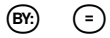
PRACTICAL CONSIDERATIONS IN EXPERIMENTAL COMPUTATIONAL SENSING

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.

Amit Ashok — Date: 9 December 2016

Rongguang Liang — Date: 9 December 2016

Michael E. Gehm — Date: 9 December 2016

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.

Dissertation Director: Amit Ashok — Date: 9 December 2016

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 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 methods of analysis. Professor Vera is directly responsible for much of my training as an experimentalist. I consider Professor Vera as an older brother who was always there to protect me from the pitfalls of the graduate school journey.

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 Interference 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 would 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 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

List of Figures . . . 9

List of Tables . . . 11

ABSTRACT . . . 12

Chapter 1 Introduction . . . 13
1.1 Isomorphic Sensing . . . 15
1.2 Development of Multiplexing in Sensing . . . 19
1.3 Forward Models and Inverse Problems . . . 20
1.4 Indirect Imaging . . . 21
1.5 The Digital Imaging Revolution . . . 23
1.6 Compressive Sensing . . . 24
1.7 Practical Considerations in Computational Sensing . . . 27
1.8 Dissertation Overview . . . 29

Chapter 2 Formalism . . . 31
2.1 Isomorphic Sensing . . . 32
2.2 Multiplexing . . . 32
2.2.1 Coding Schemes . . . 33
2.2.2 The Fellgett Advantage . . . 34
2.3 Principal Component Analysis . . . 36
2.4 Bayesian Statistics . . . 38
2.4.1 Example: Updating Probabilities with Bayes' Theorem . . . 39
2.4.2 Maximum A Posteriori . . . 41
2.5 Compressive Sensing . . . 42
2.5.1 The Nyquist-Shannon Sampling Theorem . . . 43
2.5.2 Sparsity, Incoherence, and the Restricted Isometry Property . . . 43
2.5.3 Solving Inverse Problems For Compressive Sensing . . . 46
2.6 Conclusion . . . 51

Chapter 3 Static Computational Optical Undersampled Tracker . . . 53
3.1 Motivation for the Static Computational Undersampled Tracker . . . 53
3.2 SCOUT Architecture . . . 56
3.3 Optimizing the SCOUT . . . 58
3.3.1 Simulating a SCOUT System . . . 58
3.3.2 Quantifying Reconstruction Error . . . 60
3.3.3 Optimizing Optical System Parameters . . . 60
3.4 Experiment . . . 62
3.4.1 Experimental Setup . . . 62
3.4.2 Calibration . . . 63
3.4.3 Reconstruction: ℓ_1 regularized Least Squares Minimization . . . 65
3.4.4 Experimental Results . . . 66

Contents – *Continued*

3.5 Conclusion 69

Chapter 4 Adaptive Feature Specific Spectral Imaging-Classifier 71

Chapter 5 Computational Spectral Unmixing 72

Chapter 6 Conclusion 73

Appendix A Derivation of the Least Squares Estimator 74

Appendix B SCOUT Experimental Results 76
B.1 Zero Background Difference Frames 76
B.2 Non-Zero Background Difference Frames 80

Appendix C The Psuedo-Code For SCOUT Experiment 84

Appendix D SCOUT Simulation Code 88

Glossary 97

Acronyms 100

Symbols 102

List of Figures

1.1 A systems view of a traditional sensing scheme. 14
1.2 A systems view of a computational sensing scheme. 14
1.3 A pinhole camera. 16
1.4 An isomorphic slit spectrometer with a 4F configuration. 17
1.5 A general flowchart for image and data compression techniques. . . . 24
1.6 A single pixel camera architecture. 26

2.1 The architecture of the Fourier Transform Spectrometer. 35
2.2 Graphical demonstration of joint probability. 38
2.3 Candy example for updating probabilities with Bayes' theorem 40
2.4 Example of sparse signal recovery using ℓ_1 regularized least squares algorithms 50
2.5 The sensing matrix used for Figure 2.4 50
2.6 Geometric interpretation of ℓ_1 regularized least squares. 51

3.1 The architecture of a hypothetical parallel single pixel camera to capture simultaneous projections. 54
3.2 Flowchart of SCOUT system architecture. 55
3.3 The SCOUT architecture. 57
3.4 An example of the system matrix of the SCOUT. 59
3.5 The coherence and reconstruction error versus defocus distance. . . . 62
3.6 The coherence and reconstruction error versus pitch of mask 2. 63
3.7 Photograph of SCOUT recording a moving object scene of a black background. 64
3.8 Photograph of SCOUT camera disassembled to show the lens and the first mask. 64
3.9 Difference frame 1 of a sequence of two movers on a black background. 67
3.10 Difference frame 9 of a sequence of two movers on a black background. 68
3.11 Difference frame 1 of a reconstruction video of two movers on a non-zero background. 68

B.1 Difference frame 1 of a sequence of two movers on a black background. 76
B.2 Difference frame 2 of a sequence of two movers on a black background. 76
B.3 Difference frame 3 of a sequence of two movers on a black background. 77
B.4 Difference frame 4 of a sequence of two movers on a black background. 77
B.5 Difference frame 5 of a sequence of two movers on a black background. 77
B.6 Difference frame 6 of a sequence of two movers on a black background. 78
B.7 Difference frame 7 of a sequence of two movers on a black background. 78
B.8 Difference frame 8 of a sequence of two movers on a black background. 78
B.9 Difference frame 9 of a sequence of two movers on a black background. 79
B.10 Difference frame 10 of a sequence of two movers on a black background. 79
B.11 Difference frame 1 of a sequence of one mover on a non-zero background. 80
B.12 Difference frame 2 of a sequence of one mover on a non-zero background. 80

List of Figures – *Continued*

B.13 Difference frame 3 of a sequence of one mover on a non-zero background. 80
B.14 Difference frame 4 of a sequence of one mover on a non-zero background. 81
B.15 Difference frame 5 of a sequence of one mover on a non-zero background. 81
B.16 Difference frame 6 of a sequence of one mover on a non-zero background. 81
B.17 Difference frame 7 of a sequence of one mover on a non-zero background. 82
B.18 Difference frame 8 of a sequence of one mover on a non-zero background. 82
B.19 Difference frame 9 of a sequence of one mover on a non-zero background. 82
B.20 Difference frame 10 of a sequence of one mover on a non-zero background. 83

List of Tables

ABSTRACT

Implementing computational optical sensors often comes with various issues that many traditional sensors may not encounter.

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 Magnetic Resonance Imaging (MRI) technique, where the measurements are in Fourier space. While *isomorphic* sensors can provide flexible sensing in multiple applications, joint design of a *computational sensor* will lead to performance increases often in resource constrained scenarios. Throughout this chapter and the rest of this dissertation, I will provide many examples 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 of computational sensing. 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

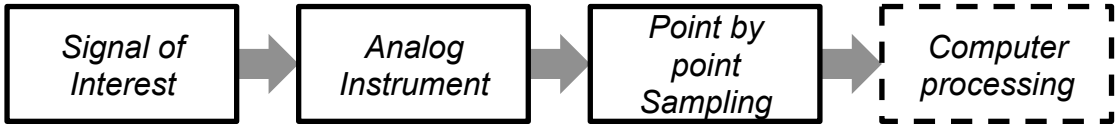


Figure 1.1: The signal-of-interest is incident upon the analog instrument. The analog instrument forms an isomorphism of the signal which is then periodically sampled point-by-point through an analog-to-digital converter (ADC) device. Once the signal is in digital form, post-processing algorithms are 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.

