simple tunable filter spectral imaging using the least squares estimator (LSE). Then using the multiplex advantage and algorithms that promote sparsity when the fractional abundance is known to be sparse, one can significantly outperform

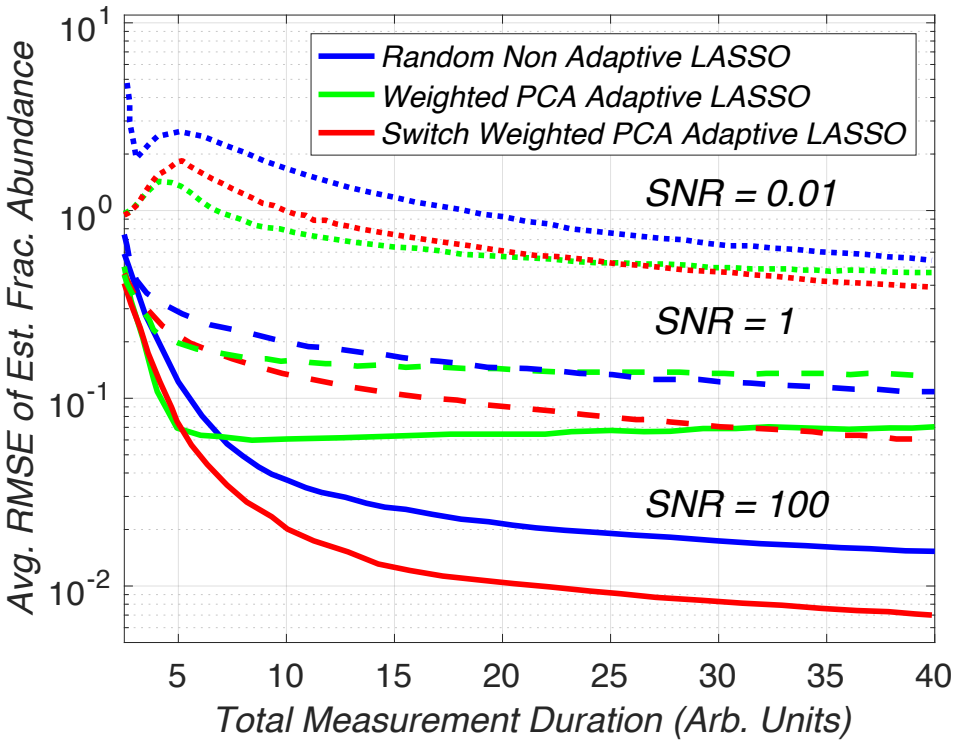

Figure 5.5: Comparison of Spectral Unmixing Techniques for the AFSSI-C versus Measurement Duration for SNR = 0.01, 1.0, and 100. Even though the spectral library is weighted by the fractional abundances from the last measurement, simply using every principal component underperforms using random spectral filters with ℓ_1 regularized least squares (lasso). Switching the principal components according the algorithm described in Section 5.6.1 significantly reduces the unmixing error.

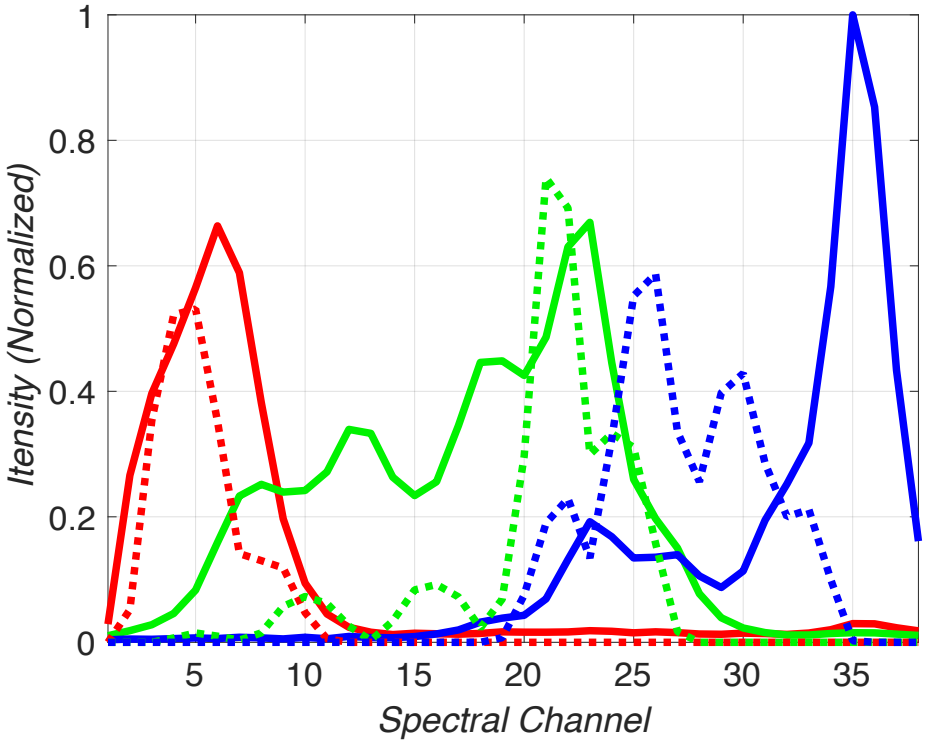

Figure 5.6: The six endmember spectra used for the spectral unmixing experiment. The library consist of the red, green, and blue spectra from an LED Monitor and the red, green, and blue spectra from the OLED. The LED monitor spectra may look different from the one shown in Chapter 4 since the spectra have been modified by transmitting through a pellicle beam splitter.

blue (RGB) spectra, producing only three endmembers. In order to have six endmembers, I added an OLED display which also has red, green, and blue spectra but are different enough that they are not simply scaled versions of the RGB spectra from the LED monitor. This is shown in Figure 5.6. Note that the RGB spectra from the LED look different from Figure 4.10 which was used for the AFSSI-C experiment because the transmission from the pelicle beam splitter also acts as a spectral filter.

The first-order optical architecture of the unmixing setup was carefully determined using the ray tracing and optimization software package Zemax. The goal was to ensure that the intermediate image of the LED monitor and the intermediate image of the OLED were at the same location with the same image size. The two images effectively create a single image which has spectrally mixed pixels with six possible endmembers. The pellicle beam splitter has a clear aperture of 2 inches, thus it needed to be close enough to the objective lens to minimize vignetting. A

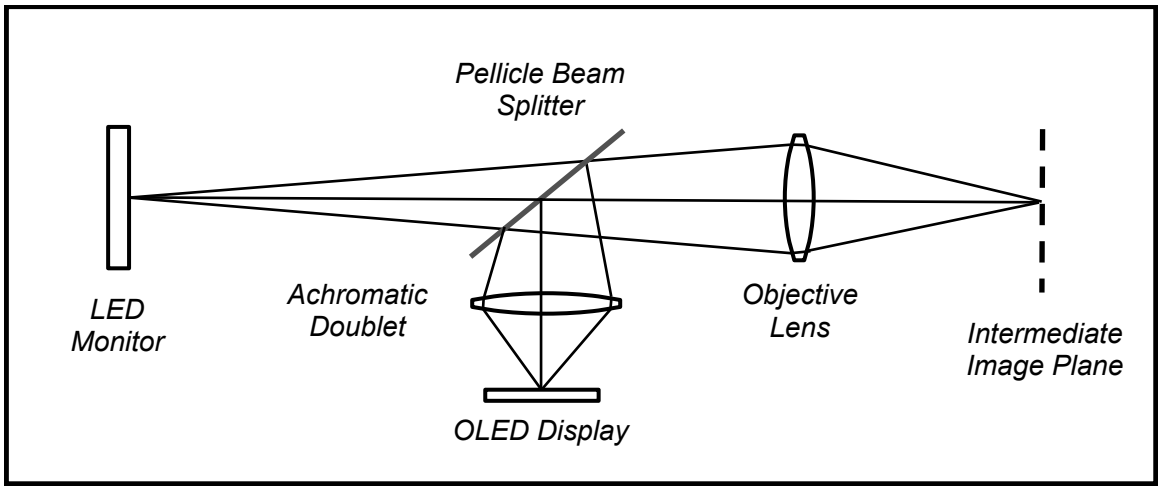

Figure 5.7: Experimental architecture used to superimpose the spectral images from the LED monitor, which already was in place, and an OLED display. ZEMAX was used to find the correct location of the achromatic doublet, OLED display, and pellicle beam splitter to ensure the intermediate image from both displays appeared at the same location and with the same transverse magnification.

commercial off-the-shelf 120 mm EFL achromatic doublet from Edmund optics was placed between the OLED and the pellicle beam splitter to move the OLED closer to the instrument. Once all the optical components were optimized in Zemax, they were exported to Solidworks, see Figure 5.8, where mounts for the achromatic doublet and the OLED display were designed for 3-D printing.

There were several practical experimental issues that had to be addressed. Once the setup was constructed, I used a similar spectral calibration procedure from the AFSSI-C experiment to determine the endmembers from the OLED display. The spatial calibration procedure was identical. The OLED was much brighter than the LED monitor, when measured by the SBIG camera. Second, even though the setup was optimized in Zemax, the actual alignment was imperfect. This caused the right side of the OLED to have a larger point-spread function than the left side of the monitor, which caused spatial cross-talk between neighboring pixels. To compensate for this, only one-fourth of the physical pixels were used in a single system pixel. Then a neutral density filter was placed front of the OLED to make the system pixel intensity on the same order-of-magnitude as the intensity from a system pixel from the monitor. Since the pellicle beam splitter acts as a partially reflecting mirror for OLED, I had to account for the image flip by writing a custom MATLAB function which accounts for the left-right flip.

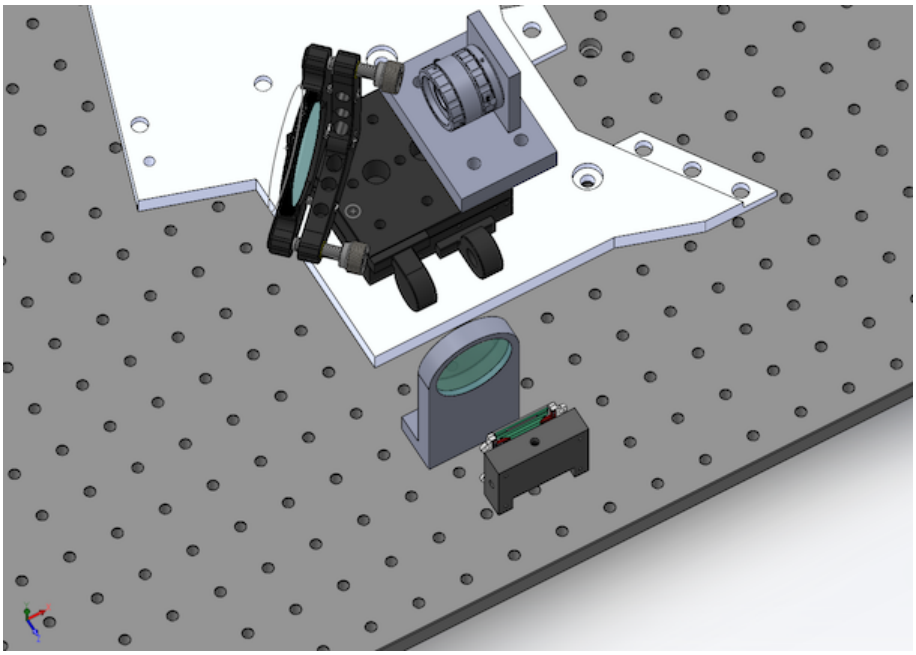

Figure 5.8: Computer Aided Design (CAD) drawing of experimental mixing setup. Various objects have been "hidden" to reduce clutter and emphasize the essential objects: the OLED, the custom design mount for the OLED, the achromatic doublet and the custom designed mount for the doublet, the pellicle beam splitter shown in a kinematic mount, the objective lens also in a custom fabricated C-mount lens.

In order to perform an experiment which demonstrates the ability of the AFSSI-C to perform unmixing on spectrally mixed images, I displayed a 32×32 green ramp on the monitor, see Figure 5.9(a). The intensity of the green increases from left to right. The opposite image is displayed on the OLED display, see Figure 5.9(b), the intensity of the green decreases from left to right. The other four endmembers which consists of the red and blue of each monitor are set to zero. The fractional abundance of green from both displays sum to one over the entire region-of-interest.

At each measurement step, the DMD is commanded to display a binary random pattern. Since the pattern is designed for $\{-1, +1\}$ values, I had to display the positive pattern, record the camera measurement, then display the negative pattern, and record another camera measurement, then subtract the negative from the positive camera images. This unfortunately introduces an additional factor of $\sqrt{\sigma}$ of measurement noise into each measurement step. Both the LSE and the MATLAB `lasso` function are used to estimate the fractional abundance. Figure 5.10 shows the average RMSE versus measurement number over the 32×32 region. Unfortunately, due to time constraints, I was unable to develop a procedure to estimate the

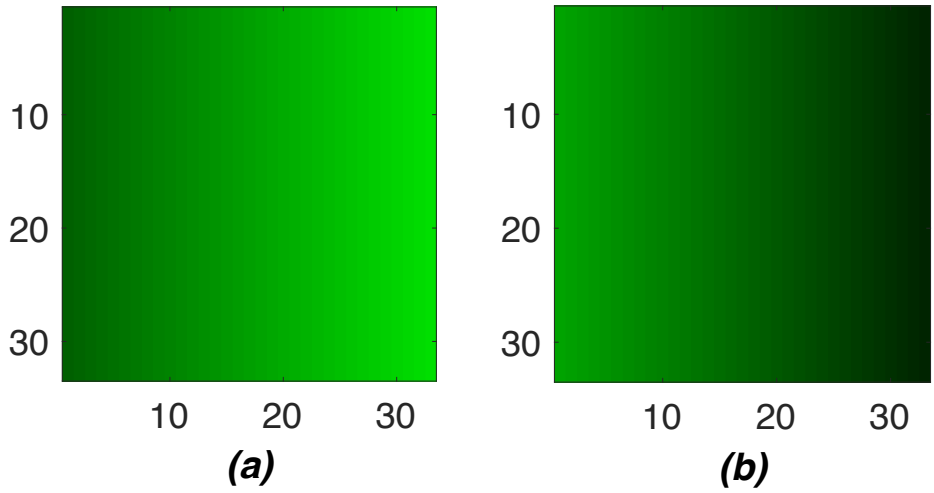

Figure 5.9: Ground Truth Images for Spectral Unmixing Experiment. (a) A linear green ramp on the monitor. (b) A linear green ramp on the OLED. Remember that the green of each display are different.

noise in the experiment, therefore the results cannot be directly compared to the simulation results. The result after the 40^{th} measurement is shown in Figure 5.11

By looking at an example pixel, see Figure 5.12, one can see that the experiment properly identified the two endmembers that were actually present out of the six total endmembers and was able to accurately estimate the fractional abundance after five measurements.

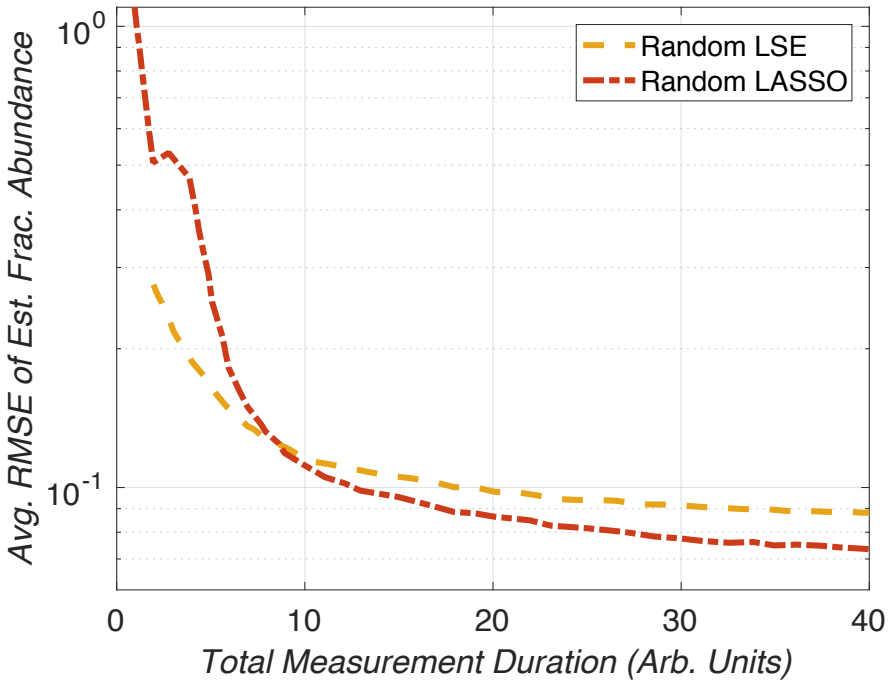

Figure 5.10: 32 × 32 Experimental Spectral Unmixing Results Using the AFSSI-C with random binary spectral filters. The MATLAB `lasso` function outperforms the least squares estimator as the number of measurements increase. Thus have an opportunity to beat traditional systems using significantly fewer measurements

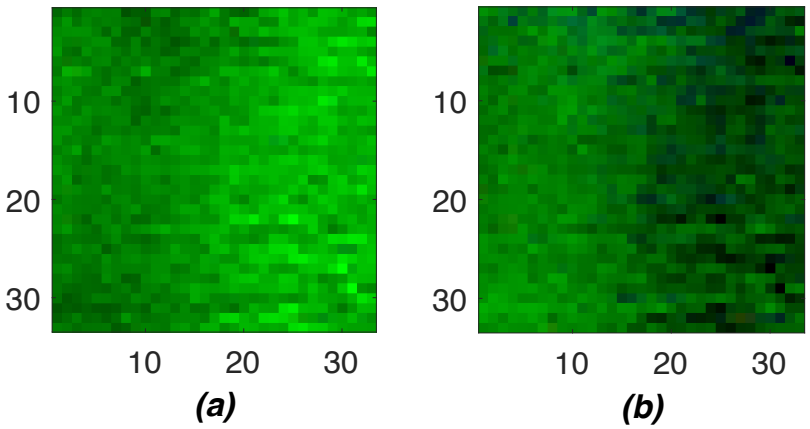

Figure 5.11: Image of 32 × 32 Experimental Compressive Spectral Unmixing Results Using the AFSSI-C after the 40^{th} measurement.