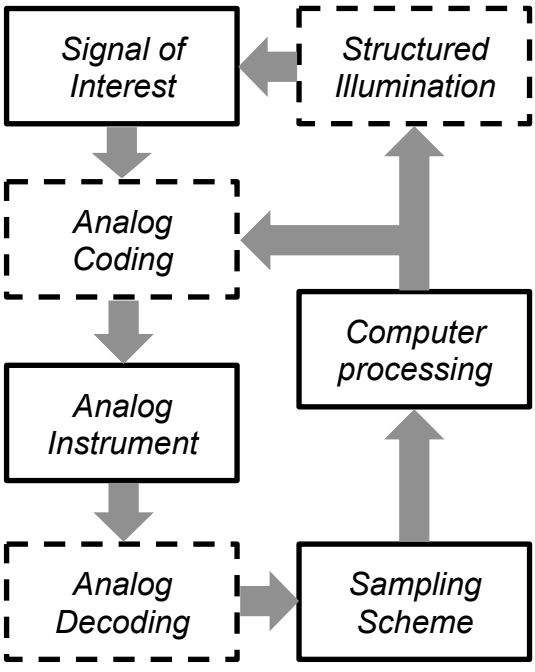


Figure 1.2: In a computational sensor, coding, decoding, and structured illumination are used in addition to the analog instrument, the algorithms, and sampling scheme. The dashed boxes represent optional processes that may or may not be needed.

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 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 focal-plane array (FPA). The more the image resembles the object the better the optics. 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 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

[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

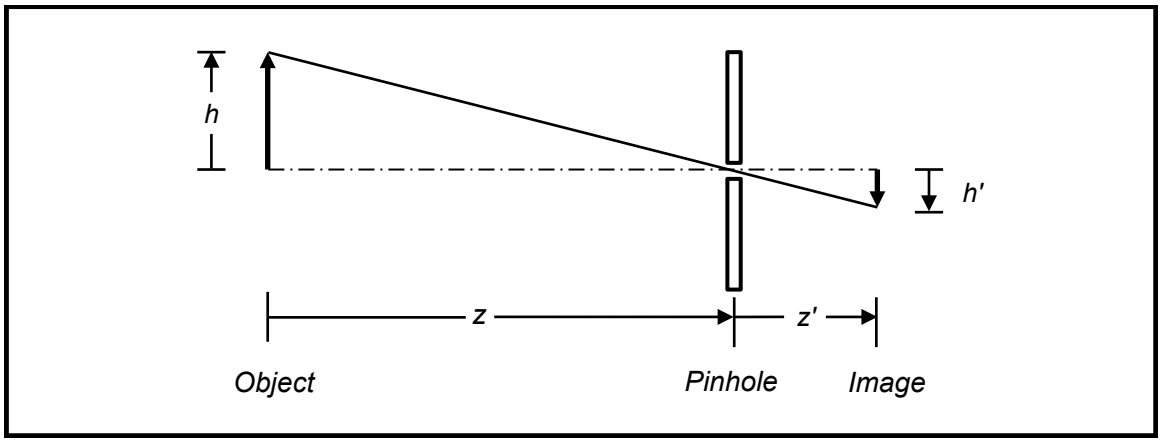


Figure 1.3: A pinhole camera is a simple imaging system that forms an image without a lens or mirror. This is due to the ray nature of light. A small hole will only admit a small amount of rays from an object point that is radiating light. Each point on a object emits light at different angles, and the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

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 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

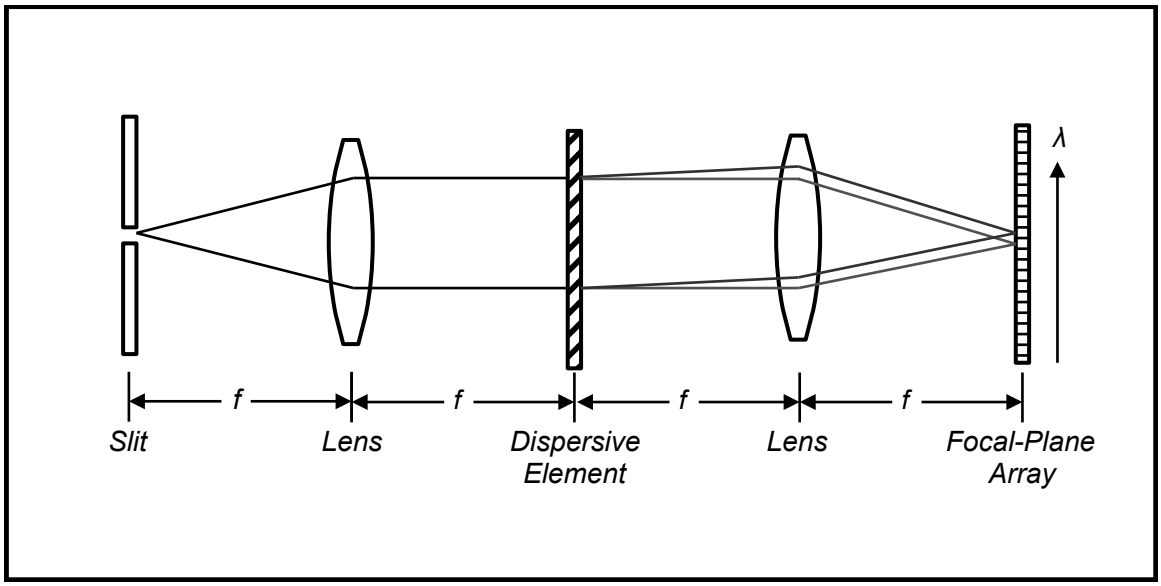


Figure 1.4: An isomorphic slit spectrometer with a 4F configuration. The slit limits the lateral extent of the object (or intermediate image). The light from the slit is collimated and separated into different angles based on the wavelength. A second lens then images wavelength shifted copies of the 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.

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 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 parallel fashion at each exposure, each pixel 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 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* [3]. 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 [4, 5].

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 [6].

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 [7]. 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 imaging. 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 [8, 9]. 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 [10, 11], spectral domain [12, 13], and temporal domain [14, 15]. 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), MRI and certain forms of sonic and radio wave imaging. All require a data-processing or reconstruction step to solve an inverse problem [16].

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 [17]. The inversion relies on computing the inverse Radon transform [18].

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 [19, 20].

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 [21]. 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 [22].

The invention of the digital camera by Sasson followed shortly after [23]. 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 [24].

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 [25, 26].

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 reconstructing the image.

In the next section I will discuss how many of the techniques and algorithms used

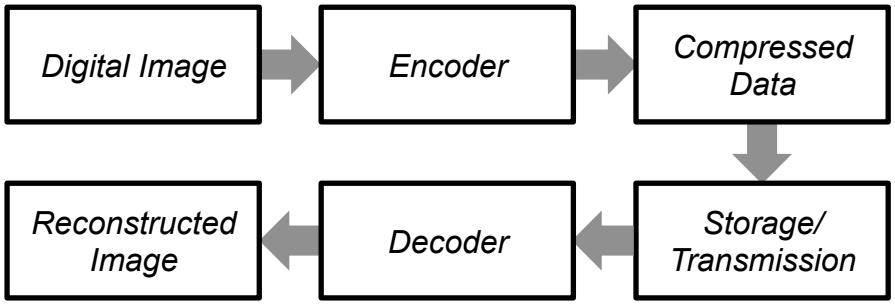


Figure 1.5: A general flowchart for image and data compression techniques. The digital image is analyzed by the encoder and compressed into a compressed form where it can be efficiently stored or transmitted. The decoder takes the compressed image and converts it back into an image that resembles the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

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 [27]. 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 [28, 29, 30]. 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

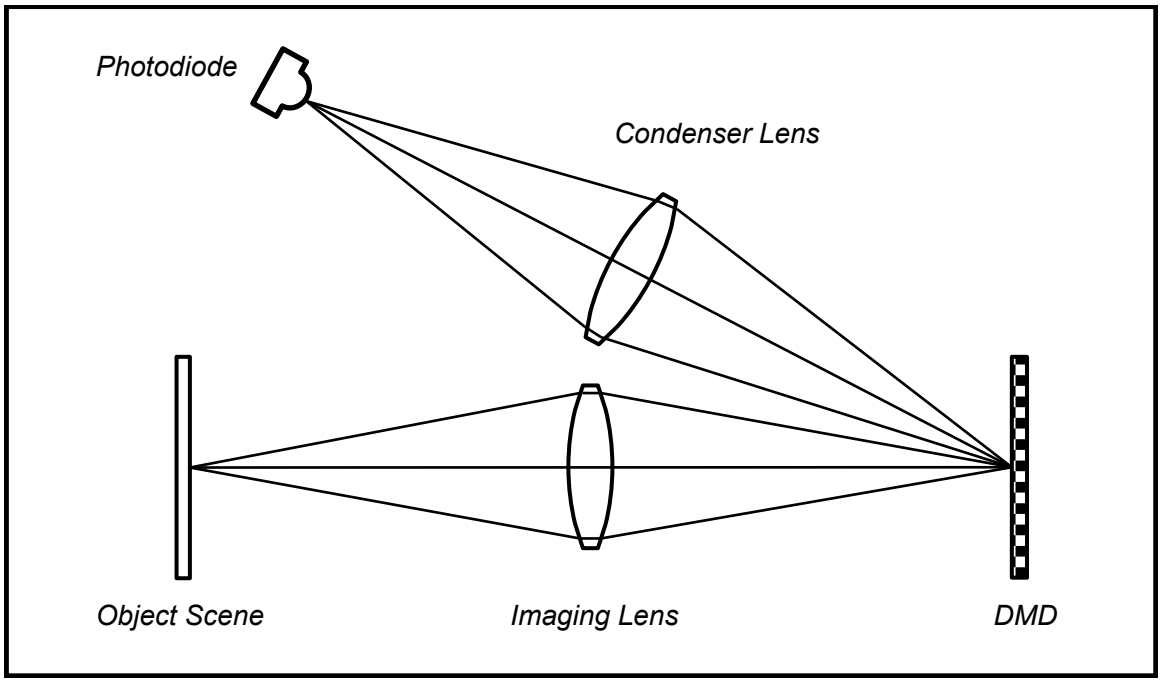


Figure 1.6: A single pixel camera architecture. The object scene (or intermediate image) is imaged onto a Digital Micro-Mirror Display (DMD). The each micro-mirror of DMD reflects light towards the photodiode or to another direction. This acts as a point-by-point multiplication of the discrete image 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.

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 [10]. 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 [31], 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 [32].

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

1.7 Practical Considerations in Computational Sensing

So far I have the 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 [33]. There is now a major Optical Society of America (OSA) meeting dedicated to computational sensing [34] and textbooks dedicated to its study and development [1, 35]. 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 [36]. 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.

Another issue facing sensors that rely on multiplexing is lack of dynamic range. In the context of optics, if more photons are being combined onto a single detector pixel, there is a higher chance of saturating the detector or operating in a non-linear regime. 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 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 [10]. 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 are 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 the 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 [30, 37, 38]. However, designed codes have been shown to outperform random codes in various applications. 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 show in the AFSSI-C, I will demonstrate 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 the next chapter 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 sensor to create a context for the discussion of the Fellgett advantage. Then I will discuss Principal Component Analysis (PCA), a technique for finding useful discriminating features from seemingly complicated data. I will also cover the fundamentals of Bayesian statistics and how is 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 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. For the most part, classical optics—the ray and wave nature of light—can be approximated a linear. The solutions to the paraxial wave equation are linear in free space and in most glasses. Similarly, paraxial optics can be model 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 [39]. 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 the estimated weights is the variance

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

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

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.

[1] Unbiased means that on average the estimator will produce the ground truth

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, meaning that $\mathbf{f}$ and $\mathbf{g}$ from Equation (2.1) are both vectors in N dimensional space [39].

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 [39]. 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 $\text{MSE} = \frac{\sigma^2}{N}$ [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 incoherent light where a phase delay maybe used to produce a negative electric field. For incoherent light, the system response is linear in intensity [40].

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 [39]. In the weighing example, a spring balance rather than a two pan scale 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 [41, 42]. Physically the Fellgett advantage occurs because a single detector element produces a noise contribution whether it is sampling a single part of the signal or multiple parts 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

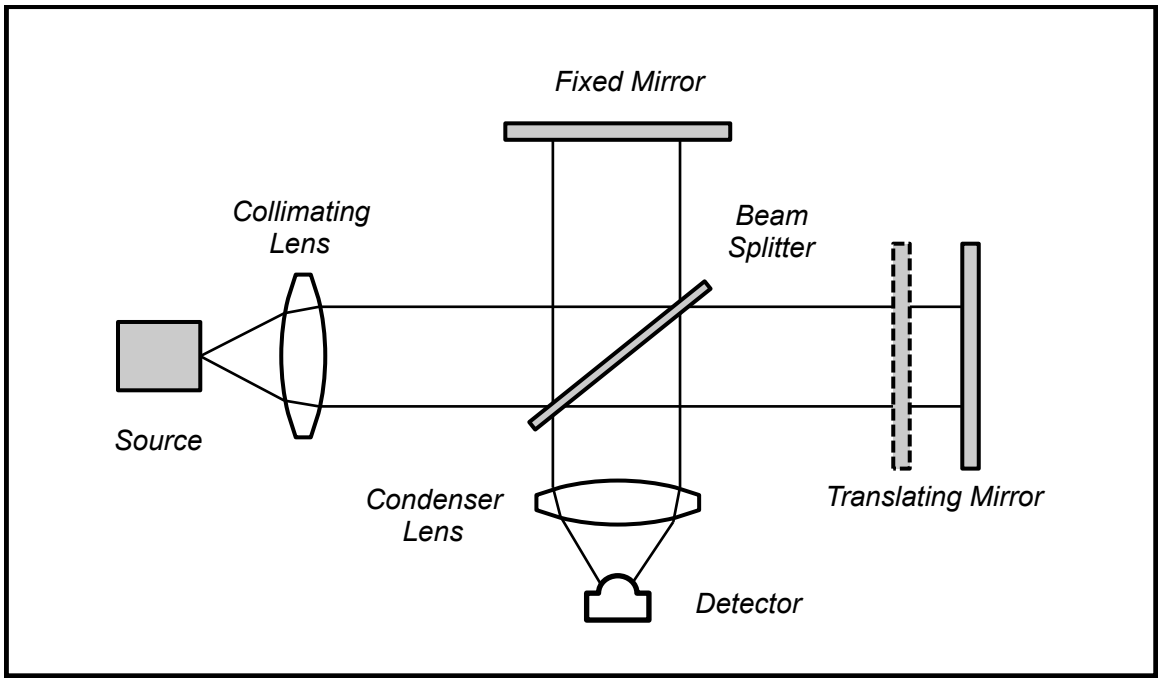


Figure 2.1: The architecture of the Fourier Transform Spectrometer resembles the Michelson interferometer. One of the mirrors is translated back and forth. The interferogram is the detected intensity versus mirror delay which is related to the autocorrelation of signal. The Fourier transform of the autocorrelation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

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 [42]. The Wiener-Khinchin theorem says that for a wide-sense stationary random process, the Fourier transform of the autocorrelation is the power spectral density [43]. 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 [44].

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

where N_λ is the number of spectral channels.

2.3 Principal Component Analysis

Principal component analysis (PCA) is a technique that attempts to recast a dataset in 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 dimensionalpoints al 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 that also maximizes variance but is orthogonal to the first principal component. The third principal component is orthogonal to the first and second principal components and points in the direction that maximizes the variance given those constraints. 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 allows one to recast the data to make it as uncorrelated as possible in a new basis.

$$\mathbf{Y} = \mathbf{PS} \tag{2.15}$$

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

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

is a now diagonal matrix. Indeed, the principal components are the optimal way to discriminate the spectra in the dataset [45]. It turns out the principal components are the *eigenvectors of the covariance matrix* $\mathbf{C_S}$ *are the principal components* [46].