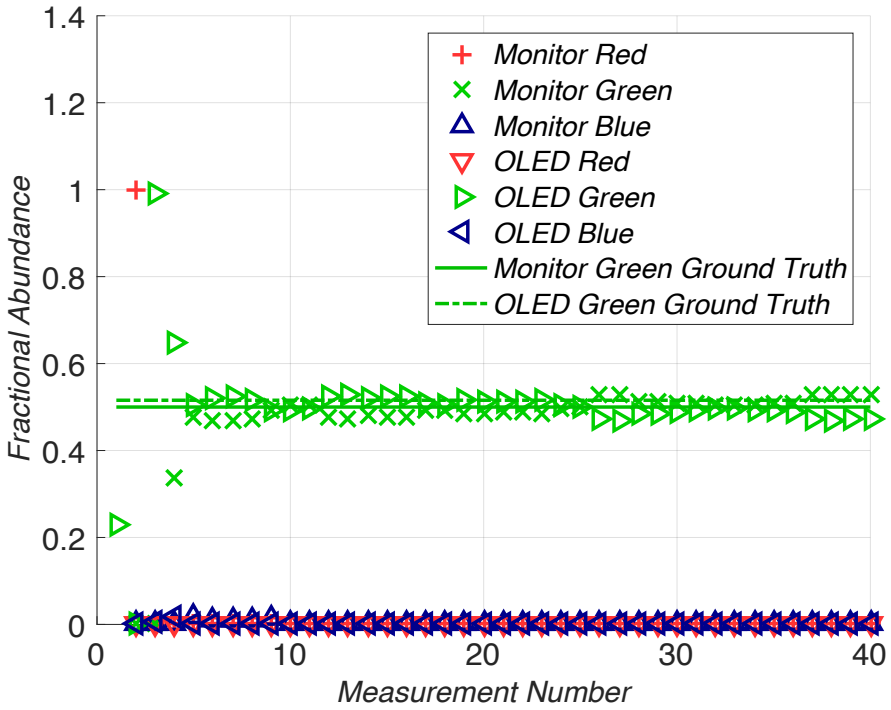

Figure 5.12: Single Pixel Spectral Unmixing Results with the AFSSI-C with random binary spectral filters. The MATLAB `lasso` function correctly identifies fractional abundance of the two green spectra from each display within five measurement steps. The four other spectra are estimated to be zero.

5.7.3 Simulation Results For the LCSI

In order to accurately simulate the spectral filters I first performed a calibration procedure to measure how each grayscale level filters an input spectrum. A high intensity custom-built white LED lamp was used to illuminate the LCSI. At the output we placed an Ocean Optics USB 4000 spectrometer. Then the spectrum of the white light filtered by LCSI at each grayscale level is recorded. Then the spectrum at each grayscale level is normalized by the maximum intensity at each wavelength which produces the calibration spectral filters shown in Figure 5.3.

In order to quantify the filter selection techniques discussed in Section 5.6.2, I also produced several simulations at SNR = 100. Figure 5.13 demonstrates the three techniques: randomly selecting filters, the hybrid PCA-Random filter selection, and the hybrid MNF-Random filter selection. Randomly selecting the filters adequately reduces the unmixing error monotonically with measurement error as expected. By using PCA to select the first six spectral filters and then using a psuedo-random filter

selection technique for the rest, a significant performance gain is obtained within the first 20 measurements. As the number of measurements increases, the diversity of filter sets decreases. At measurement 256, when all the spectral filters have been used, the average RMSE is the same. Using a hybrid MNF-Random filter selection produces an additional reduction in unmixing error.

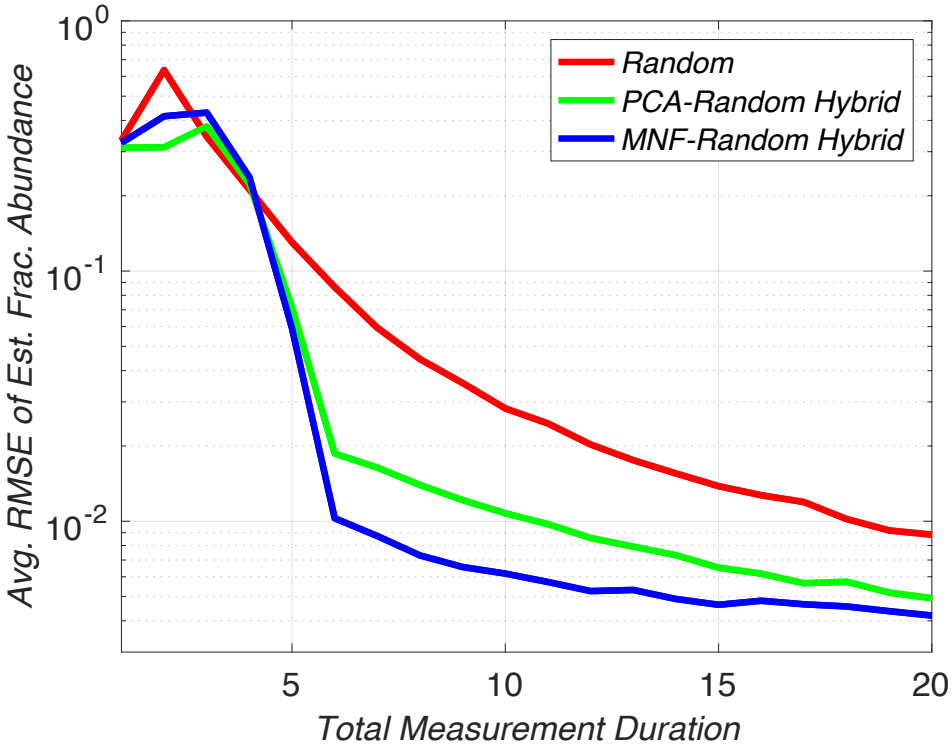

Figure 5.13: Comparing Random and Hybrid Spectral Filter Selections for the LCSI at SNR = 100. One cannot generate arbitrary spectral filters with the LCSI, therefore one must decide how to select from a set of 255 possible spectral filters. A hybrid of using the spectral filters that are closest in angle to the first six principal components and then using random filter selections outperforms purely random filter selections. Similarly, a hybrid of using the spectral filters that are closest in angle to the MNF components for the first six measurement steps and then using random filter selections outperforms the random and the PCA-random hybrid filter selection.

5.8 Conclusion

Computational spectral unmixing is a computational sensing approach to estimating the fractional abundances of mixed spectra without the need to directly reconstruct the hyperspectral datacube. I discussed two separate architectures for spectral unmixing, the DMD based AFSS and the LCOS bases LCSI. I developed two separate

approaches for coding the mixed spectral measurements for a compressing sensing approach for unmixing. The first used adaptive spectral filters combined with conditions to prevent the unmixing error from plateauing. The second relies on variation of Principal Component Analysis (PCA) or Maximum Noise Fraction (MNF) to find spectral filters provide very informative measurements compared to simply selecting spectral filters at random. Simulations show the viability these techniques to significantly reduce the unmixing error over naively random coding (or filter selection) alone, demonstrating the importance of incorporation the prior knowledge of the statistics of the signal-of-interest.

Chapter 6

Conclusion

This dissertation discussed several important practical considerations of computational sensing and how they are addressed in three separate applications: compressive object tracking, adaptive spectral image classification, and compressive spectral unmixing. As computational sensing continues to make rapid progress in the coming years many of these issues must be addressed and confronted.

The first chapter introduced the reader to the concepts of isomorphic and computational sensing. Computational sensing was developed from the disparate fields of multiplexing spectroscopy and indirect imaging. However, the development of the CCD, the digital computer, and data compression allowed scientists to begin creating practical and reliable demonstrations of computational sensing. The advent of the mathematics and algorithms of compressive sensing generated an additional technological leap which dramatically reduced the number of measurements in order to reconstruct the signal of interest. Once the history and benefits of computational sensing was discussed, I then discussed the practical issues of computational sensors: calibration, over multiplexing due to finite dynamic range and quantization resolution, code design and optimization, and the need for a priori knowledge.

The second chapter formally developed the concepts related to computational sensing. I discussed the Fellgett advantage, using Hadamard and S-matrix multiplexing. Then a formal discussion of Principal Component Analysis (PCA) was developed, which demonstrates how it can be used as a dimensionality reduction technique which creates a basis that decorrelates the data. Then a formal discussion on Bayesian statistics was given which justified their use in the AFSSI-C. Sparsity,

incoherence, and the restricted isometry property (RIP) were discussed in-depth to demonstrate why random coding is popular for acquiring sparse signals in compressive sensing. The ℓ_1 minimization subject to data agreement constraints and its equivalent problem minimizing the ℓ_1 regularized least squares objective function were also discussed with some intuition as to why it works well for promoting sparse solutions to undetermined inverse problems.

The Static Computational Optical Undersampled Tracker (SCOUT) was discussed in the third chapter. The SCOUT showed how intentionally blurring the point-spread function (PSF) in a compressive imager can actually help reduce the effects of overmultiplexing. As the amount of signals being sensed increases the variations in the sensed signal become more difficult to discern. Overmultiplexing occurs due to the finite dynamic range and quantization resolution. In the single pixel camera [11], one is forced to measure a series of projections in a time sequential manner prior to solving an inverse problem, in the SCOUT the projections occur in parallel for a single difference scene. The SCOUT introduced the reader to the importance of calibration in a compressive sensor. The measurement matrix must be carefully measured to produce optimal results. It also demonstrated that optimizing the measurement matrix can be extremely time consuming by hand. Developing a heuristic method based on the coherence metric and ray based simulations reduced the time it took to find optimal design parameters.

The fourth chapter discussed the Adaptive Feature Specific Spectral Imaging-Classifier (AFSSI-C) which is a computational spectral image sensor designed to adaptively classify spectra at each location in the field-of-view of the sensor. It used back-to-back spectrographs with a digital micro-mirror device (DMD) to produce psuedo-arbitrary spectral filters at each system pixel. After each measurement, an algorithm updates the probabilities of each spectra and weighted the spectral library before recomputing principal components. The first principal component vector is then used as the spectral filter for each measurement. The AFSSI-C is shown to quickly converge to the correct spectrum significantly outperforming traditional spectral imaging architectures especially in low TSNR scenarios. The chapter discussed the development of spatial and spectral calibration procedures, vital for optimal performance of the AFSSI-C. A procedure for estimating the system noise was also described in-depth.

The fifth chapter discussed an extension of spectral classification called spectral unmixing. Two architectures were discussed for spectral unmixing: the Adaptive Feature Specific Spectrometer (AFSS) and the LCOS Computational Spectral Imager (LCSI). The LCSI relies on the wavelength dependent nature of birefringence to generate multiplex spectral measurements. One of the major constraints of the LCSI is that psuedo-arbitrary spectral filters are not possible like in the AFSS or AFSSI-C, therefore I had to develop techniques for selecting spectral filters using random selections and a hybrid of using PCA or MNF to select spectral filters and then using the random filter selections.

6.1 Future Outlook

One research project that has been an active area of research in the Laboratory for Engineering Non-Traditional Sensors (LENS) is the coded memory effect imaging system being developed by my colleague Xiaohan Li. The project combines the memory effect of speckle with Coded Aperture Compressive Temporal Imaging (CACTI) to temporally code speckle. This allows one to infer dynamic object information that is not directly observable with the speckle alone.

Computational sensing continues to have an extremely promising future. Large companies are actively conducting research and developing products which take advantage of the principles of computational sensing. For example, Microsoft sells a depth sensor, called the Kinect, which relies on a form computational sensing to infer the distance of objects and their shapes. Meanwhile, a recently announced grant from NASA will fund the development of snapshot hyperspectral imagers based on the Computed Tomography Imaging Spectrometer (CTIS) for space-borne applications [112]. Many university based research groups around the world are actively conducting research that demonstrate compact spatial, spectral, temporal, and polarization computational sensing. Computational sensing will become even more prevalent as the demand for higher resolutions in resource constrained environments abound.

Appendix A

Derivation of the Least Squares Estimator

Suppose

$$\mathbf{g} = \mathbf{A}\mathbf{x} \tag{A.1}$$

Given $\mathbf{g}$ and $\mathbf{A}$ we want to solve for $\mathbf{x}$. If the matrix is full rank then we can simply multiply both sides of equation A.1 by $\mathbf{A}^{-1}$

$$\mathbf{A}^{-1}\mathbf{g} = \mathbf{A}^{-1}\mathbf{A}\mathbf{x} = \mathbf{I}\mathbf{x} = \mathbf{x} \tag{A.2}$$

where $\mathbf{I}$ is the identity matrix.

If $\mathbf{A}$ is not full rank then its inverse does not exist. However we can try to find a solution $\hat{\mathbf{x}}$ that minimizes the squared error. This is called the *Least Squares Solution* also known as the *Least Squares Estimator*, *Ordinary Least Squares* and by many other names. We define the squared error as

$$\|\epsilon\|^2 = \|\mathbf{A}\mathbf{x} - \mathbf{g}\|^2 \tag{A.3}$$

To minimize the error, we take the derivative of equation A.3 with respect to $\mathbf{x}$ and set it equal to zero and solve for $\mathbf{x}$. Note that the equation A.3 can be expanded in terms of an inner product

$$\|\epsilon\|^2 = \|\mathbf{A}\mathbf{x} - \mathbf{g}\|^2 = \sum_{i=1}^{N} \epsilon_i^2 = \epsilon^T\epsilon = (\mathbf{A}\mathbf{x} - \mathbf{g})^T(\mathbf{A}\mathbf{x} - \mathbf{g}) \tag{A.4}$$

The transpose is distributive

$$(\mathbf{Ax} - \mathbf{g})^T = (\mathbf{Ax})^T - \mathbf{g}^T \tag{A.5}$$

The transpose of a product of matrices equals the product of their transposes in reverse order

$$(\mathbf{Ax})^T = \mathbf{x}^T\mathbf{A}^T \tag{A.6}$$

So equation A.4 becomes

$$\begin{aligned} \|\epsilon\|^2 &= (\mathbf{x}^T\mathbf{A}^T - \mathbf{g}^T)(\mathbf{Ax} - \mathbf{g}) \\ &= \mathbf{x}^T\mathbf{A}^T\mathbf{Ax} - \mathbf{x}^T\mathbf{A}^T\mathbf{g} - \mathbf{g}^T\mathbf{Ax} + \mathbf{g}^T\mathbf{g} \end{aligned} \tag{A.7}$$

We can see that the two middle terms $\mathbf{x}^T\mathbf{A}^T\mathbf{x} = \mathbf{g}^T\mathbf{Ax}$ because they are just scalars.

$$\|\epsilon\|^2 = \mathbf{x}^T\mathbf{A}^T\mathbf{Ax} - 2\mathbf{g}^T\mathbf{Ax} + \mathbf{g}^T\mathbf{g} \tag{A.8}$$

To find the least squares solution, take the gradient with respect to $\mathbf{x}$ and set it equal to zero.

It should be noted that there are two different notations for writing the derivative of a vector with respect to a vector $\frac{\partial \mathbf{y}}{\partial \mathbf{x}}$. If the numerator $\mathbf{y}$ is of size m and the denominator $\mathbf{x}$ of size n, then the result can be laid out as either an $m \times n$ matrix or $n \times m$ matrix, i.e. the elements of $\mathbf{y}$ laid out in columns and the elements of $\mathbf{x}$ laid out in rows, or vice versa. They are both correct and equal, which leads to confusion when switching back in forth. I will write both to reduce confusion.

Clearly the gradient of the third term in equation A.8 w.r.t $\mathbf{x}$ is 0, so it goes away. We first tackle the first term on the right hand side in equation A.8

$$\frac{\partial}{\partial \mathbf{x}}\mathbf{x}^T\mathbf{A}^T\mathbf{Ax} \tag{A.9}$$

Let $\mathbf{K} = \mathbf{A}^T\mathbf{A}$. Since $\mathbf{K}$ is symmetric, we can use the identity

$$\frac{\partial}{\partial \mathbf{x}}\mathbf{x}^T\mathbf{Kx} = 2\mathbf{x}^T\mathbf{K} = 2\mathbf{K}^T\mathbf{x} \tag{A.10}$$

since $\mathbf{K} = \mathbf{K}^T$ then

$$\frac{\partial}{\partial \mathbf{x}}\mathbf{x}^T\mathbf{A}^T\mathbf{Ax} = 2\mathbf{x}^T\mathbf{A}^T\mathbf{A} = 2\mathbf{A}^T\mathbf{Ax} \tag{A.11}$$

and the gradient of the middle term in equation A.8 is simply $-2\mathbf{A}^T\mathbf{g}$ so

$$\frac{\partial}{\partial \mathbf{x}}\|\epsilon\|^2 = 2\mathbf{A}^T\mathbf{A}\mathbf{x} - 2\mathbf{g}^T\mathbf{A} \tag{A.12}$$

setting it equal to zero and solving for $\mathbf{x}$ gives the least squares estimate

$$\hat{\mathbf{x}} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{g} \tag{A.13}$$

Appendix B

SCOUT Experimental Results

This appendix contains the experimental reconstruction results of two movers in 10 difference frames from the zero (black) background SCOUT data. It also contains the experimental reconstruction results of one mover in 10 difference frames from non-zero (black) background. Each figure is a reconstructed difference frame from a video. Due to the nature of books, showing video is impossible, the next best thing is to show several representative frames from the video.

B.1 Zero Background Difference Frames

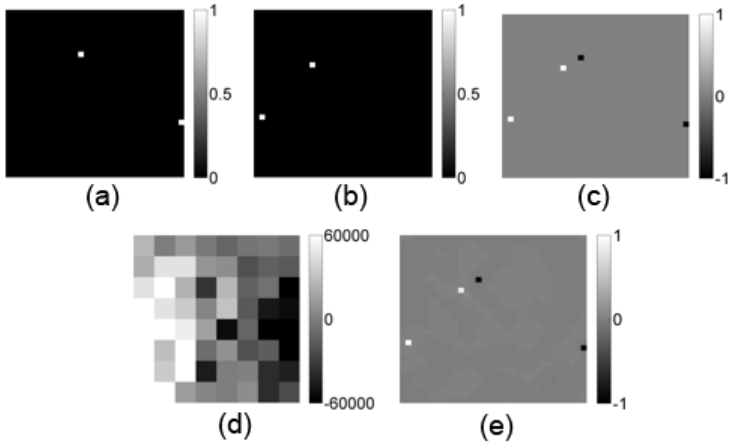

Figure B.1: Difference frame 1 of a sequence of one mover on a black background.

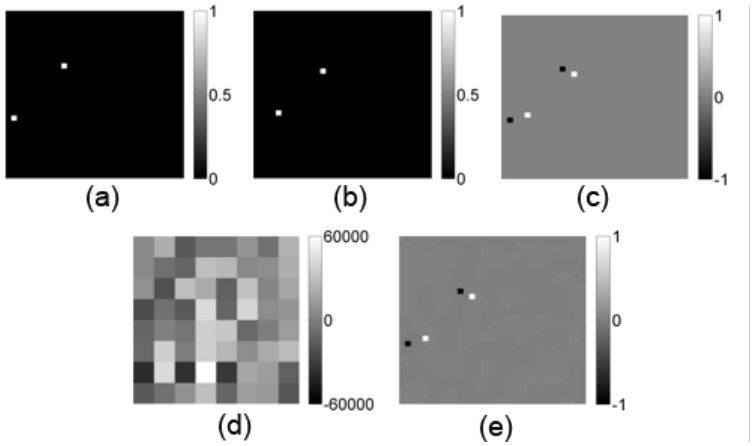

Figure B.2: Difference frame 2 of a sequence of one mover on a black background.

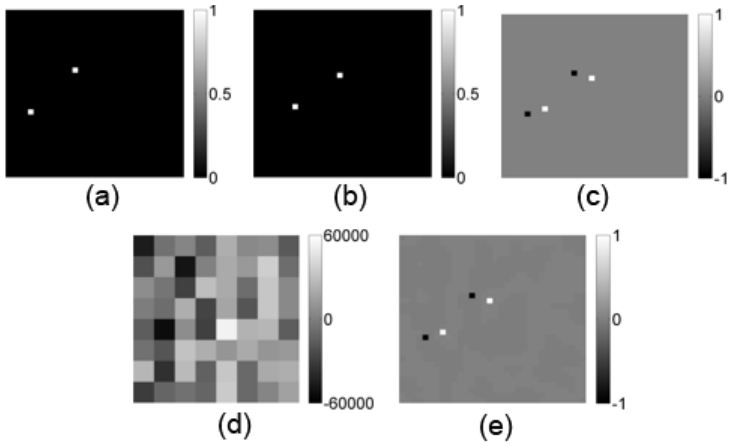

Figure B.3: Difference frame 3 of a sequence of one mover on a black background.