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

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

Where γ_n is the n^{th} eigenvalue 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, $M \ll N_\lambda$. 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 [47]. The magnitude of the eigenvalue γ_n tells us how well useful the associated eigenvector is at discriminating the data.

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 for 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.

In 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.

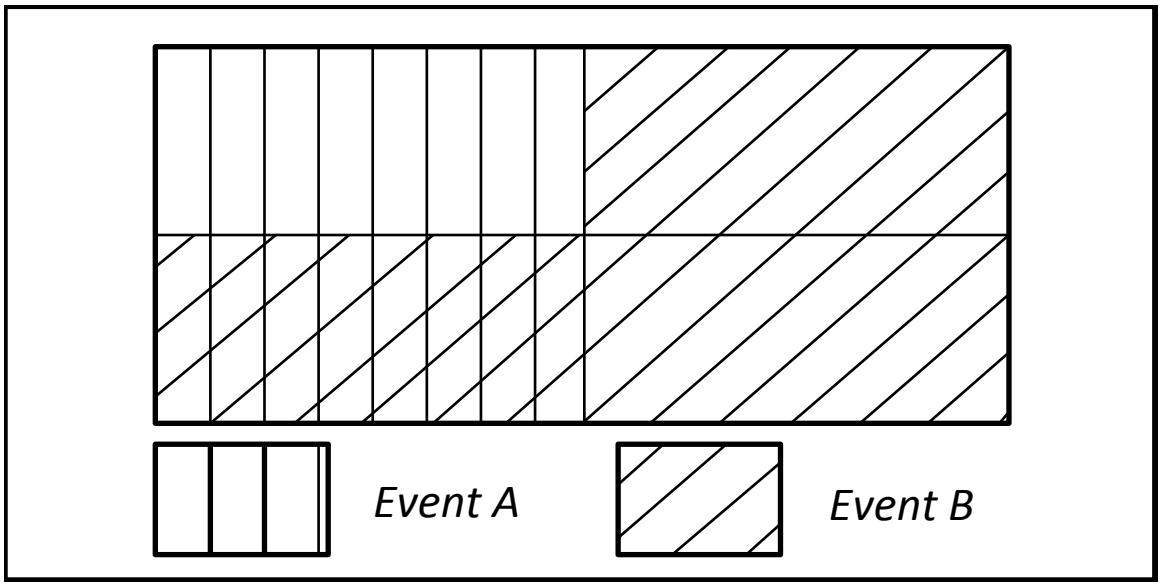


Figure 2.2: Graphical demonstration of joint probability. The probability of event A is $P(A) = 3/4$. The probability of event B is $P(B) = 3/4$. The joint probability of events A and B is $P(A \cap B) = 1/4$. The probability of event A occuring given that event B has occured is $P(A \,|\, B) = 1/3$.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

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. For example, Bayes' theorem can be used to decide which of two bags of candy has been opened or if a spectrum is present. The idea is that one can make 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 $P(A \cap B)$ gives

$$P(A \cap B) = P\left(A \mid B\right) P(B) \tag{2.19}$$

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

$$P(A \cap B) = P\left(B \mid A\right) P(A) \tag{2.20}$$

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

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

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

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

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

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

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

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

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

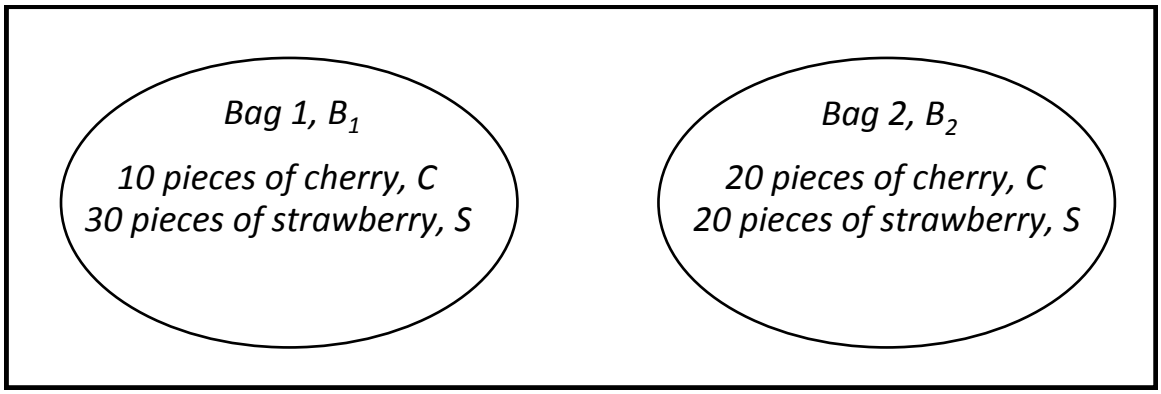


Figure 2.3: Two bags are present: B_1 and B_2. The probability of choosing either bag is $P(B_1) = P(B_2) = 1/2$. Bag 1 has 40 pieces of candy: 10 pieces of cherry candy so $P(C|B_1) = 1/4$ and 30 pieces of strawberry so $P(S|B_1) = 3/4$. Bag 2 has 40 pieces of candy: 20 pieces of cherry candy so $P(C|B_2) = 2/4$ 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.

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 given a set of hypotheses, $\{\mathbf{h}_1, \mathbf{h}_2, \cdots, \mathbf{h}_{N_c}\}$, one is 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 $\mathbf{h}_k$ which maximizes the posterior probability is the most likely one.

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

Using Bayes' theorem I can rewrite Equation 2.31 as

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

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

$$\frac{p(\mathbf{h}_i|g)}{p(\mathbf{h}_j|g)} = \frac{p(g|\mathbf{h}_i)}{p(g|\mathbf{h}_j)} \frac{p(\mathbf{h}_i)}{p(\mathbf{h}_j)} \tag{2.33}$$

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

$$\Lambda_m = \frac{p(g_m|\mathbf{h}_i)}{p(g_m|\mathbf{h}_j)} \tag{2.34}$$

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

$$\frac{p(\mathbf{h}_i|\{g\}_{N_m})}{p(\mathbf{h}_j|\{g\}_{N_m})} = \prod_{m=1}^{N_m} \Lambda_m \frac{p(\mathbf{h}_i)}{p(\mathbf{h}_j)} \tag{2.35}$$

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 [48, 49].

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 [50]. 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 [51].

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 $N_m \ll N$ 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 [52, 28, 35, 37]. 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.

Most continuous signals $f(x)$ 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

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

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 [37].[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,

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

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 compressive imaging experiments, the sensing basis is the binary random coded aperture mask [10]. In an LCOS based compressive sensing spectrometer, the sensing basis is a finite set of spectral filters [53, 54]. 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 less 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

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

for a high probability of reconstruction, where C is just a constant and N is the dimensionality of the original signal [37].

Equation (2.43) shows the importance of sparsity and coherence in compressive sensing. Lower coherence and sparsity allows one to use less measurements [10]. The more incoherent, and therefore lower correlation, the two baseds 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 [37].

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 [37]. 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 [37, 10, 35].

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 do not work well for the compressive sensing problem.

There is 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 relavent 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

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

[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.

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) [35].

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

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

where $\|x\|_0$ 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 [55]. This means that one can do no better than a brute force search for the answer.

Fortunately, one can minimize the ℓ_1 norm

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

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 [30, 28, 35]. 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

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

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

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

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 [56].

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) [57], truncated Newton interior point methods [58], and Conjugate Gradients algorithm [52].

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 [58, 59]. 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 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: $\|\mathbf{A}\mathbf{x} = \mathbf{g}\|_2^2$. 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, because of the shape of the ℓ_2 ball the solutions may or may not occur along the

[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.

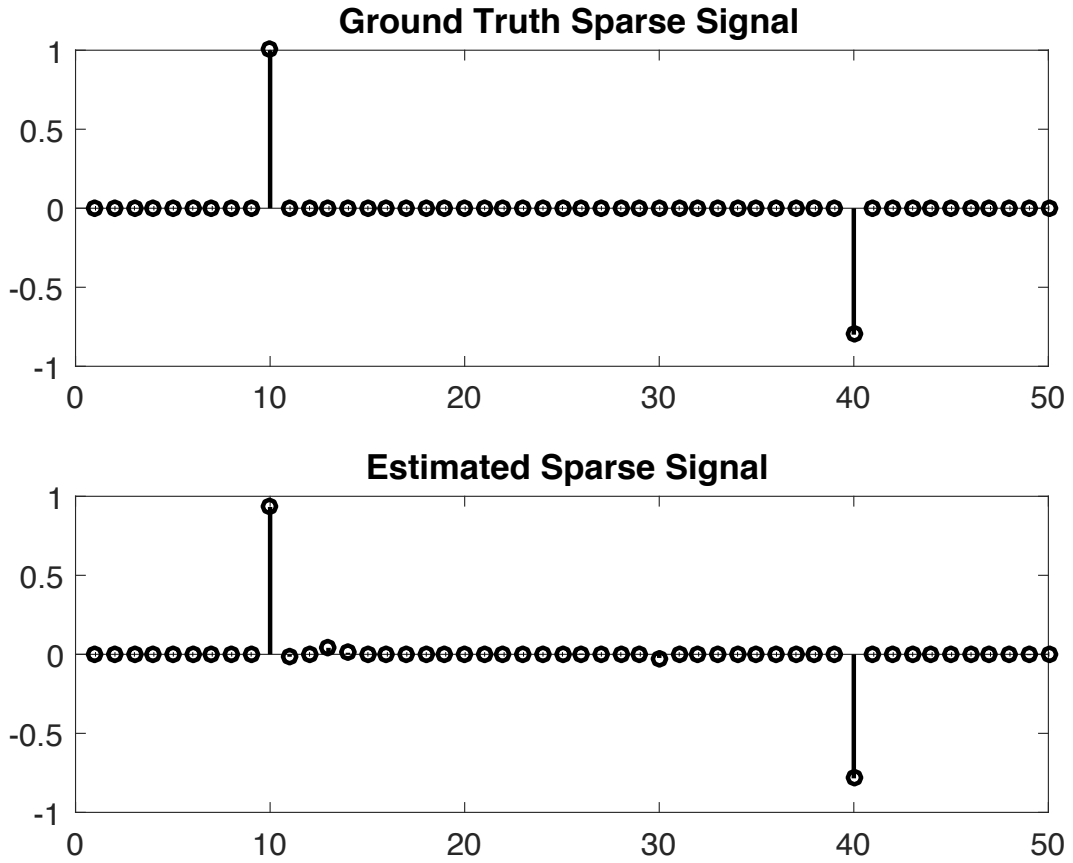


Figure 2.4: Example of sparse vector recovery using ℓ_1 regularized least squares algorithms. SNR = 100. The top plot shows the ground truth signal $\mathbf{x}$. The bottom plot shows the estimated signal $\hat{\mathbf{x}}$ and $\tau = 0.02$. The number of measurements is $N_m = 10$. The dimensionality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

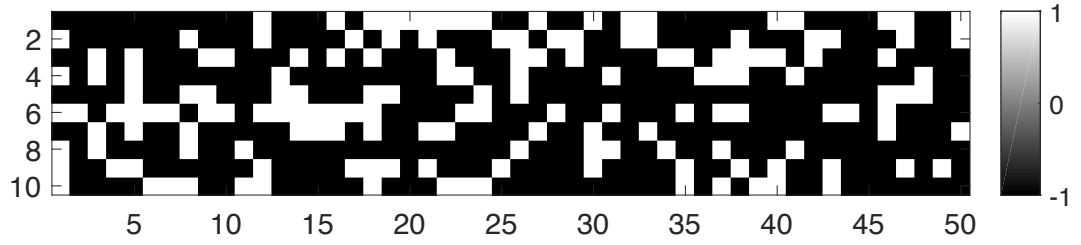


Figure 2.5: The sensing matrix $\mathbf{H}$ used for Figure 2.4. Since the signal was already sparse, the representation basis can be written as the identity matrix $\mathbf{\Psi} = \mathbf{I}$. The number of rows correspond to the number of measurements N_m and the number of columns correspond to the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

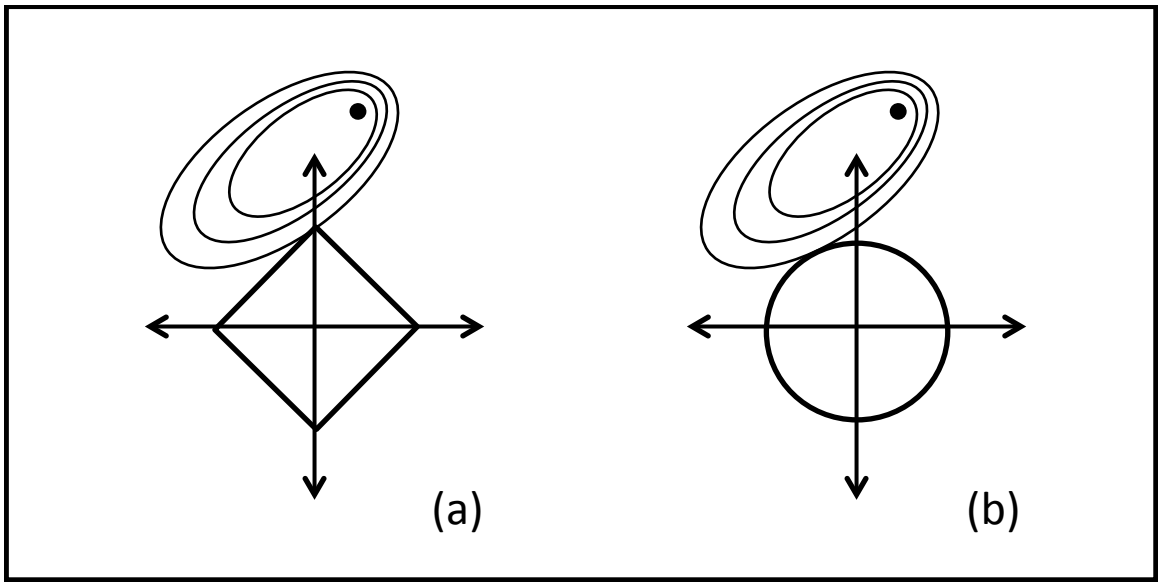


Figure 2.6: (a) ℓ_1 regularized least squares which is called lasso regression. (b) ℓ_2 regularized least squares which is called ridge regression.

coordinate axes.

The lasso regression technique in statistics is a way to perform variable selection without overfitting the data [60, 61]. 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 [57]. 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 and has a multitude of algorithms which can be used to solve it.