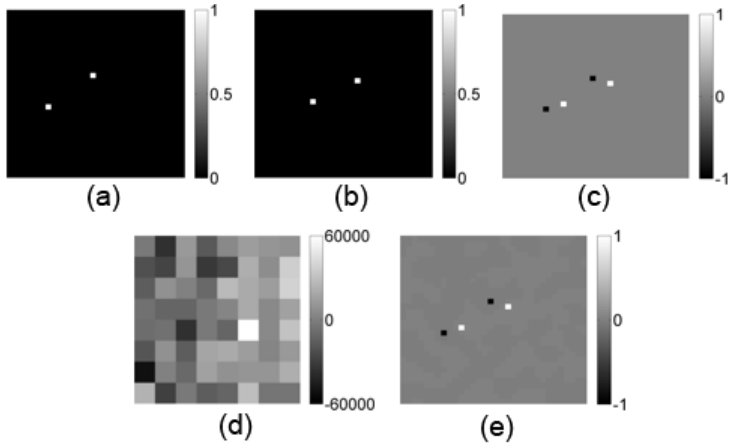

Figure B.4: Difference frame 4 of a sequence of one mover on a black background.

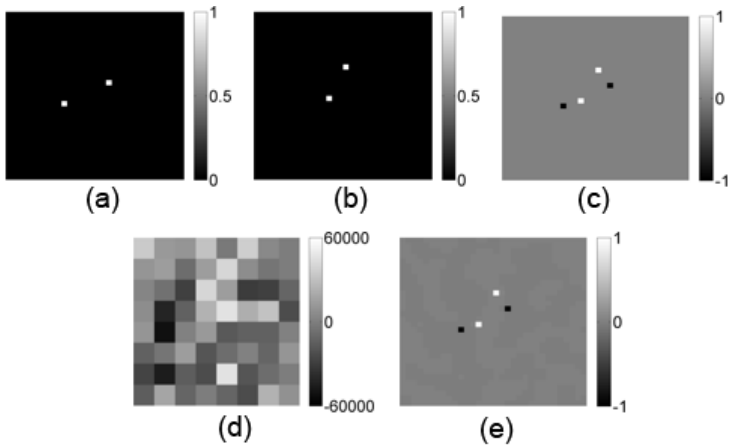

Figure B.5: Difference frame 5 of a sequence of one mover on a black background.

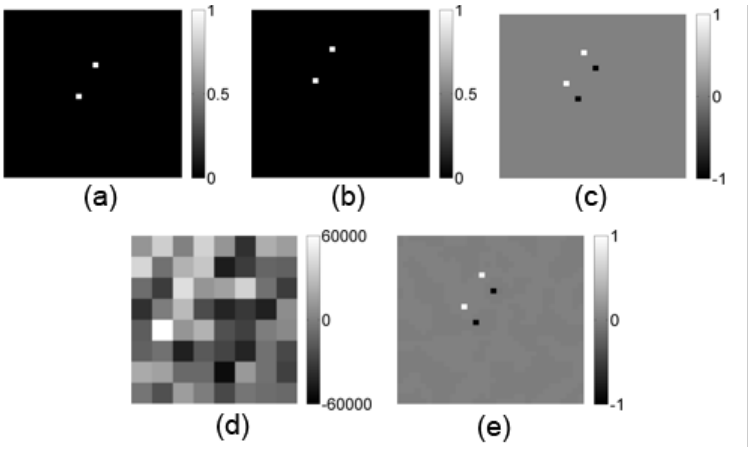

Figure B.6: Difference frame 6 of a sequence of one mover on a black background.

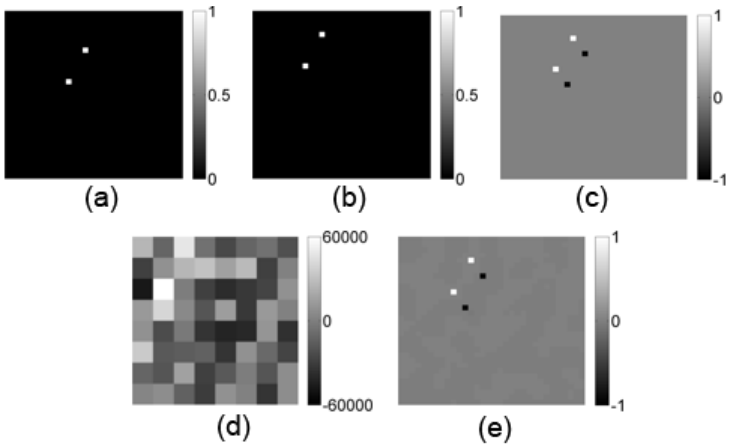

Figure B.7: Difference frame 7 of a sequence of one mover on a black background.

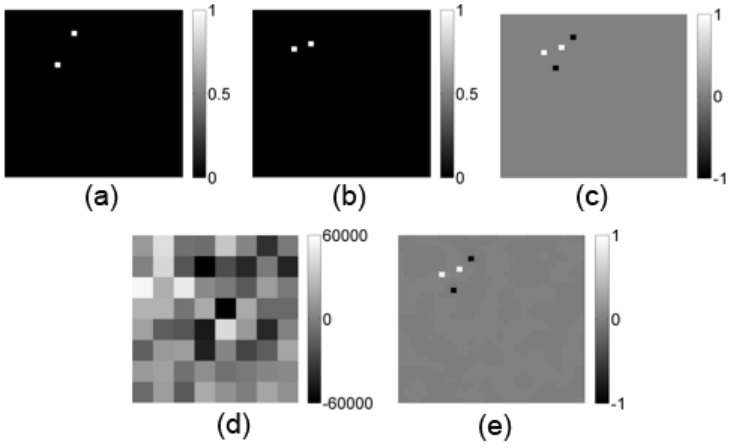

Figure B.8: Difference frame 8 of a sequence of one mover on a black background.

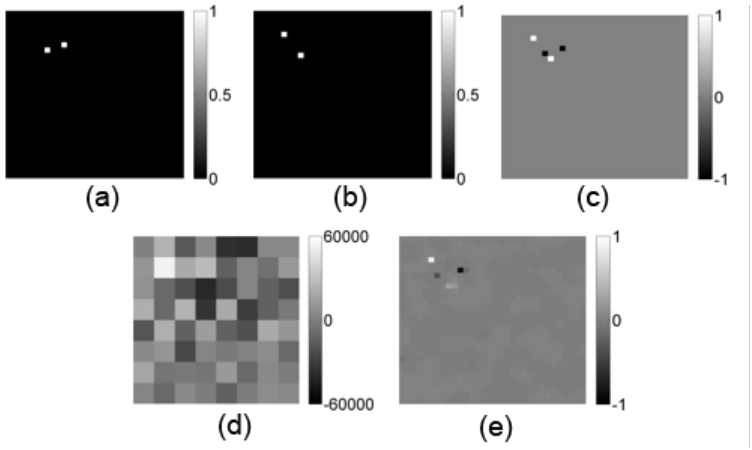

Figure B.9: Difference frame 9 of a sequence of one mover on a black background.

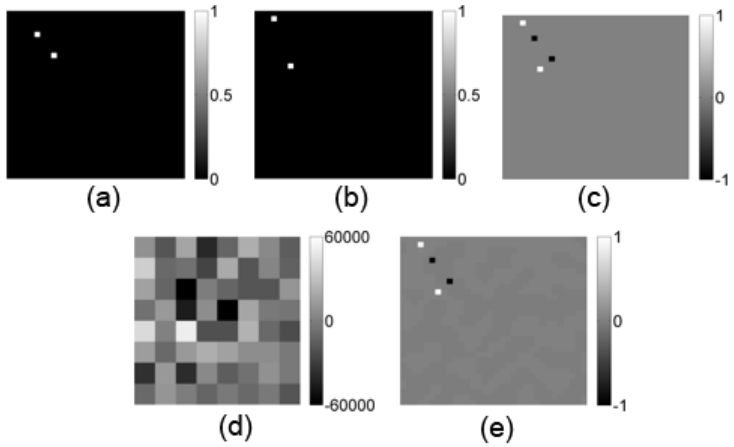

Figure B.10: Difference frame 10 of a sequence of one mover on a black background.

B.2 Non-Zero Background Difference Frames

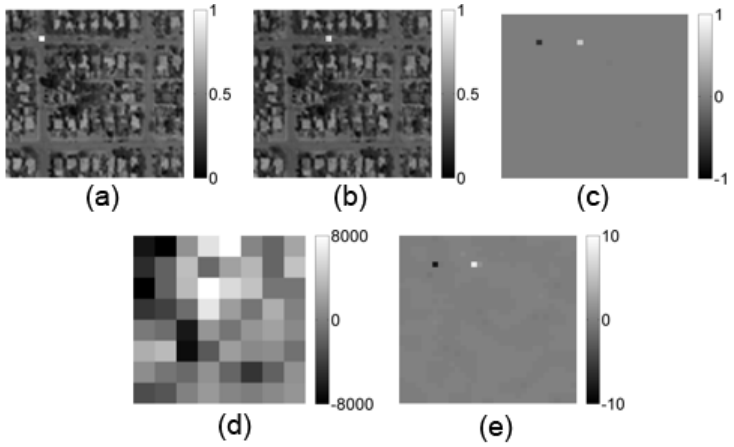

Figure B.11: Difference frame 1 of a sequence of one mover on a non-zero background.

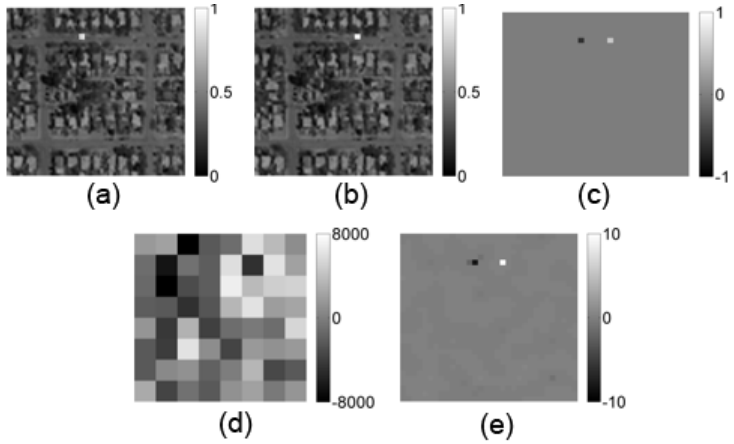

Figure B.12: Difference frame 2 of a sequence of one mover on a non-zero background.

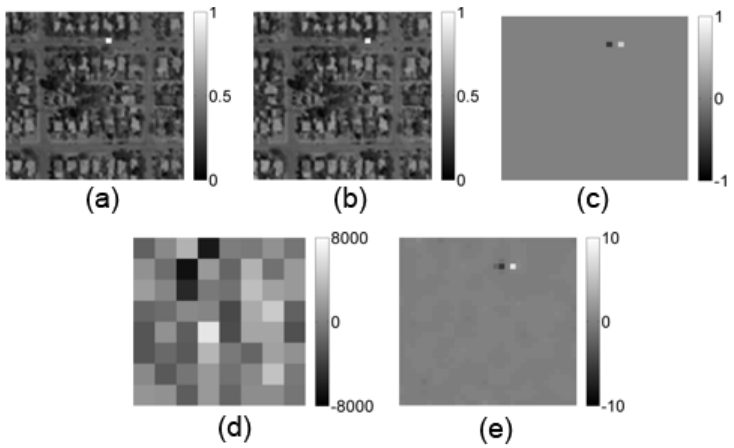

Figure B.13: Difference frame 3 of a sequence of one mover on a non-zero background.

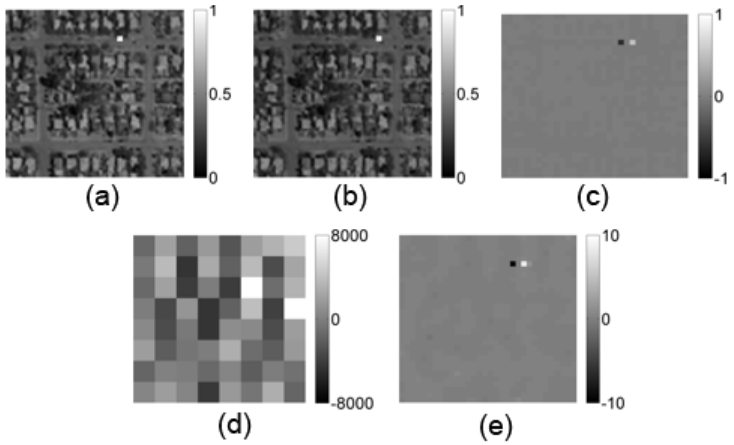

Figure B.14: Difference frame 4 of a sequence of one mover on a non-zero background.

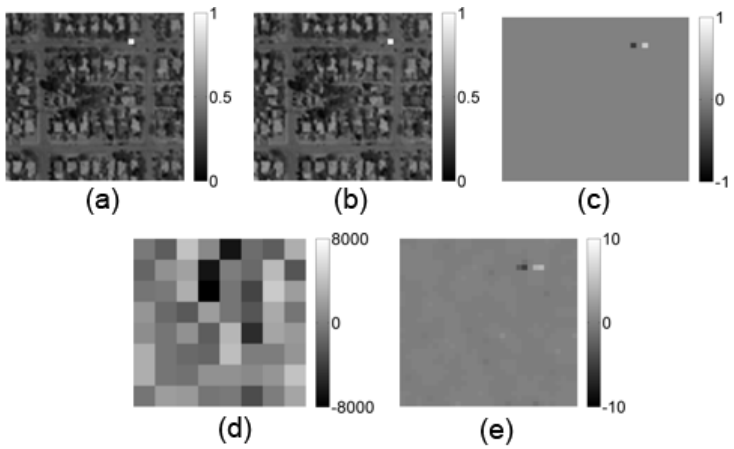

Figure B.15: Difference frame 5 of a sequence of one mover on a non-zero background.

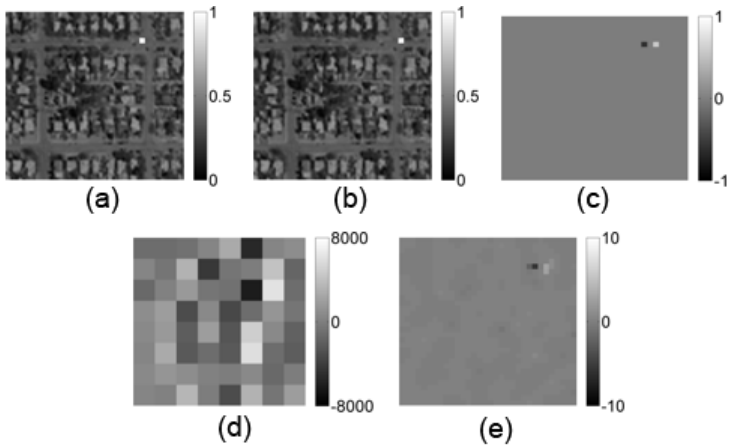

Figure B.16: Difference frame 6 of a sequence of one mover on a non-zero background.

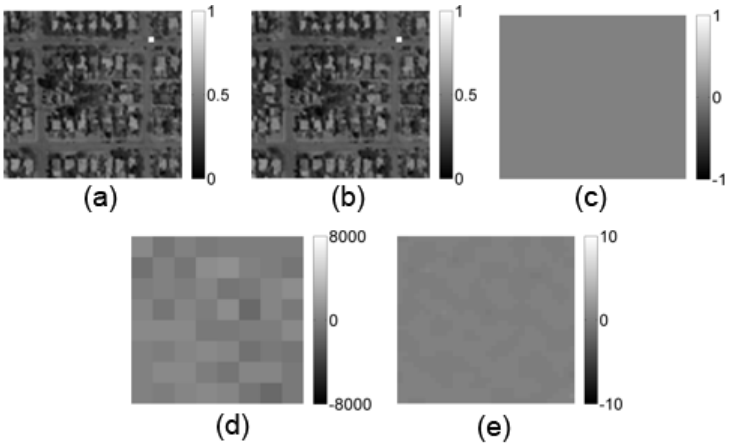

Figure B.17: Difference frame 7 of a sequence of one mover on a non-zero background.

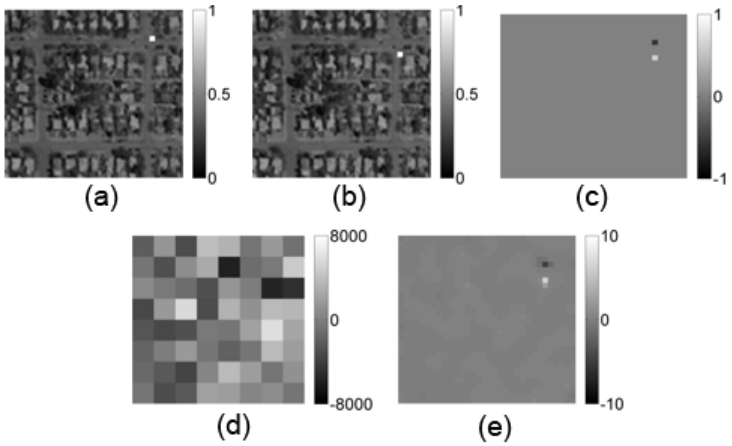

Figure B.18: Difference frame 8 of a sequence of one mover on a non-zero background.

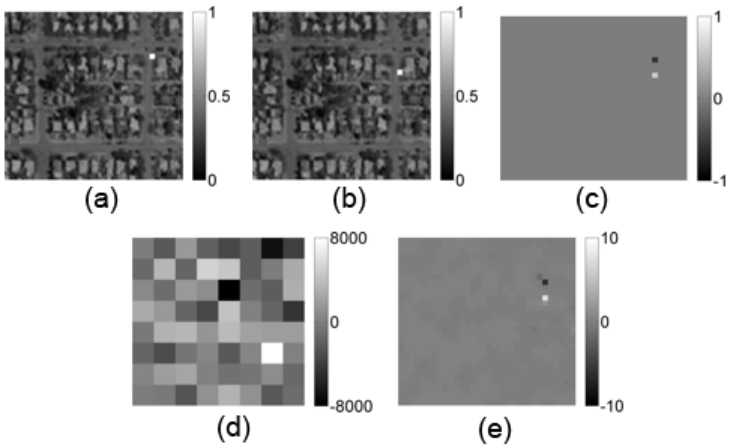

Figure B.19: Difference frame 9 of a sequence of one mover on a non-zero background.

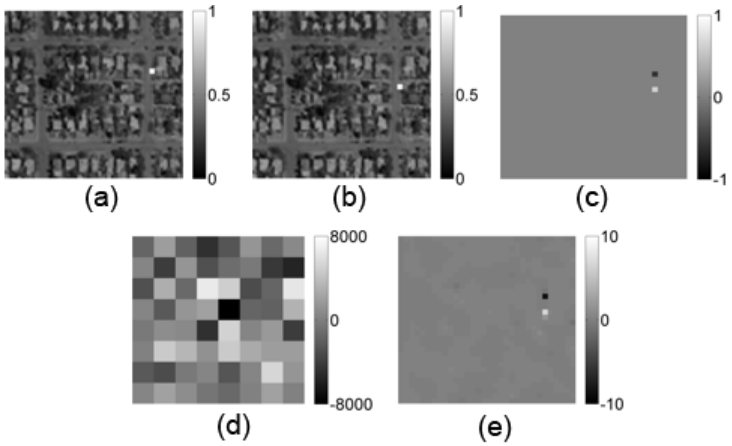

Figure B.20: Difference frame 10 of a sequence of one mover on a non-zero background.

Appendix C

The Psuedo-Code For SCOUT Experiment

This appendix contains the psuedo-code for running the SCOUT experiment which is described in Chapter 3.

Algorithm 1 Dark Frame Measurement algorithm

1: **function** getDarkFrame
2: Open Connection to Camera
3: Set camera temperature to 0 degrees Celsius
4: Start a stopwatch timer
5: $tv := 60$ seconds
6: $c := 0$
7: **for** $k = 1$ **to** numExp **do** {Loop takes $numExp$ camera exposures}
8: **begin**
9: $te :=$ Read the stopwatch timer (units of seconds).
10: **if** $te > c * tv$ **then**
11: **begin**
12: Display an all white screen on monitor
13: $c := c + 1$
14: **end**
15: Sets the entire plamsa to 0
16: $df(k) :=$ Camera Readout
17: **end**
18: Close Connection to Camera
19: average dark frame measurements
20: Return averaged dark frame measurements

Algorithm 2 SCOUT Calibration algorithm

```
1: function getHcal
2: df := getDarkFrame                {Runs the Dark Frame Measurement Function}
3: Open Connection to Camera
4: Set camera temperature to 0 degrees Celsius
5: Start a stopwatch timer
6: tv := 60 seconds
7: c := 0
8: locList := randomize list of locations n to N
9: for n = 1 to N do
10:     begin
11:     te := Read the stopwatch timer (units of seconds).
12:     if te > c * tv then
13:         begin
14:         Display an all white screen on monitor
15:         c := c + 1
16:         end
17:     Sets the entire plamsa to 0
18:     Turn on location locList(n)
19:     p := Camera Readout
20:     Sets the entire plamsa to 0
21:     p := p − df      {Subtract dark frame from the readout of the nth location}
22:     Convert image to vector
23:     Store in the nth column of H
24:     end
25: Close Connection to Camera
26: Save H
```

Algorithm 3 Take Measurements for SCOUT Zero Background

1: **function** scoutMeasZeroBg
2: Open Connection to Camera
3: Set camera temperature to 0 degrees Celsius
4: Start a stopwatch timer
5: $tv := 60$ seconds
6: $c := 0$
7: **for** $k = 1$ **to** numFrames **do**
8: **begin**
9: $te :=$ Read the stopwatch timer (units of seconds).
10: **if** $te > c * tv$ **then**
11: **begin**
12: Display an all white screen on monitor
13: $c := c + 1$
14: **end**
15: Sets the entire plamsa to 0
16: $movLocList(k) :=$ generate random list of mover locations over all possible N locations
17: Create image $f(k)$ where all zero except the locations from $movLocList(k)$.
18: Display image $f(k)$ {Display the ground truth frame}
19: $r :=$ Camera Readout
20: $g(k) :=$ Convert camera readout r to vector
21: Sets the entire plamsa to 0
22: **end**
23: Close Connection to Camera
24: Save all measurements and ground truth frames g, f

Algorithm 4 Reconstruct SCOUT Experiment Data

1: **function** reconScoutExp
2: Load all measurements g
3: Load H {From the calibration function getHcal}
4: **for** $k = 2$ **to** numFrames **do**
5: **begin**
6: $\Delta g(k-1) := g(k) - g(k-1)$ {Subtract previous measurement from current one}
7: $\tau := (1 \times 10^{-9}) \left(2H^T \Delta g(k-1)\right)$
8: $reltol := 1 \times 10^{-9}$
9: $\Delta \hat{f}(k-1) := l1\_ls(H, \Delta g, \tau, reltol, quiet)$ {Reconstruct the difference frame using the $l1_ls$ MATLAB code. }
10: **end**
11: Return all reconstructed frame differences $\Delta \hat{f}$

Appendix D

SCOUT Simulation Code

Listing D.1: Main Simulation Script

```
% GENERATING SYSTEM MEASUREMENT MATRIX 'H'

clc; clearvars; close all
rand_trials = 5
% lens to camera image sensor defocus in mm
defocus_value=35;
% pitch (in mm)of mask1 (next to image sensor)
p1=0.01;
% pitch (in mm) of mask2 (next to lens)
p2=0.5;
%focal length of lens
f=35;

%% comments about image sensor pitch
%  reading 288 x 288 part on image sensor and binning
%  (36x36) to obtain 8x8 pixels
%  sensor pitch is 9 microns. For binnned 8x8, each pixel
%  is 36*9=   324 microns

% pitch of image sensor in mm
p=0.324;
```

```
22
23 for i = 1:rand_trials
24
25     % d1 is distance between Mask 1 and image plane of lens
26     % here 2.04 mm is the spacing between camera sensor
27     % image plane and Mask 1
28     d1=defocus_value-2.04;
29
30     % d2 is distance between Mask 2 and image plane of lens
31     % here 6.35 mm is the spacing between
32     d2=f-6.35;
33
34     % calculating pitch and positions relative to detector
35     % for locating Mask1 away from focus
36     pos1 = (defocus_value -d1)/(f+defocus_value);
37
38     % position of Mask 2
39     pos2=(d2+defocus_value)/(f+defocus_value);
40
41     %calculating pitch values of Masks on detector
42     pd1=(p1*defocus_value)/d1; % for mask1
43     pd2=(p2*defocus_value)/d2; % for mask2
44     pitch1=pd1/p;
45     pitch2=pd2/p;
46
47
48     pitch = [pitch1 pitch2];
49     pos=[pos1 pos2];
50
51     % Mask leaakge based on experimental mask transmission
52     % values
53     leakage1=0.12;%masks open part leakage of 12%
54     leakage=0.00;%masks black part no leakage
```

```
55
56      % fill factor of mask1 and mask2 , 50%
57      through=[0.5 0.5];
58
59      lens_defocus=defocus_value;
60
61      %scaling mask patterns, using smaller values speeds up
           code
62      upsamp=0.25;
63
64      directory=pwd;
65
66      date='feb16'; %date for file names
67
68      name = ...
69          sprintf(['%s/pos_%.3g_%.3g_through_%.3g_%.3g_'...
70              'pitch_%.3g_%.3g.mat'],...
71          directory,pos(1),pos(2),through(1),through(2),...
72          pitch(1),pitch(2));
73
74      % choose blur type on masks
75      psfblur='fourierpsf';
76      %psfblur='false' for no blur
77      % for adding blur based on fourier filter
78      %psfblur='fourierpsf'
79
80      varargin={'upsample',upsamp,'mask_pos',...
81          pos,'throughput',through,'maskpitch', ...
82          pitch,'defocus', lens_defocus,'psfblur',...
83          psfblur,'leakage',leakage,'leakage1',leakage1};
84
85      N=32;%size of the scene (NxN)
86      K=8;%size of detector (KxK)
```

```
87
88      % call function to generate system matrix, with point
89      % by point calibration. Selecting display as 'false'
90      % to prevent figure display while generating matrix
91      H(:,:,i) = struct_projection(N, 'display',false,...
92          'compression', N/K, varargin{:});
93
94
95      save(['condnum_df_' num2str(defocus_value) ...
96          'normalization_on.mat'],'H','rand_trials')
97
98  end
99
100     % use the statement below for locating Mask1 before the
            focus (i.e. on the same side of focus as Mask 2)
101     %  pos1=(d1+defocus_value)/(f+defocus_value);
102
103     pos2=(d2+defocus_value)/(f+defocus_value); % position
           of Mask 2
104
105     %calculating pitch values of Masks on detector
106     pd1=(p1*defocus_value)/d1; % for mask1
107     pd2=(p2*defocus_value)/d2; % for mask2
108     pitch1=pd1/p;
109     pitch2=pd2/p;
110
111
112     pitch = [pitch1 pitch2];
113     pos=[pos1 pos2];
114
115     %Mask leaakge based on experimental mask transmission
           values
116     leakage1=0.12;%masks open part leakage of 12%
```

```
117     leakage=0.00;%masks black part no leakage
118
119
120     through=[0.5 0.5]; % fill factor of mask1 and mask2 ,
           50%
121     lens_defocus=defocus_value;
122     upsamp=0.25; %scaling mask patterns, using smaller
           values speeds up code
123     directory=pwd;
124     date='feb16'; %date for file names
125
126     name = sprintf('%s/pos_%.3g_%.3g_through_%.3g_%.3
           g_pitch_%.3g_%.3g.mat',directory,pos(1),pos(2),
           through(1),through(2),pitch(1),pitch(2));
127
128     %%choose blur type on masks
129     psfblur='fourierpsf';
130     %psfblur='false' for no blur
131     %psfblur='fourierpsf' for adding blur based on fourier
           filter
132
133
134     varargin={'upsample',upsamp,'mask_pos',pos,'throughput'
           ,through,'maskpitch', pitch,'defocus', lens_defocus,
           'psfblur',psfblur,'leakage',leakage,'leakage1',
           leakage1};
135
136     N=32;%size of the scene (NxN)
137     K=8;%size of detector (KxK)
138
139     % call function to generate system matrix, with point
           by point calibration.
```

```
140     %selecting display as 'false' to prevent figure display
            while generating matrix
141     H(:,:,i) = struct_projection(N, 'display',false,...
142         'compression', N/K, varargin{:});
143
144     % system_mat_name=sprintf('%s_defocus_%d.mat',date,
            lens_defocus);% file
145     % name to save system matrix 'H'
146
147     %     system_mat_name = ['trial' num2str(i) '
            _default_matrix.mat']
148     save(['condnum_df_' num2str(defocus_value) '
            normalization_on.mat'],'H','rand_trials')
149
150 end
```

Listing D.2: struct_projection function

```
1  function H = struct_projection(N,varargin)
2
3  defopt.mask_pos = [.1 .9];
4  defopt.compression = 4;
5  defopt.throughput = 0.5;
6  defopt.defocus = 2;
7  defopt.display = true;
8  defopt.upsample = 4;
9
10 % Sets pitch equal to reconstruction resolution
11 defopt.maskpitch = 1;
12 defopt.psfblur='false';
13 defopt.leakage1=0;%mask clear part
14 defopt.leakage=0;%mask dark part
15
16 opt = matlab_options(varargin, defopt);
```

```
17
18 compression = opt.compression;
19 throughput = opt.throughput;
20 dd = opt.defocus;
21 leakage = opt.leakage;
22 leakage1 = opt.leakage1;
23
24 upsample = opt.upsample;  % linear upsample factor
25
26 display = opt.display;
27
28 psfblur=opt.psfblur; % read blur type
29
30 % all physical distances are in mm
31
32
33 N=32;% scene size NxN
34
35 % location pitch - desired image-space reconstruction
36 % resolution in mm
37 p  = 0.324;
38
39 f = 35; % lens focal length in mm
40 D=25.4; %lens diameter in mm
41
42 maskcount = length(opt.mask_pos);
43
44 %for locating Mask1 away from focus
45 d1 = dd - opt.mask_pos(1)*(f+dd);
46
47 % use the statement below for locating Mask1 before the
48 % focus (i.e. on the same side of focus as Mask 2)
49
```

```
%%% testing for defocus=1, mask1 behind mask2

if maskcount>1
  d2 = opt.mask_pos(2)*(f+dd) - dd; %%%edit
else
  d2 = 0;
end
if maskcount>2
  d3 = dd+opt.mask_pos(3)*(f+dd); %%edit
else
  d3 = 0;
end

if maskcount>3
  error('mask_pos is too long, must have length 1-3');
end

throughput = throughput(:);
if length(throughput)==1 && maskcount>1
  throughput=throughput^(1/maskcount)*ones(maskcount,1);
end
if length(throughput) <3
  throughput = [throughput;ones(3-length(throughput))];
end

if length(opt.maskpitch) == 1 && maskcount > 1
  maskpitch=opt.maskpitch*ones(maskcount,1);
else
  maskpitch = opt.maskpitch(:);
end

% Code assumes all masks present even if they aren't,
% so pad to length 3
```

```
83  maskpitch = [maskpitch ; ones(3 - length(maskpitch),1)];
84
85  % make mask pitch (pd) equal to reconstruction pitch (p)
86  pd = maskpitch * p; % mask pitch in sensor plane
87
88  %% perform geometrical calculations
89
90  p1 = pd(1)*d1/dd; % physical mask pitch---- mm real mask
91  p2 = pd(2)*d2/dd;
92  p3 = pd(3)*d3/dd;
93
94  % this is the pitch of the points in the aperture,
95  % a larger defocus (dd) gives a smaller pitch
96  pL = p*f/dd;
97
98  fprintf('mask 1: d1 = %.1f, p1 = %.3f\n', d1, p1);
99  fprintf('mask 2: d2 = %.1f, p2 = %.3f\n', d2, p2);
100 fprintf('mask 3: d3 = %.1f, p3 = %.3f\n', d3, p3);
101
102 Sf = p * (N-1) / (1 + dd/f);  % full "image" width at focus
        (mm)
103 Sd = Sf * (1 + dd/f); % image width at sensor (at z = dd)
104 S1 = Sf * (1 + d1/f); % at mask 1
105 S2 = Sf * (1 + d2/f); % at mask 2
106
107 S3 = Sf * (1 - d3/f); % at mask 3
108
109 Bd = D * dd/f; % blur width at sensor
110 B1 = D * d1/f;
111 B2 = D * d2/f;
112 B3 = D * d3/f;
113
114 %% define mask structures
```

```
115 m0.z = f; % aperture
116 m0.x = [-D/2:pL:D/2];
117 m0.y = m0.x;
118 [X,Y] = meshgrid(m0.x, m0.y);
119
120 %%## adding gaussian blur to model lens defocus
121
122 % 'blursize' indicates the size of the blur filter which is
123 % computed as per the defocus level above(bigger blur for a
      larger defocus).
124 blursize=size(X);
125
126 % the standard deviation of 12 is chosen to model lens
127 % defocus with Gaussian blur
128 m0.v=fspecial('gaussian',blursize(1),12);
129
130 m1.z = d1; % first mask
131 m1.x = [-(B1+S1)/2:p1:(B1+S1)/2];
132 m1.y = m1.x;
133 m1.v = double(rand(length(m1.x)) <= throughput(1));
134
135 % including leakage of dark part
136 leak = leakage*ones(length(m1.v));
137
138 % including leakage of clear part
139 leak1 = leakage1*ones(length(m1.v));
140 m1.v = max(m1.v.*(1-leak1),leak);
141 m2.z = d2; % second mask
142 m2.x = [-(B2+S2)/2:p2:(B2+S2)/2];
143 m2.y = m2.x;
144 m2.v = double(rand(length(m2.x)) <= throughput(2));
145
146 % including leakage of dark part
```

```
147 leak = leakage*ones(length(m2.v));
148
149 % including leakage of clear part
150 leak1 = leakage1*ones(length(m2.v));
151 m2.v = max(m2.v.*(1-leak1),leak);
152 m3.z = d3; % third mask
153 m3.x = [-(B3+S3)/2:p3:(B3+S3)/2];
154 m3.y = m3.x;
155 m3.v = double(rand(length(m3.x)) <= throughput(3));
156 % including leakage of dark part
157 leak = leakage*ones(length(m3.v));
158 % including leakage of clear part
159 leak1 = leakage1*ones(length(m3.v));
160 m3.v = max(m3.v.*(1-leak1),leak);
161
162 %% perform calculations
163 H = [];Hcrop=[];
164 % range and sampling for masks in sensor plane
165 xx = [-Sd/2:pd/upsample:Sd/2];
166 % range of point locations (mapped to image space)
167 xlist = [-Sd/2:p:Sd/2];
168 progress(1, length(xlist)^2, 1, 1);
169 ind = 1;
170
171 countval=length(xlist);
172
173 for xcount=1:countval
174     for ycount=1:countval
175         x=xlist(xcount);
176         y=xlist(ycount);
177         progress(ind); ind = ind + 1;
178         [mout, ml] = overlay_masks({m0, m1, m2}, ...
179             [x,y], dd, xx, xx,B1,B2,Bd,psfblur,D);
```

```
        full_image = mout.v;
        scalefac = upsample * p / pd;
        high_res = imresize(full_image, [N,N], 'bilinear');
        % default compression of 4 for  32x32 scene to
        % 8x8 detector
        low_res = bin_image(high_res, compression);

        if display
            if gcf ~= 2; figure(2); end
            subplot(1,2,1);
            imagesc(high_res); axis image; colormap gray;
            subplot(1,2,2);
            imagesc(low_res); axis image; colormap gray;
            drawnow;
        end

        H = [H low_res(:)]; % build the system matrix

    end
end
if display
  figure(3);
  imagesc(H);
  colormap jet;
  colorbar;
end
```

Listing D.3: overlay_masks function

```
function [m_out, ml] = overlay_masks(mask_list, origin, ...
    z, x, y,B1,B2,Bd,psfblur,D)
% project a list of masks to given z plane and overlay them
%   mask_list - list of mask structures
%   origin - [x, y] coordinates of projection origin
```

```
%   z       - z-plane to project to
%   x, y    - x and y coordinate lists to interpolate onto

if nargin < 4
    x = ml{1}.x;
    y = ml{1}.y;
end

% project the masks to common z plane
if nargin > 1
    for ind = 1:length(mask_list)
        ml{ind} = project_mask(mask_list{ind}, origin, z);
    end
else
    ml = mask_list;
end

% build the combined mask structure
m_out = ml{1};
m_out.x = x;
m_out.y = y;
[X,Y] = meshgrid(x,y); % gridded coordinates for output
   mask
m_out.v = ones(size(X));

for ind = 1:length(ml)
    [Xm, Ym] = meshgrid(ml{ind}.x, ml{ind}.y); % native
       projected coordinates
    if 0
        fprintf('ind = %i\n', ind);
        fprintf('size(Xm) = [%i, %i]\n', size(Xm,1), ...
            size(Xm,2));
        fprintf('size(Ym) = [%i, %i]\n', size(Ym,1), ...
```

```
37              size(Ym,2));
38          fprintf('size(v)   = [%i, %i]\n', ...
39              size(ml{ind}.v,1), size(ml{ind}.v,2));
40          fprintf('size(X)   = [%i, %i]\n', size(X,1), ...
41              size(X,2));
42          fprintf('size(Y)   = [%i, %i]\n', size(X,1), ...
43              size(X,2));
44      end
45      % interpolate from the native projected coordinates
46      % onto the common coordinates
47
48      switch (psfblur)
49
50          case 'fourierpsf' % adding blur on masks
51              switch (ind)
52                  case 1
53                      m_out.v = m_out.v .* interp2(Xm, Ym,
                            ...
54                          ml{ind}.v, X, Y, '*linear', 0);
55
56
57                  case 2 %blur on mask 1
58                      [sz1,sz2]=size((interp2(Xm, Ym, ...
59                          ml{ind}.v, X, Y, '*linear', 0)));
60
61                      %create filter in fourier domain
62                      blur_radius1=(sz1*((D-B1)/D))/2;
63                      % compute blur radius at chosen defocus
64                      % level
65                      [rr cc] = meshgrid(1:sz1);
66                      blurpsf1 = fftshift(sqrt((rr-round(sz1
                            /2)).^2+(cc-round(sz1/2)).^2)<=
                            blur_radius1);
```

```
67                  blur_filter1 = abs((ifft2(blurpsf1)));
68                  %apply filter
69
70                  filtered_mask =  ...
71                                              real(ifft2(
                                                   blurpsf1
                                                   .*fft2(
                                                   interp2(
                                                   Xm, Ym,
                                                   ml{ind}.
                                                   v, X, Y,
                                                    '*
                                                   linear',
                                                    0))));
72
73                  %normalized the filtered mask to it can
                       't amplify
74                  filtered_mask = filtered_mask / max(
                       filtered_mask(:));
75
76                  m_out.v = m_out.v .* filtered_mask;
77                  m_out.v(m_out.v<0)=0;
78
79              case 3 %blur on mask 2
80
81                  [sz1,sz2]=size((interp2(Xm, Ym, ml{ind
                       }.v, X, Y, '*linear', 0)));
82
83                  %create filter in fourier domain
84                  blur_radius2=(sz1*((D-B2)/D))/2; %
                       compute blur radius at chosen
                       defocus level
85                  [rr cc] = meshgrid(1:sz1);
```

```
86                      blurpsf2 = fftshift(sqrt((rr-round(sz1
                           /2)).^2+(cc-round(sz1/2)).^2)<=
                           blur_radius2);
87                      %apply filter
88                      blur_filter2 = abs(ifft2(blurpsf2));
89
90                      filtered_mask = ...
91                                                  real(ifft2(
                                                       blurpsf2
                                                       .*fft2(
                                                       interp2(
                                                       Xm, Ym,
                                                       ml{ind}.
                                                       v, X, Y,
                                                        '*
                                                       linear',
                                                        0))));
92
93                      filtered_mask = filtered_mask / max(
                           filtered_mask(:));
94                      m_out.v = m_out.v .* filtered_mask;
95                      m_out.v(m_out.v<0)=0;
96              end
97
98          case 'false' %no blur
99              m_out.v = m_out.v .* interp2(Xm, Ym, ml{ind}.v,
                    X, Y, '*linear', 0);
100
101     end
102
103 end
```