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) [11, 62]. 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 [15]. In another experimental compressive imager, a rotating cylindrical lens measures scene data using an optical Radon transform and then reconstructs a static object scene [63]. 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 [10]. Each measurement is an integrated point-by-point multiplication of the scene

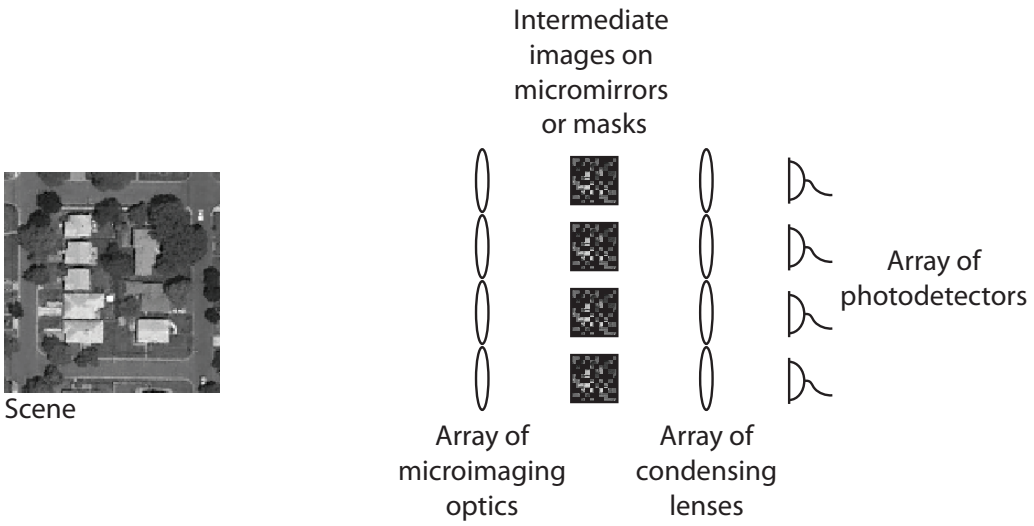


Figure 3.1: An example of a typical parallel optical CS architecture. Capturing N_m simultaneous projections requires using N_m spatial light modulators or masks and N_m detector elements.

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 [11, 64]. The SCOUT system is an important step toward practical low-cost computational sensing for optical imaging. The system enables parallel "single-shot" acquisition of compressed, task-specific sensing oriented data, using a static

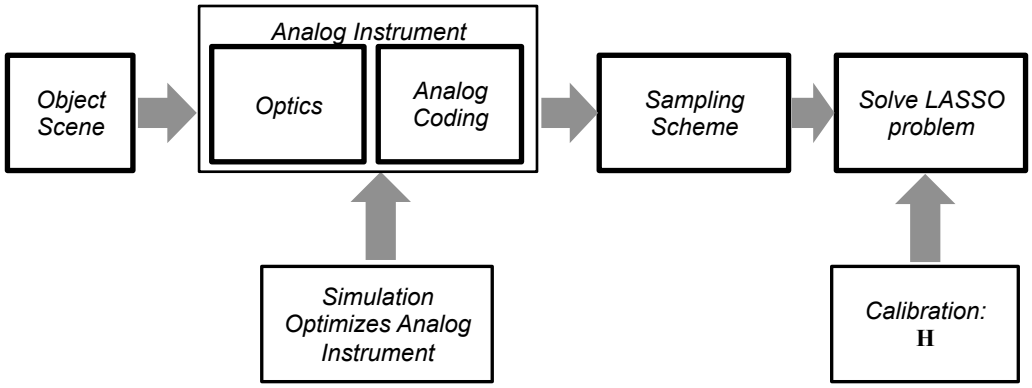


Figure 3.2: A systems level flowchart of the SCOUT system. Light from the object scene propagates to the optical instrument which consists of a coded aperture (mask 1), an objective lens, and another coded aperture (mask 2), a focal-plane array (FPA) then undersamples the coded object scene, measurement 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.

(no moving parts) architecture.

A related static approach uses optical Radon projections [65]. 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 only reconstructs frame-to-frame differences. As a result the SCOUT requires sig-

nificantly 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 [62, 66].

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 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 mea-

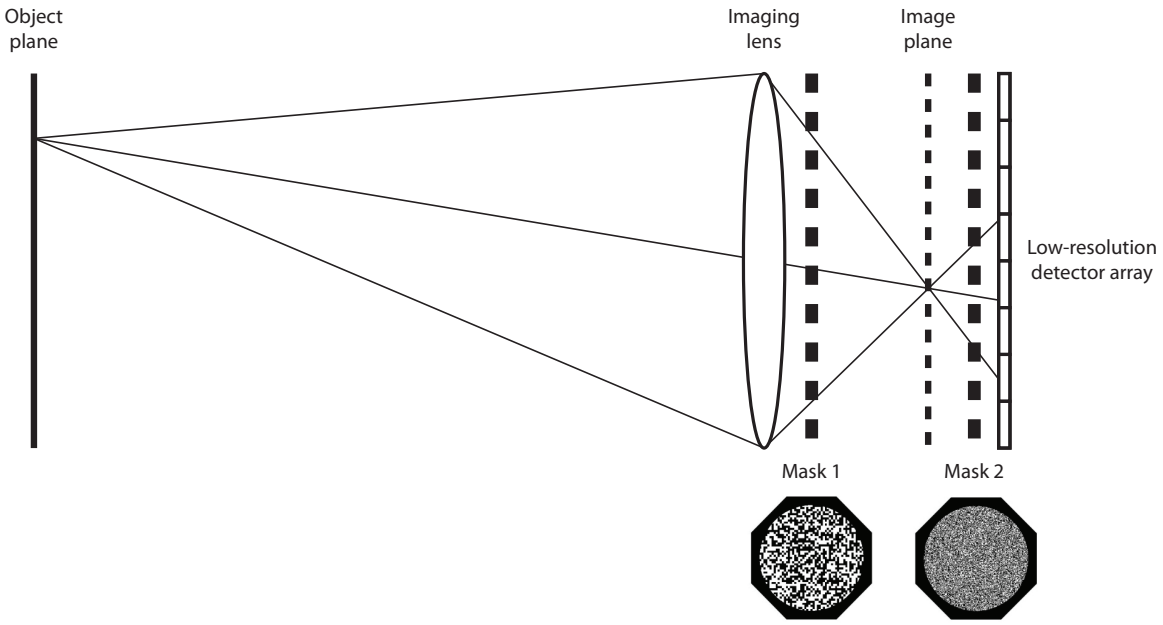


Figure 3.3: A diagram of the SCOUT architecture. A lens projects light through a pair of binary occlusion masks onto a low-resolution sensor which is defocused from the nominal image plane. This creates a spatially multiplexed shift-variant PSF incident on the FPA.

surement, $\mathbf{H}$ 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.

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

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 lexicography 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 **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.

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 **H**. Once **H** is known, we use it to simulate the low resolution measurements, **g**, of the higher resolution scenes, **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.

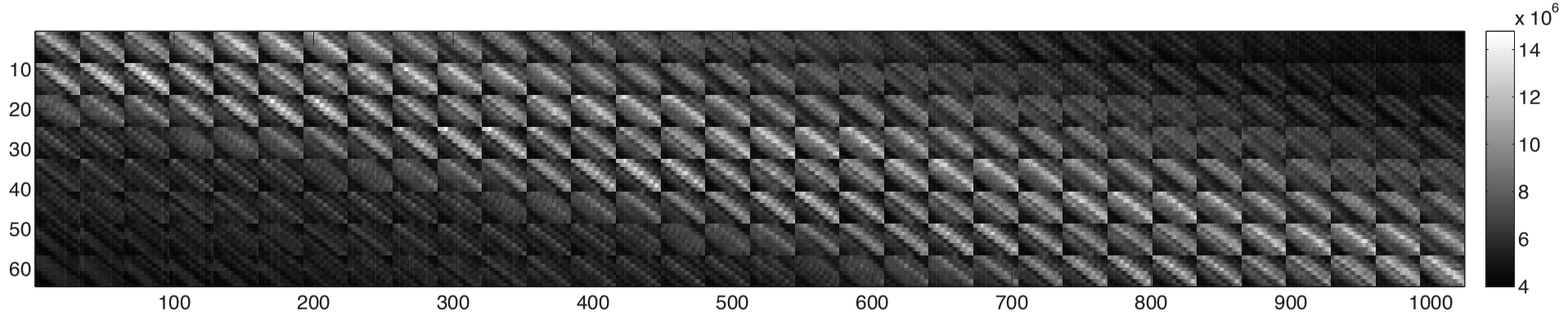


Figure 3.4: An example of an experimentally measured of the system matrix of the SCOUT system. The approximate block-Toeplitz structure is clearly evident, as is the deviation from the Bernoulli or Gaussian ensembles typically considered in CS treatments.