Listing D.4: project_mask function

```
1
2  function m_out = project_mask(m_in, origin, new_z)
3  % project the input mask to a new z plane.  The projection
       is performed
4  % via rays emanating from the provided x-y origin in the z
       =0 plane
5
6  % mask attributes: z, v, x, y (optional)
7  % if y is provided, v should be 2d, otherwise 1d
8
9  m_out.z = new_z;
10 m_out.v = m_in.v;
11 x0 = origin(1);
12 m_out.x = (m_in.x - x0) * (m_out.z/m_in.z) + x0;
13 if length(origin) > 1
14     y0 = origin(2);
15     m_out.y = (m_in.y - y0) * (m_out.z/m_in.z) + y0;
16 end
17
18 %calling function to locate the center of the aperture at
       the point source
19 %co-ordinates
20
21  % call function to center PSF
22 [m_out.x m_out.y]=centering_atlocation(origin,m_out.x,m_out
       .y);
23
24 end
```

Listing D.5: centering_atlocation function

```
1  function [out1, out2]=centering_atlocation(origin,in1,in2)
2
```

```
%function to locate the center of the aperture at the point source co-ordinates

m_out.x=in1;
m_out.y=in2;

%####calculations for x-cordinate####
len_mx=length(m_out.x);

test_even_odd=mod(len_mx,2);
if test_even_odd==0
    %even length
    mid_pt=(len_mx/2);
elseif test_even_odd==1
    %odd length
    mid_pt=((len_mx+1)/2);
end
% calculating mid point
midval=m_out.x(mid_pt);
m_out.x=(m_out.x - midval +  origin(1));%center the data around origin


%####calculations for y-cordinate####
len_my=length(m_out.y);
test_even_odd=mod(len_my,2);
if test_even_odd==0
    %even length
    mid_pty=(len_my/2);
else
    %odd length
    mid_pty=((len_my+1)/2);
end
```

```
midvaly=m_out.y(mid_pty);
m_out.y=(m_out.y - midvaly +  origin(2));%center the data around origin

out1=m_out.x;
out2=m_out.y;
%@@@@@@@@@@@@
```

Listing D.6: bin_image function

```
function imout = bin_image(im, n)
% bin images
% im - image (or set of images) to be binned
% n  - number of elements (linearly) to be binned
%
% n can be a vector of bin sizes in each dimension.  If n is a scalar,
% it is interpreted as
%    [n, n]       for 2D images,
%    [n, n, 1]    for 3D images,
%    [n, n, 1, 1] for 4D images, etc.
%
% examples:
%  size(im)         n        -->  size(imout)
%  [50, 50]         5        -->  [10, 10]
%  [50, 50, 3]      5        -->  [10, 10, 3]
%  [50, 50, 100]    [5,5,20] -->  [10, 10, 5]
%
% note: if the dimensions of im are not multiples of the elements of n,
% then the last few values of im will not be used.  That is,
%   size(im) = [51, 51]   and  n = 5   -->  size(imout) = [10, 10]
```

```
% and the last row and column of im [im(:,51) and im(51,:)] are unused.



% expand n into vector form if necessary
if length(n) == 1
    ntmp = n;
    n = ones(1, length(size(im)));
    n(1) = ntmp;
    n(2) = ntmp;
end

s = floor(size(im)./n); % target size
S.type = '()';  % used in subsref and subsasgn
subscell = {};  %
for d = 1:length(s)
    subscell{d} = ':';
end

for d = 1:length(s)  % loop over each dimension
    if n(d) == 1; % no binning along this dimension - move on
        continue
    end
    S.subs = subscell;
    ts = size(im);
    ts(d) = s(d);
    imout = zeros(ts);
    for i = 1:s(d)
        start = (i-1)*n(d)+1;
        stop  = i*n(d);
        S.subs{d} = [start:stop];
```

```
52          M = subsref(im, S);
53          S.subs{d} = i;
54          imout = subsasgn(imout, S, sum(M, d));
55      end
56      if d < length(s) % only need to reassign if we're not
           done yet
57          im = imout;
58      end
59  end
```

Appendix E

Derivation of the Update Rule for Log-Likelihood Ratios

In order to update the conditional probability of the i^{th} hypothesis given the full set of measurements $\{g\}_m$ we can use Bayes' theorem

$$\mathrm{P}\left(h_i|\{g\}_m\right) = \frac{\mathrm{P}\left(\{g\}_m|\, h_i\right)\ \mathrm{P}\left(h_i\right)}{\mathrm{P}\left(\{g\}_m\right)}. \tag{E.1}$$

We really have no way of directly calculating $\mathrm{P}\left(\{g\}_m\right)$. We can avoid the need to compute this by taking ratios. In our case, the ratio of the probability of the i^{th} spectrum given the set of all measurements up to the current measurement m $\{g_m\}$ to the probability of the j^{th} spectrum given the set of all measurements up to the current measurement m $\{g_m\}$ is

$$L_{ij}^{\{g\}_m} = \frac{\mathrm{P}\left(\{g\}_m \mid h_i\right)\mathrm{P}\left(h_i\right)}{\mathrm{P}\left(\{g\}_m \mid h_j\right)\mathrm{P}\left(h_j\right)} \tag{E.2}$$

We will assume that the joint probability are independent and can be written as a product, thus we can rewrite the about equation as:

$$L_{ij}^{\{g\}_m} = \frac{\mathrm{P}(g_m \mid h_i)\prod_{m'=1}^{m-1}\mathrm{P}(g_{m'} \mid h_i)\mathrm{P}\left(h_i\right)}{\mathrm{P}(g_m \mid h_j)\prod_{m'=1}^{m-1}\mathrm{P}(g_{m'} \mid h_i)\mathrm{P}\left(h_j\right)} \tag{E.3}$$

If we define the ratio of likelihoods up to the last measurement $m-1$ as

$$L_{ij}^{\{g_{m-1}\}} = \frac{\prod_{m'=1}^{m-1}\mathrm{P}(g_{m'} \mid h_i)\mathrm{P}\left(h_i\right)}{\prod_{m'=1}^{m-1}\mathrm{P}(g_{m'} \mid h_i)\mathrm{P}\left(h_j\right)} \tag{E.4}$$

then we can write the equation for the update procedure as

$$L_{ij}^{\{g\}_m} = \frac{\mathrm{P}(g_m \mid h_i)}{\mathrm{P}(g_m \mid h_j)} L_{ij}^{\{g_{m-1}\}} \tag{E.5}$$

At the beginning before any measurements are taken, $m = 0$, we have no bias towards any of the hypotheses. At $m = 0$

$$L_{ij}^{g_0} = \frac{\mathrm{P}(h_i)}{\mathrm{P}(h_j)} \tag{E.6}$$

Since we have no bias we set all of our probabilities at $m = 0$ to

$$\mathrm{P}(h_i) = \frac{1}{N}. \tag{E.7}$$

In other words, each spectra is equiprobable. This is known as a "uniform prior". Thus before any measurements are made by definition:

$$L_{ij}^{g_0} = 1 \text{ for all } i, j \tag{E.8}$$

We use a matrix to track the all the possible pairs of ratios. For example with a 3-class library

$$\mathbf{L}^{\{g\}_m} = \begin{bmatrix} L_{11}^{\{g\}_m} & L_{12}^{\{g\}_m} & L_{13}^{\{g\}_m} \\ L_{21}^{\{g\}_m} & L_{22}^{\{g\}_m} & L_{23}^{\{g\}_m} \\ L_{31}^{\{g\}_m} & L_{32}^{\{g\}_m} & L_{33}^{\{g\}_m} \end{bmatrix} \tag{E.9}$$

Based on the eq. E.8, the initial likelihood ratio matrix at the measurement step $m = 0$ is

$$\mathbf{L}^{g_1} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \tag{E.10}$$

Now that we have our initial conditions, and our update procedure from eq E.5, we still haven't discussed a way to calculate $P(g_m|h_i)$, the probability of observing a measurement g_m given that the i^{th} spectrum is the true spectrum. This is where we introduce our noise model. The noise model, gives the probability of the measurement we just made g_m, if we assume some Gaussian noise distribution $\mathcal{N}(0, \sigma)$.

$$\mathrm{P}(g_m|h_i) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{(g_m - \mathbf{t}_m \cdot \mathbf{s}_i)^2}{2\sigma^2}\right] \tag{E.11}$$

Remember g_m is the noisy measurement and $\mathbf{t}_m$ is the spectral code realized by the DMD pattern. For example, a given measurement number m, in the case of 3 possible spectra the inner products $\mathbf{t}_m \cdot \mathbf{s}_1$, $\mathbf{t}_m \cdot \mathbf{s}_2$, $\mathbf{t}_m \cdot \mathbf{s}_3$ will all have different values. These inner products are deterministic because $\mathbf{t}_m$ is something we choose and $\mathbf{s}_l$ for $m' = 1, 2, 3$ comprise our library and by definition are constant. Plug these inner products into the equation above assuming a given σ, and we will get different values of $\mathrm{P}(g_m|h_1)$, $\mathrm{P}(g_m|h_2)$, and $\mathrm{P}(g_m|h_3)$

In the picture above we see have 3 Gaussian distributions for each of the 3 possible spectra. The measurement $g_1 = \mathbf{t}_1 \cdot \mathbf{s}_{true} + \mathscr{N}(0, \sigma_a) \approx 0.05$ as shown on the horizontal axis. Where this intercepts the functions gives the values of the probability of the measurement given knowledge of each spectrum.

$$\mathrm{P}(g_1|s_1) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{(g_1 - \mathbf{t}_1 \cdot s_1)^2}{2\sigma^2}\right] \approx 0.30 \tag{E.12}$$

$$\mathrm{P}(g_1|s_2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{(g_1 - \mathbf{t}_1 \cdot s_2)^2}{2\sigma^2}\right] \approx 0.40 \tag{E.13}$$

$$\mathrm{P}(g_1|s_3) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{(g_1 - \mathbf{t}_1 \cdot s_3)^2}{2\sigma^2}\right] \approx 0.26 \tag{E.14}$$

The likelihood ratios become:

$$L_{12}^{g=1} = \frac{\mathrm{P}(g_1|h_1)}{\mathrm{P}(g_1|h_2)} \approx .13 \tag{E.15}$$

$$L_{13}^{g=1} = \frac{\mathrm{P}(g_1|h_1)}{\mathrm{P}(g_1|h_3)} \approx 0.25 \tag{E.16}$$

$$L_{23}^{g=1} = \frac{\mathrm{P}(g_1|h_2)}{\mathrm{P}(g_1|h_3)} \approx 1.9 \tag{E.17}$$

The diagonal will always be 1 and the elements $L_{21}^{\{g_m=1\}}$, $L_{31}^{\{g_m=1\}}$, and $L_{32}^{\{g_m=1\}}$ are just the inverse of the upper right elements in the likelihood ratio matrix.

$$\mathbf{L}^{g_1} = \begin{bmatrix} 1.0 & 0.7 & 1.2 \\ 1.3 & 1.0 & 1.6 \\ 0.9 & 0.6 & 1.0 \end{bmatrix} \tag{E.18}$$

For $m = 2$ use the update procedure

$$L_{ij}^{\{g_2,g_1\}} = \frac{\mathrm{P}(g_2 \mid h_i)}{\mathrm{P}(g_2 \mid h_j)} L_{ij}^{g_1} \tag{E.19}$$

then store the data back into the likelihood ratio matrix.

However during computing, exponentials often cause numerical problems. In MATLAB 2013b, the exponential of a number larger than approximately 709 is so large that MATLAB simply call them infinity (Inf). If these are in the denominators of the likelihood ratios then computers will simply call them not a number (NaN). To avoid, these computational issues use the logserious-likelihood ratio to eliminate the exponentials

$$\mathscr{L}_{ij}^{\{g_m\}} = \ln\left[L_{ij}^{\{g_m\}}\right] = \ln\left[\frac{\mathrm{P}(\{g\}_m|h_i)\mathrm{P}(h_i)}{\mathrm{P}(\{g\}_m|h_j)\mathrm{P}(h_j)}\right] \tag{E.20}$$

$$\mathscr{L}_{ij}^{\{g_m\}} = \ln\left[\frac{\mathrm{P}(g_m|h_i)}{\mathrm{P}(g_m|h_j)}L_{ij}^{\{g_{m-1}\}}\right] \tag{E.21}$$

We will now derive a simple update procedure. The log of a product is the sum of the logs and we can rewrite the last term:

$$\mathscr{L}_{ij}^{\{g_m\}} = \ln\left[\frac{\mathrm{P}(g_m|h_i)}{\mathrm{P}(g_m|h_j)}\right] + \mathscr{L}_{ij}^{\{g_{m-1}\}} \tag{E.22}$$

The log of a division is the difference of the logs.

$$\mathscr{L}_{ij}^{\{g_m\}} = \ln\left[\mathrm{P}(g_m|h_i)\right] - \ln\left[\mathrm{P}(g_m|h_j)\right] + \mathscr{L}_{ij}^{\{g_{m-1}\}} \tag{E.23}$$

Plug in equation E.11 for the conditional probabilities.

$$\begin{aligned}\mathscr{L}_{ij}^{\{g_m\}} = \ln\left\{\frac{1}{\sqrt{2\pi\sigma^2}}\exp\left[-\frac{(g_m - f_m\cdot h_i)^2}{2\sigma^2}\right]\right\} \\ - \ln\left\{\frac{1}{\sqrt{2\pi\sigma^2}}\exp\left[-\frac{(g_m - f_m\cdot h_j)^2}{2\sigma^2}\right]\right\} + \mathscr{L}_{ij}^{\{g_{m-1}\}}\end{aligned} \tag{E.24}$$

Continue expanding the number of terms in order to find terms that will cancel.

$$\begin{aligned}\mathscr{L}_{ij}^{\{g_m\}} = \ln\left\{\frac{1}{\sqrt{2\pi\sigma^2}}\right\} + \ln\left\{\exp\left[-\frac{(g_m - f_m\cdot h_i)^2}{2\sigma^2}\right]\right\} \\ - \ln\left\{\frac{1}{\sqrt{2\pi\sigma^2}}\right\} - \ln\left\{\exp\left[-\frac{(g_m - f_m\cdot h_i)^2}{2\sigma^2}\right]\right\} + \mathscr{L}_{ij}^{\{g_{m-1}\}}\end{aligned} \tag{E.25}$$

Two terms cancel. The natural log of an exponential is just the argument of the exponential.

$$\mathscr{L}_{ij}^{\{g_m\}} = -\frac{(g_m - f_m \cdot h_i)^2}{2\sigma^2} + \frac{(g_m - f_m \cdot h_j)^2}{2\sigma^2} + \mathscr{L}_{ij}^{\{g_{m-1}\}} \tag{E.26}$$

$$\boxed{\mathscr{L}_{ij}^{\{g_m\}} = \mathscr{L}_{ij}^{g_m} + \mathscr{L}_{ij}^{\{g_{m-1}\}}} \tag{E.27}$$

As you can see we now have a simplified update procedure which only requires addition to the previous set of log-likelihood ratios. We have completely avoided the computing the exponential of larger numbers!

Now that we have the newly updated log-likelihood ratios $\mathscr{L}_{ij}^{\{g_m\}}$, we can calculate the newly updated probabilities of each candidate spectra $\mathrm{P}(h_i|g_m)$.

We will discuss an example that shows how we begin the update procedure after the first measurement. For $m = 1$

$$\mathscr{L}_{ij}^{g_1} = -\frac{(g_1 - f_m \cdot h_i)^2}{2\sigma^2} + \frac{(g_1 - f_m \cdot h_j)^2}{2\sigma^2} + \mathscr{L}_{ij}^{g_0} \tag{E.28}$$

From equation E.6 and E.7 we know that $L_{ij}^{g_0} = 1$. The natural log of 1 is 0.

$$\mathscr{L}_{ij}^{g_0} = \ln\left(L_{ij}^{g_0}\right) = \ln(1) = 0 \tag{E.29}$$

We now have an equation that we can use to begin our update procedure

$$\mathscr{L}_{ij}^{g_1} = -\frac{(g_1 - f_m \cdot h_i)^2}{2\sigma^2} + \frac{(g_1 - f_m \cdot h_j)^2}{2\sigma^2} \tag{E.30}$$

E.1 Calculating the conditional probabilities from the log-likelihood ratio matrix

Similar to the likelihood ratio matrix, the log-likelihood ratio matrix is used to keep track of the ratio of the probabilities of each hypothesis.

After each measurement m, each log-likelihood ratio is updated using equation E.27

Then the conditional probabilities of each hypothesis $\mathrm{P}(h_l|\{g\}_m)$ can be calculated using the following algorithm:

1. Determine the row of with the maximum element

$$\texttt{maxRow := findRow}(\mathscr{L}^{\{g\}_m})) \tag{E.31}$$

2. Once you know the row, get the corresponding column

$$\mathscr{L}^{\{g_m\}}_{i,\texttt{maxRow}} \tag{E.32}$$

3. Take the exponentials for all the elements in that column. To get the unnormalized probabilities.

$$\exp(\mathscr{L}^{\{g_m\}}_{i,\texttt{maxRow}}) \tag{E.33}$$

4. Normalize probabilities assuming all the denominators are equal to 1 and then by dividing each element by the sum

$$\mathrm{P}(h_l|\,\{g\}_m) = \frac{\exp(\mathscr{L}^{\{g_m\}}_{i,\texttt{maxRow}})}{\sum_{i=1}^{N_R}\exp(\mathscr{L}^{\{g_m\}}_{i,\texttt{maxRow}})} \tag{E.34}$$

5. The true spectrum is the hypothesis with the largest probability.

$$s_{true} := \texttt{max}\,[\mathrm{P}(s_l)] \tag{E.35}$$

Appendix F

AFSSI-C Experimental Results

This appendix contains the examples of experimental classification results of a 64×64 spatial scene with a 4-class spectral library. It shows how the AFSSI-C performs at TSNR = 0, -3, and -6 dB. Notice that at the first measurement step the DMD pattern set to all "on". This is because the first feature vector is simply attempting to discriminate the spectra based on the total intensity. Each figure is a reconstructed difference frame from a video. Due to the nature of books, showing video is impossible, the next best thing is to show several representative frames from the video.