3.3.2 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.

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.

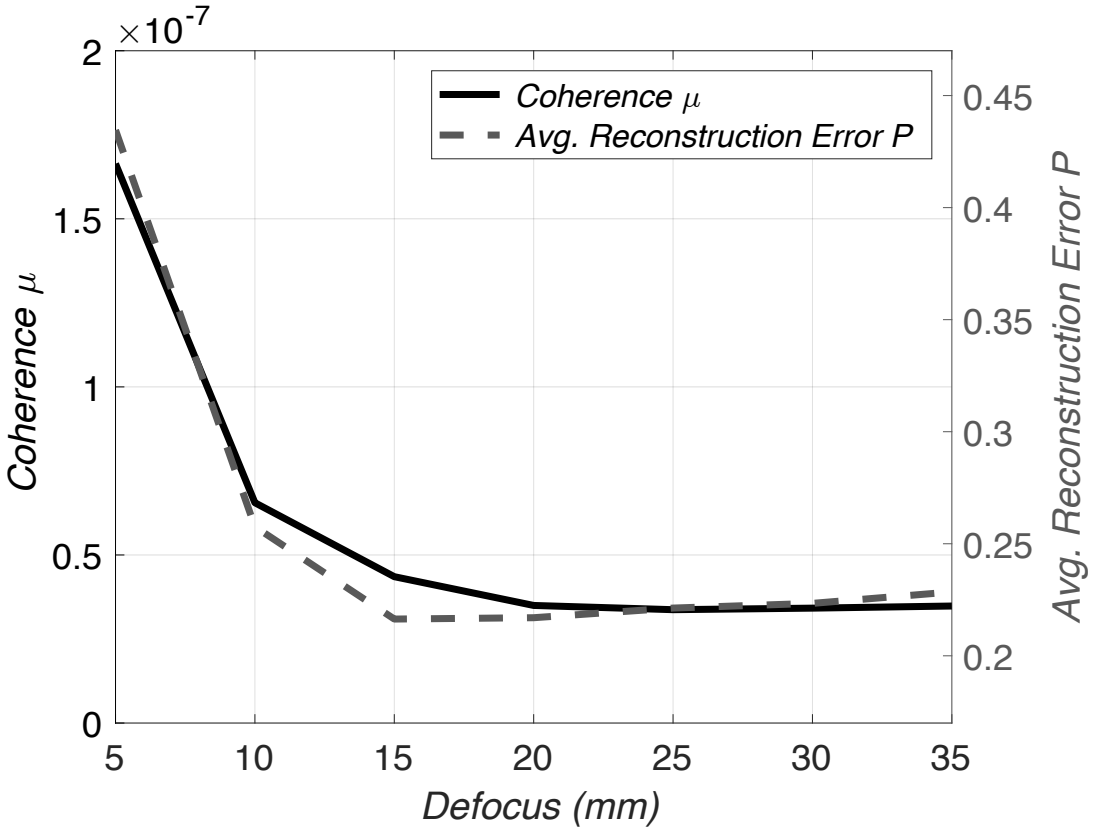


Figure 3.5: The coherence μ (left vertical axis - black) and reconstruction error P (right vertical axis - dashed gray) plotted as a function of defocus distance $d_i m$.

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 $p_1 = 30$ microns and fill factor $F_1 = 0.4$ and is located at a distance $d_{m1} = 14mm$ from the sensor. Mask 2 has pitch $p_2 = 500$ microns and fill factor $F_2 = 0.2$ and is located at a distance $d_{m2} = 57mm$ from the sensor. These parameters were chosen based on the aforementioned optimization process. A plasma monitor

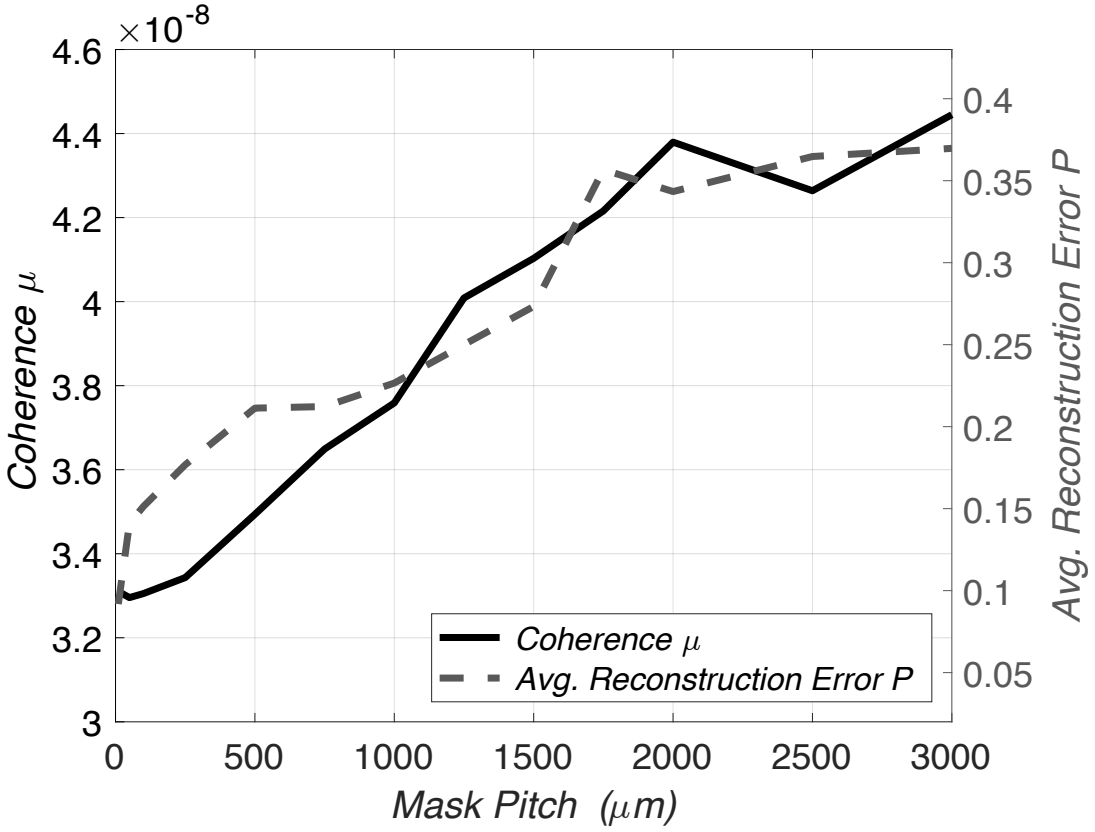


Figure 3.6: The coherence μ (left vertical axis - black) and the reconstruction error P (right vertical axis - dashed gray) is plotted as a function of the pitch of mask 2

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 **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 a straightforward. The goal is to experimentally determine the system matrix **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.

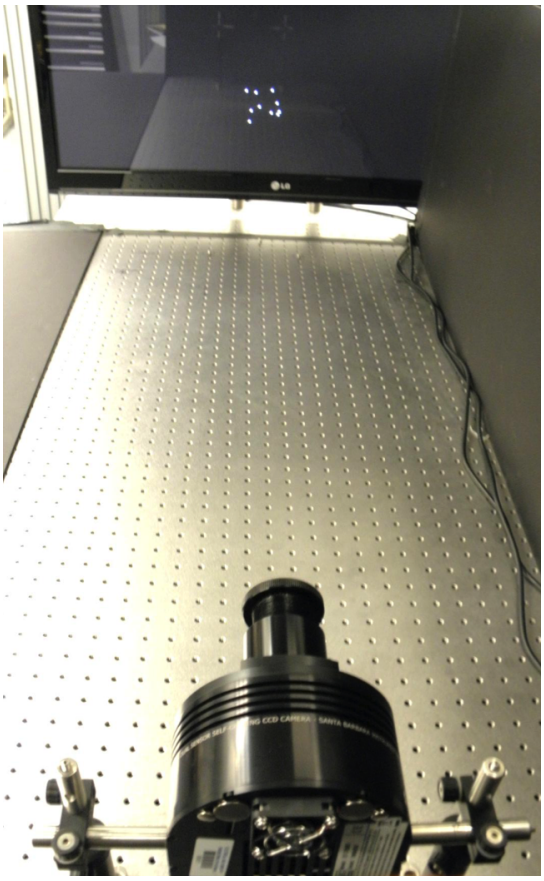


Figure 3.7: The camera captures images of scenes displayed on a plasma television approximately 2 meters away..