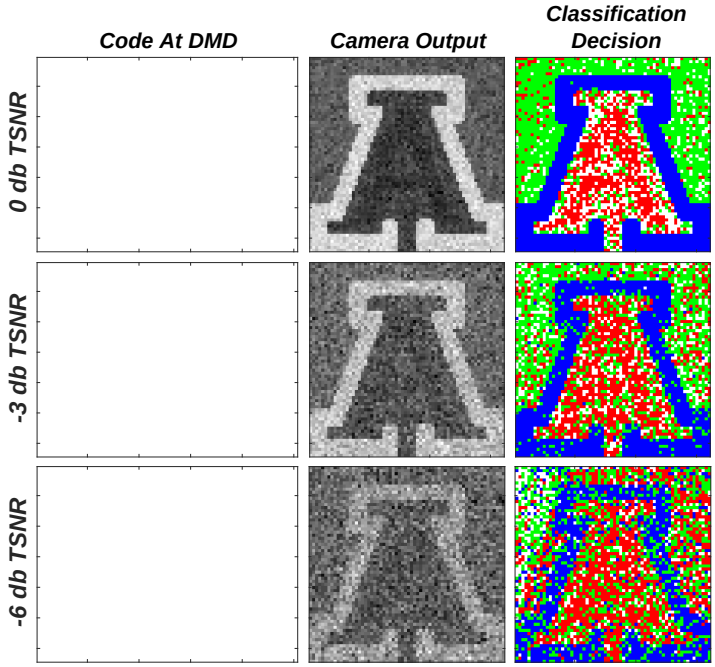

Figure F.1: Data from after the first measurement step at 0, -3, and -6 dB TSNR. The left column is a depiction of the DMD code, center spatially calibrated is the output in system pixels from the camera, and the right is the classification decision at the current measurement.

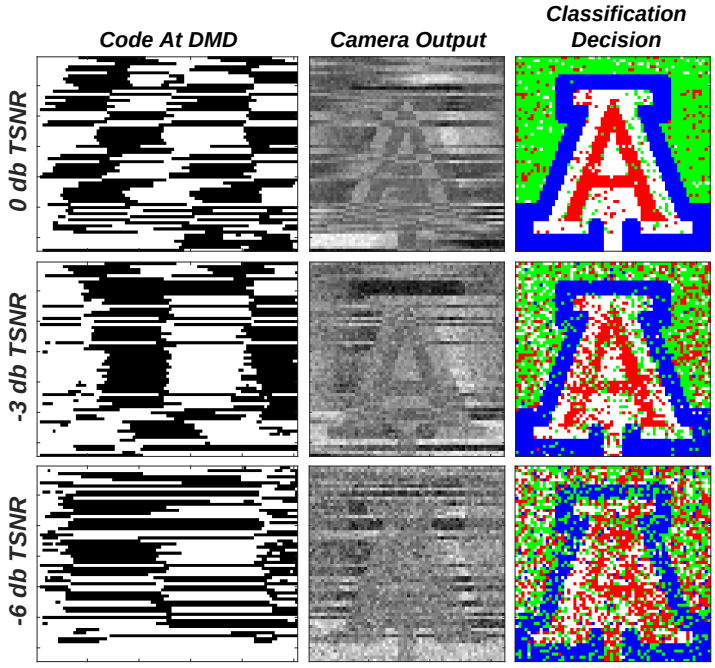

Figure F.2: Data from after the second measurement.

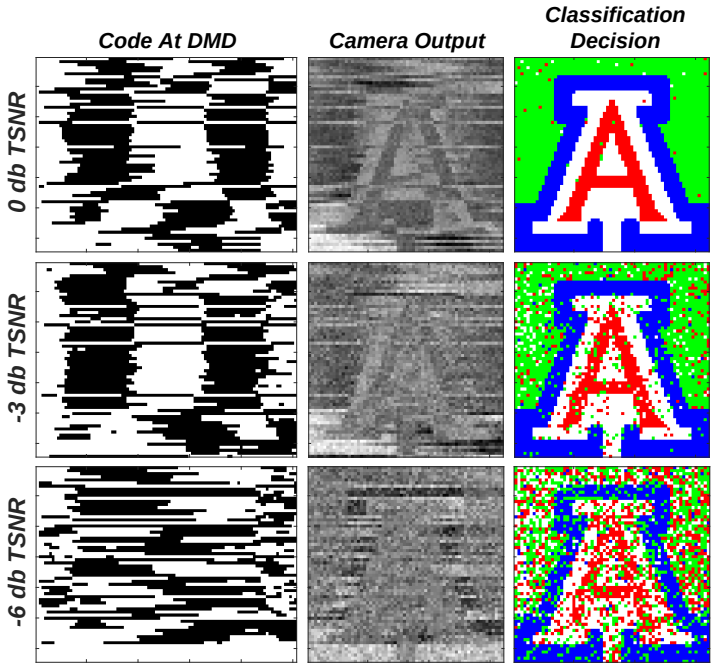

Figure F.3: Data from after the third measurement step.

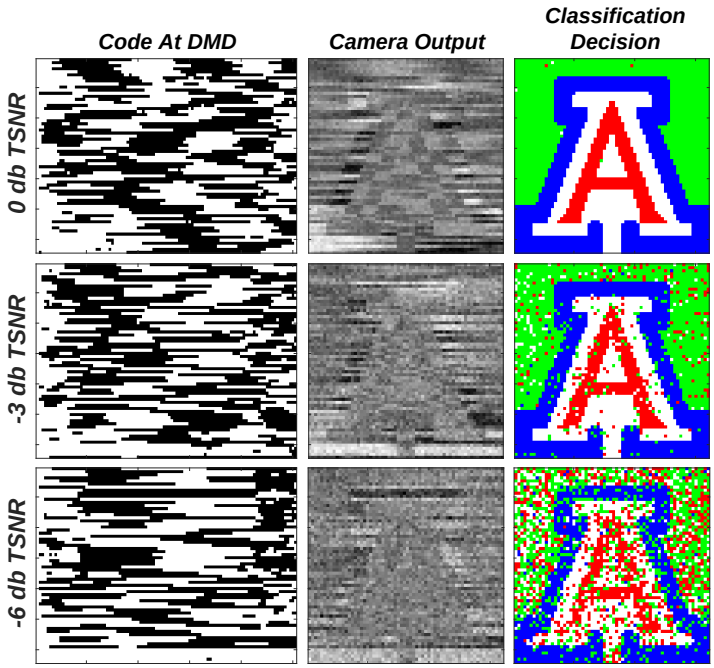

Figure F.4: Data from after the fourth measurement step.

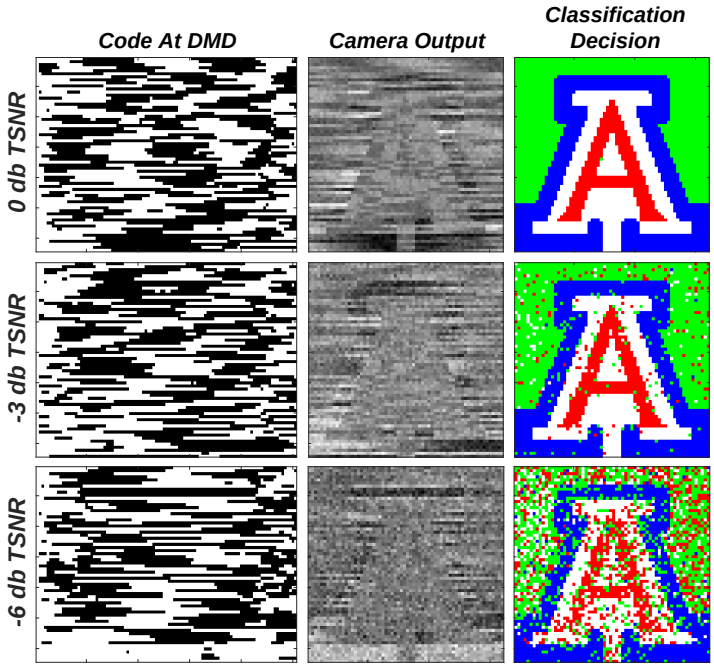

Figure F.5: Data from after the fifth measurement step.

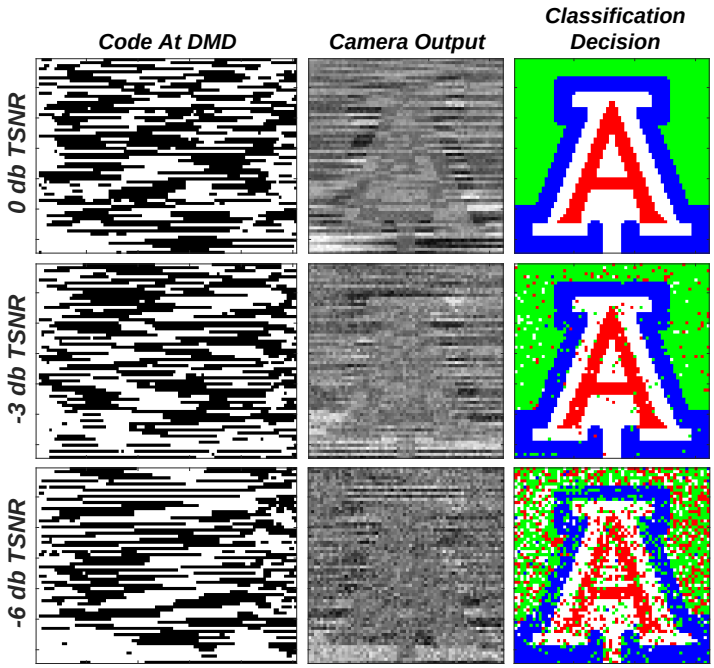

Figure F.6: Data from after the sixth measurement step.

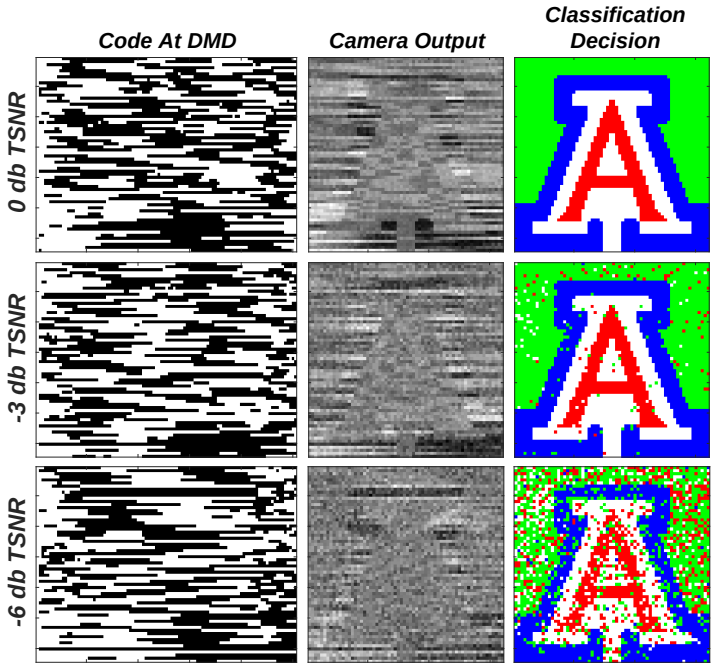

Figure F.7: Data from after the seventh measurement step.

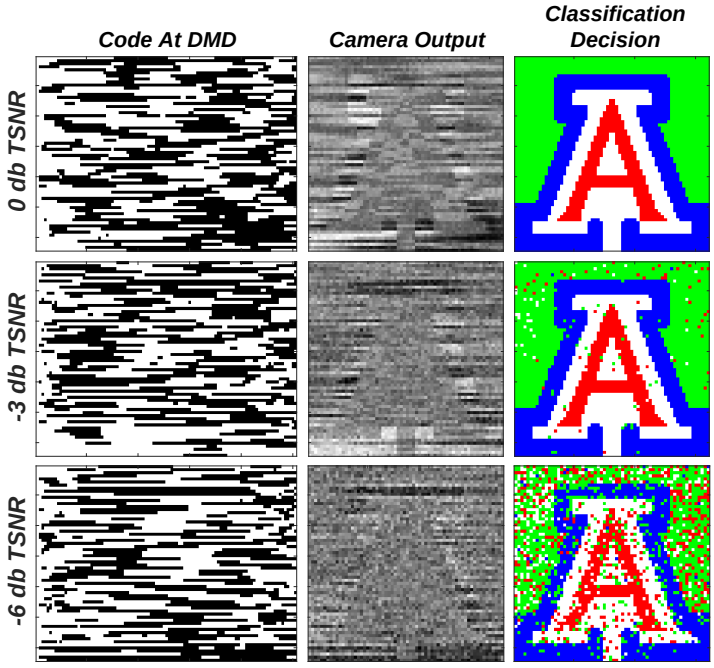

Figure F.8: Data from after the eighth measurement step.

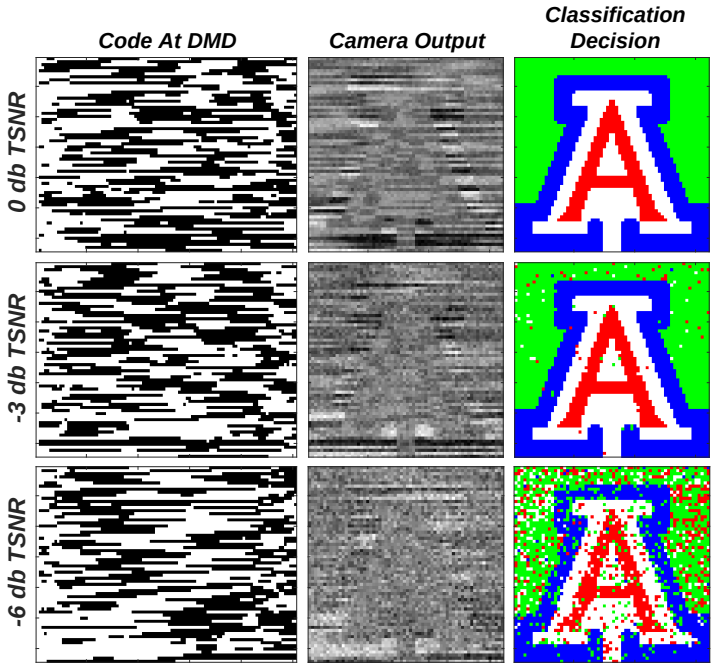

Figure F.9: Data from after the nineth measurement step.

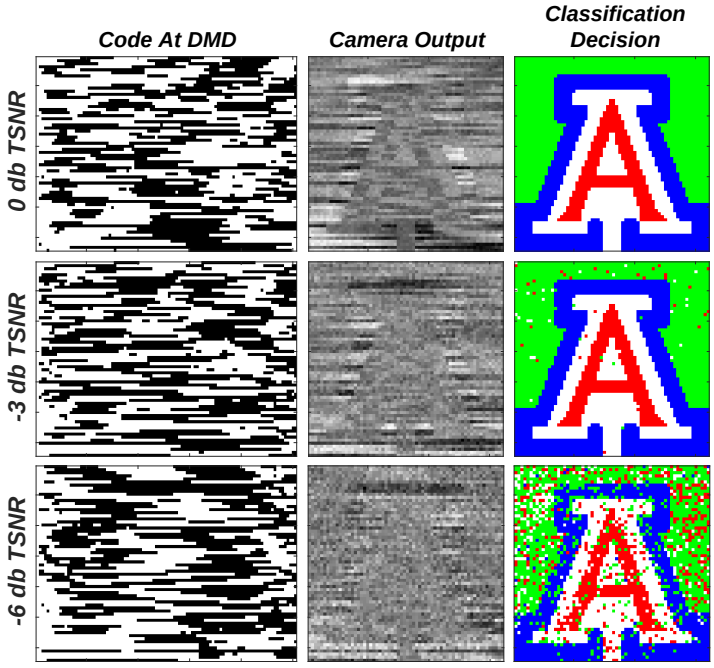

Figure F.10: Data from after the tenth measurement step.

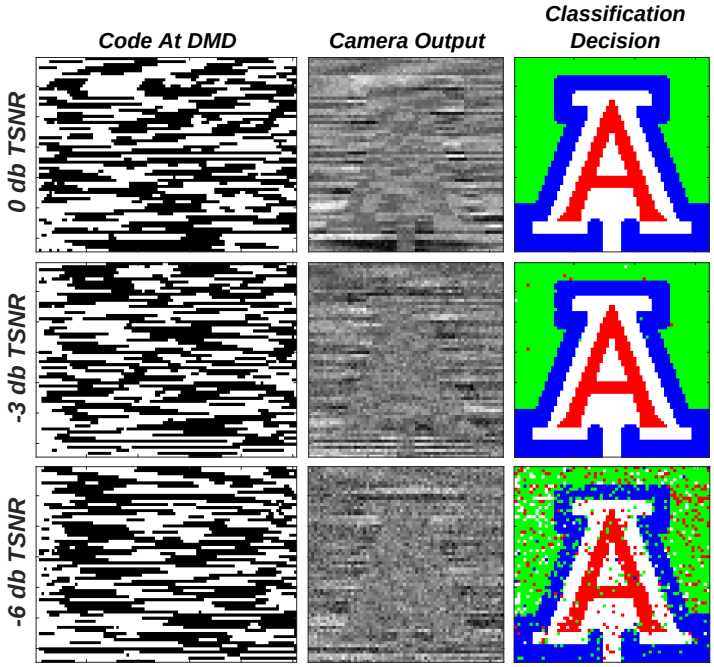

Figure F.11: Data from after the 15^{th} measurement step.

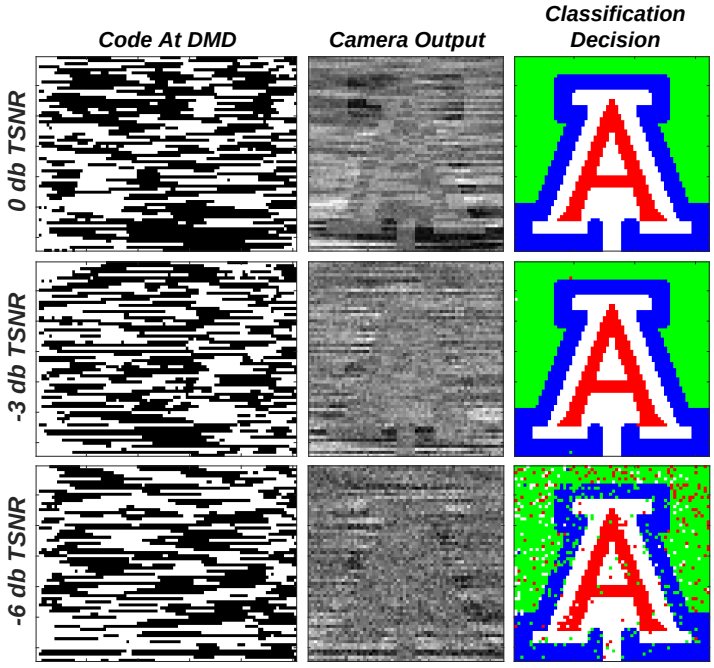

Figure F.12: Data from after the 20^{th} measurement step.

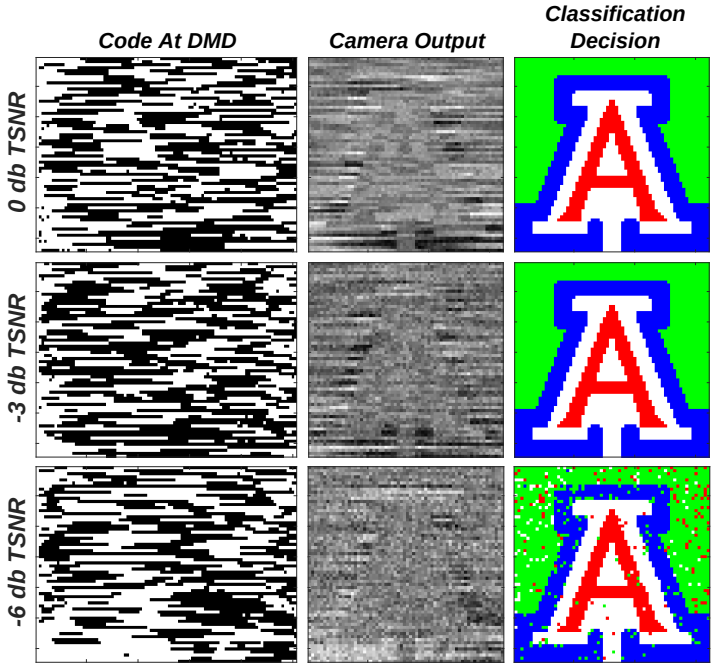

Figure F.13: Data from after the 25^{th} measurement step.

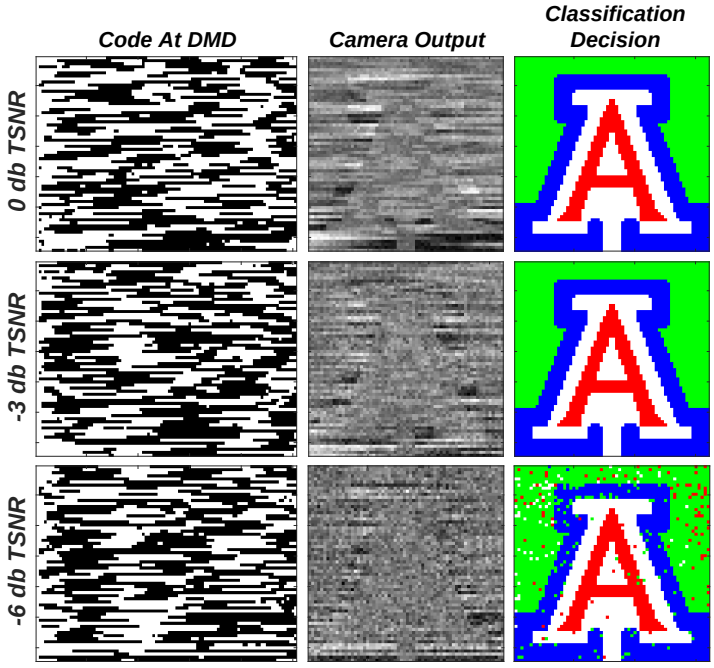

Figure F.14: Data from after the 30^{th} measurement step.

Appendix G

The Psuedo-Code For Single Pixel Adaptive Spectral Unmixing Using the AFSSI-C

Listing G.1: Wrapper Script For Launching the Simulations

```
1
2  function [] = launchUnmixingSim()
3
4  filterType = 'switchPCA'
5
6  snr = 2
7
8  % number of noise iterations
9  numNoiseItr = 200
10
11 load('aList6Endmembers.mat','aList')
12
13 load('monAndOLED_6Endmembers_1.mat','S')
14
15 %T = S(:,4:6)
16
```

```
%S = rand(40,100);

numSpecChan = 40


%T = circshift(T,[20 0])

%S = [S T];
S = imresize(S,[numSpecChan size(S,2)],'box');
figure(152);
plot(S)


rmseList = [];
rmseListLasso = [];

for lambda = 0.0175

    for nInd = 1:numNoiseItr

        a = aList(nInd,:);


        [rmseLs, rmseLasso, meas, H] = main(snr,a',S,lambda
           , filterType);

        rmseList = [rmseList;rmseLs];

        rmseListLasso = [rmseListLasso; rmseLasso];

    end

    rmseListAvg = mean(rmseList,1);
```

```
49      rmseListLassoAvg = mean(rmseListLasso,1);
50      rmseListLassoAvg = [NaN rmseListLassoAvg]
51
52      figure(352);
53      imagesc(H);
54
55      figure(353);
56      semilogy(rmseListAvg)
57      hold all
58      semilogy(rmseListLassoAvg);
59      hold off
60      legend('LS','LASSO')
61      grid on
62      title(['lambda ' num2str(lambda) ''])
63      drawnow
64
65      save(['lambda_' num2str(lambda) '_.mat'],...
66          'rmseListAvg','rmseListLassoAvg','S','aList','
            lambda')
67
68  end
```

Listing G.2: Main Simulation Script

```
1  % GENERATING SYSTEM MEASUREMENT MATRIX 'H'
2
3  function [rmseLs, rmseLasso, meas, H] = main(snr, a, S,
       lambda, filterType)
4
5  % The number of endmembers in the spectral library
6  numSpecCand = size(S,2);
7
8  % The number of spectral channels in the spectra
9  numSpecChan = size(S,1);
```

```
10
11 % The number of measurement steps
12 numMeas = 40;
13
14 pcNum = 1;
15
16 % Compute the average variance of the spectral library
17 avgVar = mean(var(S,1));
18
19 noiseStd = sqrt(avgVar/(10^snr));
20
21 %% Ground truth mixed spectrum
22
23 x = S*a;
24
25 %%
26 %
27 H = [];
28 g = [];
29 rmseLs = [];
30 rmseLasso = [];
31 meas = [];
32
33 % Initialize the fractional abundance
34 aEst = (1/numSpecCand) * ones(numSpecCand, 1);
35
36 % Begin the measurement loop
37 for mInd = 1:numMeas
38
39     if mInd == 2
40         aEst = abls;
41     elseif mInd > 2
42         aEst = ablasso;
```

```
43      end
44
45      % Compute the estimated mixed spectrum
46      xEst = S*aEst;
47
48      switch filterType
49
50          case 'switchPCA'
51              weightedS = repmat(aEst' .* aEst', numSpecChan,
                    1) .* S;
52
53              % Covariance matrix
54              X = weightedS * weightedS';
55
56              [V,D] = eig(X);
57
58              V = fliplr(V)';
59
60              %pcNum
61              h = V(pcNum,:);
62
63          case 'random'
64
65              h = randn(1,numSpecChan);
66      end
67
68
69      h = double(h>0);
70
71      h(h==0)=-1;
72
73      % Build the measurement matrix
74      H = [H; h];
```

```
75
76      A = H*S;
77
78      % Simulate Noisey Measurement
79      g_meas = h*x + noiseStd*rand(1,1);
80
81      % Build the measurement vector
82      g = [g; g_meas];
83
84      % The Estimated Measurement
85      g_est = h*S*aEst;
86
87      % Concatenate
88      meas = [meas norm(g_meas - g_est)];
89
90      % Check if we should
91      if mInd > 2
92
93          if meas(mInd) > meas(mInd - 1) - noiseStd/2
94
95              pcNum = pcNum + 1;
96
97              if pcNum == 6
98                  pcNum = 1;
99              end
100         else
101             pcNum = 1;
102         end
103     end
104
105     abls = pinv(A'*A)*A'*g;
106
107     if mInd > 1
```

```
108         ablasso = lasso(A,g,'lambda',lambda);
109     end
110
111     rmseLs = [rmseLs, mse_func(abls,a)];
112
113     if mInd > 1
114         rmseLasso = [rmseLasso, mse_func(ablasso,a)];
115     end
116
117
118 end
119
120 % end function
121 end
```

Glossary

bandlimited signal A bandlimited signal is any signal $g(x)$ that whose Fourier transform $G(f_x)$ is zero and remains zero past a certain frequency $|f_x| \geq B$.

coding In the context of computational sensing, coding is the process of modifying or modulating an analog signal during measurement. Coding is often used to reduce degeneracy in the measurement data. In the context of spectroscopy, the spectral filters act to code the spectrum.

compressible Signals with strictly sparse representation vectors are unlikely. Fortunately, it is possible to have approximately sparse representation vectors, which are called compressible. In other words, the sorted magnitudes of the coefficients $|x_n|$ quickly decay.

compressive imaging compressive sensing applied to imaging. Typically the goal is to reconstruct the entire object scene. However some compressive imaging sensors are have a task-specific sensing approach, such as the SCOUT.

compressive sampling *See* compressive sensing

compressive sensing A sensing technique that attempts to directly a compressive or sparse representation of the signal-of-interest during the measurement. Compressive sensors attempt to uses significantly less measurements than the dimensionality of the signal-of-interest. Compressive sensing relies on nonlinear optimization algorithms to perform reconstruction or task-specific sensing from highly underdetermined inverse problems. These algorithms often rely on sparsity to avoid overfitting.

computational sensing Any sensing technique in which the sensor uses *indirect-imaging*, *multiplex sensing*, compressive sensing, or *task-specific sensing*.

computational sensor *See* computational sensing

data processing inequality An theorem from information theory that proves the information of a signal cannot be increased via a local physical operation.

dimensionality reduction The process of reducing the dimensionality of the data in the scene. This step is optional and is only invoked by some algorithms to reduce the computional load of subsequent steps.

endmember The constituent spectra in a mixed spectrum is called the endmember

endmember determination The process of reducing the which endmembers are present in a mixed spectrum.

Fellgett advantage *See multiplex advantage*

forward model A numerical model, typically an equation, that explains the mapping of the analog signal-of-interest to the measurement data.

fractional abundance The relative amount of a spectral endmember in a mixed spectrum

indirect-imaging An imaging sensor that attempts to reconstruct an image of an object using non-isomorphic measurements. A computational step is required to reconstruct the object signal-of-interest. X-Ray CT and SAR are examples types of indirect-imaging.

inverse problem The problem of taking the measurement data and calculating a reconstruction of the signal-of-interst or task-specific parameters. In computational sensing, computer algorithms are used to solve inverse problems.

inversion The of solving an inverse problem.

isomorphic *See* isomorphic sensing

isomorphic sensing An isomorphic sensor is any sensor that attempts to produce measurement data that resembles the signal-of-interest. An isomorphic measurement is a measurement that resembles the signal-of-interest. An isomorphic measurement can be described as a one-to-one mapping from the signal-of-interest to the measurement, and is represented in matrix notation with an identity matrix. Isomorphic sensing is synonomous with traditional sensing.

Jacquinot advantage This results from the fact that in a dispersive instrument, the traditional spectrometer has entrance and exit slits which restrict the amount of light that passes through it. By removing the slits a spectrometer produces a higher signal-to-noise ratio.

lasso The least absolute shrinkage and selection operator (lasso) is a regression analysis method that performs both variable selection and regularization. It is refers to an optimization problem, a regression technique, and an algorithm. The lasso problem is the

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}} \ \|\mathbf{A}\mathbf{x} = \mathbf{g}\|_2^2 + \tau\|\mathbf{x}\|_1$$

likelihood The likelihood is the probability of the data g assuming that a hypothesis h or parameter θ is true. $P\left(g \,\middle|\, \theta\right)$

measurement A process that converts a physical phenomena to single datum or a set of data. In the context of this dissertation it used synonomously with the detector readout.

mixed spectrum A mixed spectrum is a measured spectrum that contains spectra from more than one spectrum.

monochromator An optical instrument that transmits a selectable narrow wavelength band of light chosen from a wider range of wavelengths available at the input.

multiplex advantage The improvement in SNR that is due to multiplexed measurements rather than isomorphic measurements. This is often referred to as the Fellgett advantage since it was first discovered by Peter Fellgett. *See* multiplex sensing

multiplex sensing A multiplexing sensor is any sensor that attempts to combine the physical phenomena of the analog signal-of-interest in to a few or one analog-to-digital sample to overcome limits due to signal-to-noise ratio. The measurement data that does resemble the signal-of-interest. A matrix representation of a multiplex measurement will not look like an identity matrix

multiplexing *See* multiplex sensing

pixel pitch The center to center distance between pixels on a focal-plane array such as a CCD or CMOS image sensor.

posterior The posterior probability is the conditional probability of a hypothesis h or parameter θ given some data g. $P\left(\theta \mid g\right)$

prior The priori probability is the probability of a hypothesis h or parameter θ without any knowledge of the data data g. $P\left(\theta\right)$

pushbroom Pushbroom scanning is an isomorphic measurement technique for acquiring a spectral datacube. In the pushbroom technique, the entire spectrum for an entire spatial row or column is acquired one at a time.

ridge regression Ridge regression is a regression technique that uses ℓ_2 regularization to prevent overfitting of the data. Ridge regression seeks coefficient estimates that fit the data well by making the objective function, the ℓ_2 norm between the data and the linear model small, however the shrinkage penalty has the effect of shrinking estimates towards zero.

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}} \ \|\mathbf{A}\mathbf{x} = \mathbf{g}\|_2^2 + \tau\|\mathbf{x}\|_2$$

sample The process of mapping a continuous signal to a discrete signal.

sampling *See sample*

sparse *See* sparsity

sparsity A set, vector, or matrix which contains an overwhelming majority of zeros relative to the size of the set, vector, or matrix.

spectral resolution The smallest the smallest difference in wavelength the instrument or sensor can discern.

spectral unmixing The procedure by which the measured spectrum of a mixed pixel is decomposed into a collection of constituent spectra, or endmembers, and a set of corresponding fractional abundances.

task-specific sensing A sensor that does not attempt to reconstruct the signal-of-interest to perform a signal processing task such as detection, estimation, and classification. The AFSSI-C is an example of a task-specific sensor.

tunable filter Pushbroom scanning is an isomorphic measurement technique for acquiring a spectral datacube. In the pushbroom technique, the entire spectrum for an entire spatial row or column is acquired one at a time.

whiskbroom Whiskbroom scanning is an isomorphic measurement technique for acquiring a spectral datacube. In the whiskbroom technique, the entire spectrum at each spatial location is acquired one at a time.

Acronyms

ADC analog-to-digital converter

AFSS Adaptive Feature Specific Spectrometer

AFSSI-C Adaptive Feature Specific Spectral Imaging-Classifier

AVIRIS Airborne Visible/Infrared Imaging Spectrometer

AWGN Additive white Gaussian noise

CACTI Coded Aperture Compressive Temporal Imaging

CASSI Coded Aperture Snapshot Spectral Imaging

CCD Charge-Coupled Device

CMOS Complementary Metal–Oxide–Semiconductor

CT Computed Tomography

CTIS Computed Tomography Imaging Spectrometer

DISP Duke Imaging and Spectroscopy Program

DMD Digital Micro-Mirror Display

FOV field-of-view

FPA focal-plane array

FTS Fourier Transform Spectrometer

LAR Least Angle Regression

LCOS Liquid Crystal on Silicon

LCSI LCOS Computational Spectral Imager

LENS Laboratory for Engineering Non-Traditional Sensors

LMM Linear Mixing Model

LS least squares

MAP Maximum A Posteriori

MLE Maximum Likelihood Estimation

MNF Maximum Noise Fraction

MRI Magnetic Resonance Imaging

MSE Mean Squared Error

OSA Optical Society of America

PBS Polarizing Beam Splitter

PCA Principal Component Analysis

PET Positron Emission Tomography

pPCA probabilistically weighted Principal Component Analysis

PSF point-spread function

RIP restricted isometry property

RMSE Root Mean Squared Error

ROI Region of Interest

SAR Synthetic Aperture Radar

SCOUT Static Computational Optical Undersampled Tracker

SLM Spatial Light Modulator

SNR signal-to-noise ratio

SP system pixel

SPECT Single-Photon Emission Computed Tomography

SWAP-C size, weight and power-cost

SWPCA Switch Weighted Principal Component Analysis

TEC thermoelectric cooler

TSNR Task Signal-To-Noise Ratio

Symbols

$\mathbf{A}$ The product of the sensing matrix and the representation matrix $\mathbf{A} = \mathbf{H\Psi}$

c Spectral channel index

D Notation for the spectral datacube.

δ_λ Spectral resolution

$\mathbf{e}$ Additive noise vector

$\mathbf{f}$ Object signal-of-interest

$\hat{\mathbf{f}}$ Estimated object

$\mathbf{g}$ Measurement vector

$\mathbf{H}$ The system matrix which captures all of the physical phenomena in a measurement. Also called the measurement matrix and sensing matrix.

$\mathbf{H}_N$ A Hadamard matrix of size $N \times N$

K Notation for sparsity which is defined as the number of non-zero entries in a vector.

N_λ Number of spectral channels

N_m Total number of measurements

N_R Number of spectra in the spectral library

τ Notation for a regularization parameter in an objective function for any kind of optimization

Bibliography

[1] D. J. Brady, *Optical imaging and spectroscopy.* John Wiley & Sons, 2009.

[2] M. A. Neifeld, A. Mahalanobis, and D. J. Brady, "Task-specific sensing-introduction," *Appl. Opt.*, vol. 45, no. 13, pp. 2857–2858, May 2006. [Online]. Available: http://ao.osa.org/abstract.cfm?URI=ao-45-13-2857

[3] D. Dinakarababu, D. Golish, and M. Gehm, "Adaptive feature specific spectroscopy for rapid chemical identification," *Optics express*, vol. 19, no. 5, pp. 4595–4610, 2011.

[4] T. M. Cover and J. A. Thomas, *Elements of information theory.* John Wiley & Sons, 2012.

[5] C. M. Watts, D. Shrekenhamer, J. Montoya, G. Lipworth, J. Hunt, T. Sleasman, S. Krishna, D. R. Smith, and W. J. Padilla, "Terahertz compressive imaging with metamaterial spatial light modulators," *Nature Photonics*, vol. 8, no. 8, pp. 605–609, 2014.

[6] I. Noor, O. Furxhi, and E. L. Jacobs, "Compressive sensing for a submillimeter-wave single pixel imager," in *SPIE Defense, Security, and Sensing.* International Society for Optics and Photonics, 2011, pp. 80 220K–80 220K.

[7] D. Taubman and M. Marcellin, *JPEG2000 Image Compression Fundamentals, Standards and Practice: Image Compression Fundamentals, Standards and Practice.* Springer Science & Business Media, 2012, vol. 642.

[8] M. J. Golay, "Multi-slit spectrometry," *JOSA*, vol. 39, no. 6, pp. 437–444, 1949.

[9] E. E. Fenimore and T. Cannon, "Coded aperture imaging with uniformly redundant arrays," *Applied optics*, vol. 17, no. 3, pp. 337–347, 1978.

[10] S. R. Gottesman and E. Fenimore, "New family of binary arrays for coded aperture imaging," *Applied optics*, vol. 28, no. 20, pp. 4344–4352, 1989.

[11] M. F. Duarte, M. A. Davenport, D. Takhar, J. N. Laska, T. Sun, K. E. Kelly, R. G. Baraniuk *et al.*, "Single-pixel imaging via compressive sampling," *IEEE Signal Processing Magazine*, vol. 25, no. 2, p. 83, 2008.

[12] D. Townsend, P. Poon, S. Wehrwein, T. Osman, A. Mariano, E. Vera, M. Stenner, and M. Gehm, "Static compressive tracking," *Optics express*, vol. 20, no. 19, pp. 21 160–21 172, 2012.

[13] M. E. Gehm, S. T. McCain, N. P. Pitsianis, D. J. Brady, P. Potuluri, and M. E. Sullivan, "Static two-dimensional aperture coding for multimodal, multiplex spectroscopy," *Applied optics*, vol. 45, no. 13, pp. 2965–2974, 2006.

[14] T.-H. Tsai and D. J. Brady, "Coded aperture snapshot spectral polarization imaging," *Applied optics*, vol. 52, no. 10, pp. 2153–2161, 2013.

[15] J. Holloway, A. C. Sankaranarayanan, A. Veeraraghavan, and S. Tambe, "Flutter shutter video camera for compressive sensing of videos," in *Computational Photography (ICCP), 2012 IEEE International Conference on.* IEEE, 2012, pp. 1–9.

[16] P. Llull, X. Liao, X. Yuan, J. Yang, D. Kittle, L. Carin, G. Sapiro, and D. J. Brady, "Coded aperture compressive temporal imaging," *Optics express*, vol. 21, no. 9, pp. 10 526–10 545, 2013.

[17] H. H. Barrett and K. J. Myers, *Foundations of Image Science.* John Wiley & Sons, 2013.

[18] J. Radon, "1.1 über die bestimmung von funktionen durch ihre integralwerte längs gewisser mannigfaltigkeiten," *Classic papers in modern diagnostic radiology*, vol. 5, 2005.

[19] "The Nobel Prize in Physiology or Medicine, 1979," https://www.nobelprize.org/nobel_prizes/medicine/laureates/1979/perspectives.html, accessed: 2016-08-22.

[20] X. X. Zhu and R. Bamler, "Tomographic sar inversion by-norm regularization—the compressive sensing approach," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 48, no. 10, pp. 3839–3846, 2010.

[21] C. Chen and J. Huang, "Compressive sensing mri with wavelet tree sparsity," in *Advances in neural information processing systems*, 2012, pp. 1115–1123.

[22] W. S. Boyle and G. E. Smith, "Charge coupled semiconductor devices," *Bell System Technical Journal*, vol. 49, no. 4, pp. 587–593, 1970.

[23] H. W, N. L, and S. Joe, "Image orthicon," Feb. 11 1975, uS Patent 3,866,078. [Online]. Available: https://www.google.com/patents/US3866078

[24] J. Estrom, "Kodak's First Digital Moment," http://lens.blogs.nytimes.com/2015/08/12/kodaks-first-digital-moment/, August 12 2015, accessed: 2016-08-24.

[25] K. L. Moore, "Spectrometer with electronic readout," Mar. 27 1979, uS Patent 4,146,332.

[26] H. Kobayashi and L. R. Bahl, "Image data compression by predictive coding i: Prediction algorithms," *IBM Journal of Research and Development*, vol. 18, no. 2, pp. 164–171, 1974.

[27] J. Ziv and A. Lempel, "Compression of individual sequences via variable-rate coding," *IEEE transactions on Information Theory*, vol. 24, no. 5, pp. 530–536, 1978.

[28] B. E. Usevitch, "A tutorial on modern lossy wavelet image compression: foundations of jpeg 2000," *IEEE signal processing magazine*, vol. 18, no. 5, pp. 22–35, 2001.

[29] E. J. Candès, J. Romberg, and T. Tao, "Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information," *IEEE Transactions on information theory*, vol. 52, no. 2, pp. 489–509, 2006.

[30] E. J. Candes and T. Tao, "Near-optimal signal recovery from random projections: Universal encoding strategies?" *IEEE transactions on information theory*, vol. 52, no. 12, pp. 5406–5425, 2006.

[31] D. L. Donoho, "Compressed sensing," *IEEE Transactions on information theory*, vol. 52, no. 4, pp. 1289–1306, 2006.

[32] A. Wagadarikar, R. John, R. Willett, and D. Brady, "Single disperser design for coded aperture snapshot spectral imaging," *Applied optics*, vol. 47, no. 10, pp. B44–B51, 2008.

[33] H. S. Pal, D. Ganotra, and M. A. Neifeld, "Face recognition by using feature-specific imaging," *Applied optics*, vol. 44, no. 18, pp. 3784–3794, 2005.

[34] A. Stern, I. Y. August, and Y. Oiknine, "Hurdles in the implementation of compressive sensing for imaging and ways to overcome them," in *SPIE Commercial+ Scientific Sensing and Imaging*. International Society for Optics and Photonics, 2016, pp. 987 006–987 006.

[35] "The Optical Society of America, Meeting of Computational Optical Sensing and Imaging (COSI), 2016," http://www.osa.org/en-us/meetings/osa_meetings/imaging_and_applied_optics/computational_optical_sensing_and_imaging/, accessed: 2016-09-04.

[36] S. Foucart and H. Rauhut, *A mathematical introduction to compressive sensing*. Springer, 2013, vol. 1, no. 3.

[37] M. Gehm, "Calibration–an open challenge in creating practical computational-and compressive-sensing systems," 2013.

[38] E. J. Candès and M. B. Wakin, "An introduction to compressive sampling," *IEEE signal processing magazine*, vol. 25, no. 2, pp. 21–30, 2008.

[39] J. A. Tropp, "Just relax: Convex programming methods for identifying sparse signals in noise," *IEEE transactions on information theory*, vol. 52, no. 3, pp. 1030–1051, 2006.

[40] M. Dunlop-Gray, P. K. Poon, D. Golish, E. Vera, and M. E. Gehm, "Experimental demonstration of an adaptive architecture for direct spectral imaging classification," *Optics Express*, vol. 24, no. 16, pp. 18 307–18 321, 2016.

[41] A. Ashok and M. A. Neifeld, "Compressive imaging: hybrid measurement basis design," *JOSA A*, vol. 28, no. 6, pp. 1041–1050, 2011.

[42] Y. Li, A. C. Sankaranarayanan, L. Xu, R. Baraniuk, and K. F. Kelly, "Realization of hybrid compressive imaging strategies," *JOSA A*, vol. 31, no. 8, pp. 1716–1720, 2014.

[43] M. Harwit and N. J. Sloane, *Hadamard transform optics*. Elsevier, 2012.

[44] J. W. Goodman, *Introduction to Fourier optics*. Roberts and Company Publishers, 2005.

[45] P. Fellgett, "I.—les principes généraux des méthodes nouvelles en spectroscopie interférentielle-a propos de la théorie du spectromètre interférentiel multiplex," *J. phys. radium*, vol. 19, no. 3, pp. 187–191, 1958.

[46] S. P. Davis, M. C. Abrams, and J. W. Brault, *Fourier transform spectrometry*. Academic Press, 2001.

[47] J. W. Goodman, *Statistical optics*. John Wiley & Sons, 2015.

[48] M. H. Tai and M. Harwit, "Fourier and hadamard transform spectrometers: a limited comparison," *Applied optics*, vol. 15, no. 11, pp. 2664–2666, 1976.

[49] I. Jolliffe, *Principal component analysis*. Wiley Online Library, 2002.

[50] J. Shlens, "A tutorial on principal component analysis," *arXiv preprint arXiv:1404.1100*, 2014.

[51] P. K. Poon, W.-R. Ng, and V. Sridharan, "Image denoising with singular value decompositon and principal component analysis," December 2009.

[52] C. E. Shannon, "Communication in the presence of noise," *Proceedings of the IRE*, vol. 37, no. 1, pp. 10–21, 1949.

[53] J. G. Proakis and D. G. Manolakis, *Introduction to digital signal processing*. Prentice Hall Professional Technical Reference, 1988.

[54] E. Candes and J. Romberg, "l1-magic: Recovery of sparse signals via convex programming," *URL: www. acm. caltech. edu/l1magic/downloads/l1magic. pdf*, vol. 4, p. 14, 2005.

[55] Y. Oiknine, I. August, and A. Stern, "Along-track scanning using a liquid crystal compressive hyperspectral imager," *Optics express*, vol. 24, no. 8, pp. 8446–8457, 2016.

[56] X. Yuan, T.-H. Tsai, R. Zhu, P. Llull, D. Brady, and L. Carin, "Compressive hyperspectral imaging with side information," *IEEE Journal of Selected Topics in Signal Processing*, vol. 9, no. 6, pp. 964–976, 2015.

[57] C. C. Aggarwal, *Data streams: models and algorithms*. Springer Science & Business Media, 2007, vol. 31.

[58] S. S. Chen, D. L. Donoho, and M. A. Saunders, "Atomic decomposition by basis pursuit," *SIAM review*, vol. 43, no. 1, pp. 129–159, 2001.

[59] B. Efron, T. Hastie, I. Johnstone, R. Tibshirani *et al.*, "Least angle regression," *The Annals of statistics*, vol. 32, no. 2, pp. 407–499, 2004.

[60] S.-J. Kim, K. Koh, M. Lustig, S. Boyd, and D. Gorinevsky, "An interior-point method for large-scale-regularized least squares," *IEEE journal of selected topics in signal processing*, vol. 1, no. 4, pp. 606–617, 2007.

[61] A. Neumaier, "Solving ill-conditioned and singular linear systems: A tutorial on regularization," *SIAM review*, vol. 40, no. 3, pp. 636–666, 1998.

[62] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 267–288, 1996.

[63] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An introduction to statistical learning*. Springer, 2013, vol. 6.

[64] M. D. Stenner, D. J. Townsend, and M. E. Gehm, "Static architecture for compressive motion detection in persistent, pervasive surveillance applications," in *Imaging Systems*. Optical Society of America, 2010, p. IMB2.

[65] S. Evladov, O. Levi, and A. Stern, "Progressive compressive imaging from radon projections," *Optics express*, vol. 20, no. 4, pp. 4260–4271, 2012.

[66] P. Poon, E. Vera, and M. E. Gehm, "Advances in the design, calibration and use of a static coded aperture compressive tracking and imaging system," in *Computational Optical Sensing and Imaging*. Optical Society of America, 2012, pp. CTu3B–2.

[67] Y. Kashter, O. Levi, and A. Stern, "Optical compressive change and motion detection," *Applied optics*, vol. 51, no. 13, pp. 2491–2496, 2012.

[68] Y. Rivenson, A. Stern, and B. Javidi, "Single exposure super-resolution compressive imaging by double phase encoding," *Optics Express*, vol. 18, no. 14, pp. 15 094–15 103, 2010.

[69] W. U. Bajwa, J. D. Haupt, G. M. Raz, S. J. Wright, and R. D. Nowak, "Toeplitz-structured compressed sensing matrices," in *2007 IEEE/SP 14th Workshop on Statistical Signal Processing*. IEEE, 2007, pp. 294–298.

[70] H. Rauhut, "Circulant and toeplitz matrices in compressed sensing," *arXiv preprint arXiv:0902.4394*, 2009.

[71] J. Romberg, "Compressive sensing by random convolution," *SIAM Journal on Imaging Sciences*, vol. 2, no. 4, pp. 1098–1128, 2009.

[72] F. Sebert, Y. M. Zou, and L. Ying, "Toeplitz block matrices in compressed sensing and their applications in imaging," in *2008 International Conference on Information Technology and Applications in Biomedicine*. IEEE, 2008, pp. 47–50.

[73] B. Liu, F. Sebert, Y. Zou, and L. Ying, "Sparsesense: randomly-sampled parallel imaging using compressed sensing," in *In: Proceedings of the 16th Annual Meeting of ISMRM*. Citeseer, 2008.

[74] C.-I. Chang, *Hyperspectral imaging: techniques for spectral detection and classification*. Springer Science & Business Media, 2003, vol. 1.

[75] A. Ibrahim, S. Tominaga, and T. Horiuchi, "Spectral imaging method for material classification and inspection of printed circuit boards," *Optical Engineering*, vol. 49, no. 5, pp. 057 201–057 201, 2010.

[76] G. A. Shaw and H.-H. K. Burke, "Spectral imaging for remote sensing," *Lincoln Laboratory Journal*, vol. 14, no. 1, pp. 3–28, 2003.

[77] Y. Garini, I. T. Young, and G. McNamara, "Spectral imaging: principles and applications," *Cytometry Part A*, vol. 69, no. 8, pp. 735–747, 2006.

[78] M. T. Eismann, *Hyperspectral remote sensing*. SPIE Bellingham, 2012.

[79] W. L. Wolfe, *Introduction to imaging spectrometers*. SPIE Press, 1997, vol. 25.

[80] C. Yang, J. H. Everitt, M. R. Davis, and C. Mao, "A ccd camera-based hyperspectral imaging system for stationary and airborne applications," *Geocarto International*, vol. 18, no. 2, pp. 71–80, 2003.

[81] C. Fabry and A. Pérot, "Sur les franges des lames minces argentées et leur application à la mesure de petites épaisseurs d'air [on the fringes of thin layers of silver and their application to the measurement of small thicknesses of air]," *Ann. Chim. Phys*, vol. 12, pp. 459–501, 1897.

[82] A. Perot and C. Fabry, "On the application of interference phenomena to the solution of various problems of spectroscopy and metrology," *The Astrophysical Journal*, vol. 9, p. 87, 1899.

[83] C. Fabry and A. Perot, "On a new form of interferometer," *The Astrophysical Journal*, vol. 13, p. 265, 1901.

[84] N. Gat, "Imaging spectroscopy using tunable filters: a review," in *AeroSense 2000*. International Society for Optics and Photonics, 2000, pp. 50–64.

[85] R. O. Green, M. L. Eastwood, C. M. Sarture, T. G. Chrien, M. Aronsson, B. J. Chippendale, J. A. Faust, B. E. Pavri, C. J. Chovit, M. Solis *et al.*, "Imaging spectroscopy and the airborne visible/infrared imaging spectrometer (aviris)," *Remote Sensing of Environment*, vol. 65, no. 3, pp. 227–248, 1998.

[86] M. Descour and E. Dereniak, "Computed-tomography imaging spectrometer: experimental calibration and reconstruction results," *Applied Optics*, vol. 34, no. 22, pp. 4817–4826, 1995.

[87] M. K. Steven, "Fundamentals of statistical signal processing," *PTR Prentice-Hall, Englewood Cliffs, NJ*, 1993.

[88] T. K. Moon, "The expectation-maximization algorithm," *IEEE Signal processing magazine*, vol. 13, no. 6, pp. 47–60, 1996.

[89] G. R. Arce, D. J. Brady, L. Carin, H. Arguello, and D. S. Kittle, "Compressive coded aperture spectral imaging: An introduction," *IEEE Signal Processing Magazine*, vol. 31, no. 1, pp. 105–115, 2014.

[90] A. A. Wagadarikar, N. P. Pitsianis, X. Sun, and D. J. Brady, "Spectral image estimation for coded aperture snapshot spectral imagers," in *Optical Engineering+ Applications*. International Society for Optics and Photonics, 2008, pp. 707 602–707 602.

[91] J. M. Bioucas-Dias and M. A. Figueiredo, "A new twist: two-step iterative shrinkage/thresholding algorithms for image restoration," *IEEE Transactions on Image processing*, vol. 16, no. 12, pp. 2992–3004, 2007.

[92] N. Hagen and M. W. Kudenov, "Review of snapshot spectral imaging technologies," *Optical Engineering*, vol. 52, no. 9, pp. 090 901–090 901, 2013.

[93] E. M. DuPont, D. Chambers, J. Alexander, and K. Alley, "A spatial-spectral classification approach of multispectral data for ground perspective materials," in *Systems, Man, and Cybernetics (SMC), 2011 IEEE International Conference on*. IEEE, 2011, pp. 3125–3129.

[94] C. Liu and J. Gu, "Discriminative illumination: Per-pixel classification of raw materials based on optimal projections of spectral brdf," *IEEE transactions on pattern analysis and machine intelligence*, vol. 36, no. 1, pp. 86–98, 2014.

[95] M. J. Dunlop-Gray, "Seeing beyond sight: The adaptive, feature-specific, spectral imaging classifier," Ph.D. dissertation, The University of Arizona, 2015.

[96] W. J. Smith, *Modern optical engineering*. Tata McGraw-Hill Education, 1966.

[97] M. A. Folkman, J. Pearlman, L. B. Liao, and P. J. Jarecke, "Eo-1/hyperion hyperspectral imager design, development, characterization, and calibration," in *Second International Asia-Pacific Symposium on Remote Sensing of the Atmosphere, Environment, and Space*. International Society for Optics and Photonics, 2001, pp. 40–51.

[98] N. Keshava and J. F. Mustard, "Spectral unmixing," *IEEE signal processing magazine*, vol. 19, no. 1, pp. 44–57, 2002.

[99] N. Keshava, "A survey of spectral unmixing algorithms," *Lincoln Laboratory Journal*, vol. 14, no. 1, pp. 55–78, 2003.

[100] R. N. Clark and T. L. Roush, "Reflectance spectroscopy: Quantitative analysis techniques for remote sensing applications," *Journal of Geophysical Research: Solid Earth*, vol. 89, no. B7, pp. 6329–6340, 1984.

[101] A. A. Green, M. Berman, P. Switzer, and M. D. 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.

[102] W. Philpot, "Digital image processing," 2015.

[103] C. L. Lawson and R. J. Hanson, *Solving least squares problems*. SIAM, 1995, vol. 15.

[104] G. Lazarev, A. Hermerschmidt, S. Krüger, and S. Osten, "Lcos spatial light modulators: trends and applications," pp. 1–30, 2012.

[105] T.-H. Tsai, X. Yuan, and D. J. Brady, "Spatial light modulator based color polarization imaging," *Optics express*, vol. 23, no. 9, pp. 11 912–11 926, 2015.

[106] C. Li, T. Sun, K. F. Kelly, and Y. Zhang, "A compressive sensing and unmixing scheme for hyperspectral data processing," *IEEE Transactions on Image Processing*, vol. 21, no. 3, pp. 1200–1210, 2012.

[107] J. Monsalve, H. Vargas, and H. Arguello, "Spectral raman unmixing from cassi system compressive measurements," in *Signal Processing, Images and Computer Vision (STSIVA), 2015 20th Symposium on*. IEEE, 2015, pp. 1–6.

[108] Y. August and A. Stern, "Compressive sensing spectrometry based on liquid crystal devices," *Optics letters*, vol. 38, no. 23, pp. 4996–4999, 2013.

[109] R. Zhu, T.-H. Tsai, and D. J. Brady, "Coded aperture snapshot spectral imager based on liquid crystal spatial light modulator," in *Frontiers in Optics*. Optical Society of America, 2013, pp. FW1D–4.

[110] I. August, Y. Oiknine, M. AbuLeil, I. Abdulhalim, and A. Stern, "Miniature compressive ultra-spectral imaging system utilizing a single liquid crystal phase retarder," *Scientific reports*, vol. 6, 2016.

[111] D. Hundley, M. Kirby, and M. Anderle, "A solution procedure for blind signal separation using the maximum noise fraction approach: algorithms and examples," in *Proceedings of the Conference on Independent Component Analysis, San Diego, CA*, 2001, pp. 337–342.

[112] "Rice's spectral eyes bound for the skies," http://news.rice.edu/2016/11/14/rices-spectral-eyes-bound-for-the-skies-2/, accessed: 2016-11-22.