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.

All one needs to do is display the point at each location and store the measurement in the respective column of **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 cooling (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-case 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 [58]. 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, $\Delta\hat{\mathbf{f}}$.

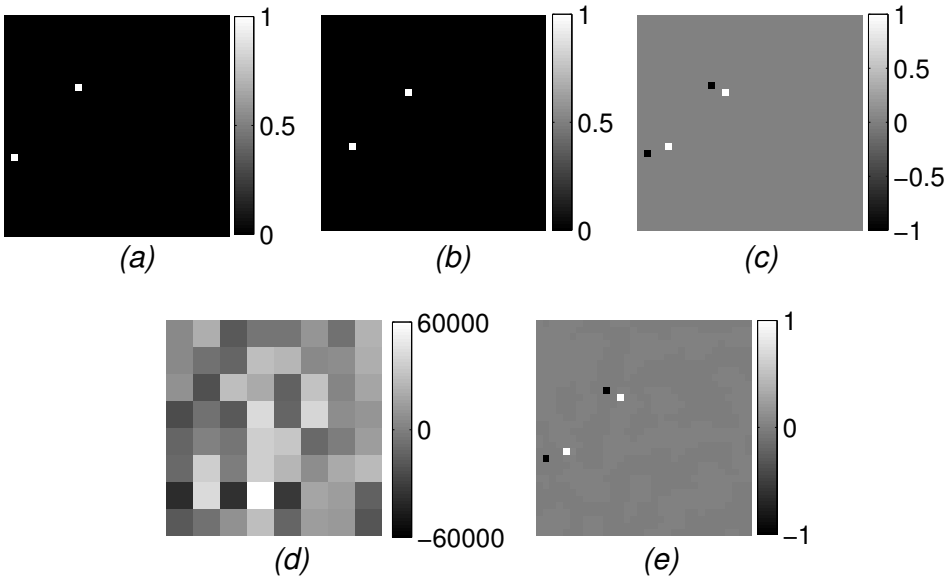


Figure 3.9: A reconstruction of a 32×32 scene with two movers of equal amplitude on a black background. (a) ground-truth scene 1 (b) ground-truth scene 2 (c) ground-truth frame difference and (d) measured 8×8 frame difference, scaled so that it is discernible (e) reconstructed 32×32 difference frame

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

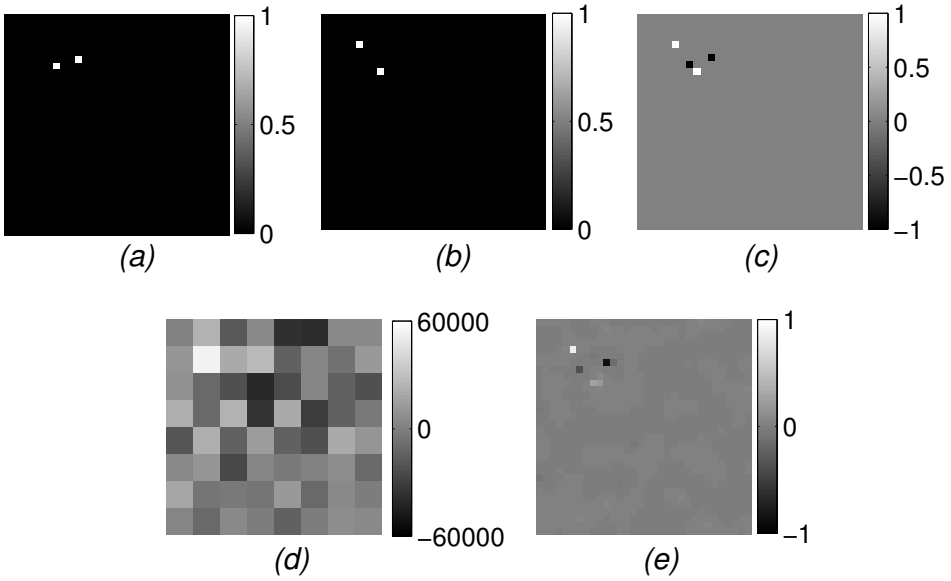


Figure 3.10: A reconstruction of 32×32 scene with two movers of equal amplitude on a black background. This frame shows the results when the past and present mover locations are adjacent in the difference frame. (a) ground-truth scene 1 (b) ground-truth scene 2 (c) ground-truth frame difference and (d)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

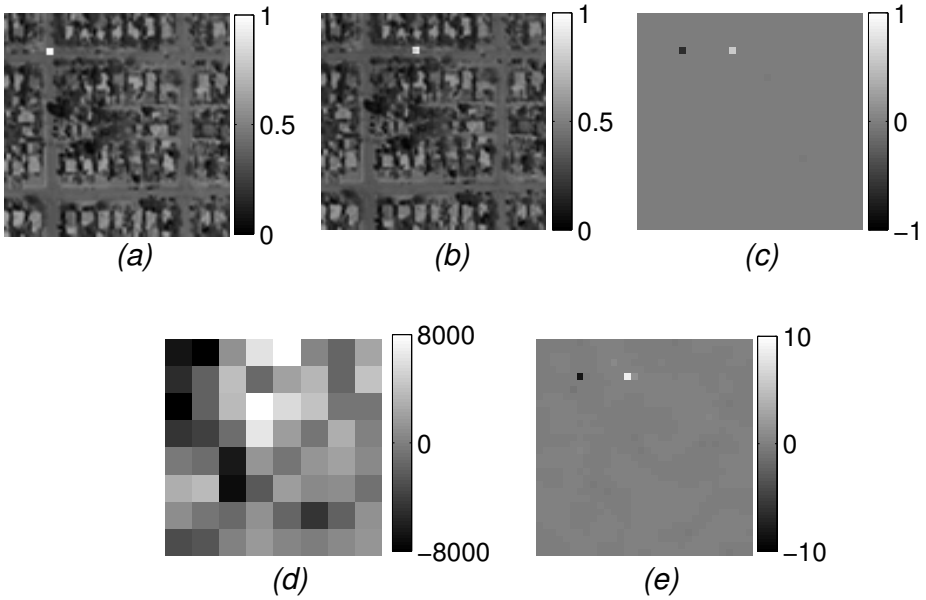


Figure 3.11: Difference frame 1 of a video demonstration of compressive tracking of a 32×32 difference scene with a non-zero background. Scene background ©2012 Google. (a) ground- truth scene 1 (b) ground-truth scene 2 (c) ground-truth frame difference and (d) measured 8×8 frame difference, scaled so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

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 fram $\Delta\mathbf{f}$ shown in Figure 3.11(c), the amplitude of the past and present mover locations in the ground-truth are lower 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 less 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 agains 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 is 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 [67, 68, 69], 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 force random coding. Two exceptions are [70, 71], 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-norm minimization algorithm and we 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

This chapter introduces the reader the AFSSI-C.

Chapter 5

Computational Spectral Unmixing

This chapter introduces the reader the computational spectral unmixer.

Chapter 6

Conclusion

This chapter concludes the dissertation.

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

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

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

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

So equation A.4 becomes

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

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

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

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

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

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

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

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

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

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.

B.1 Zero Background Difference Frames

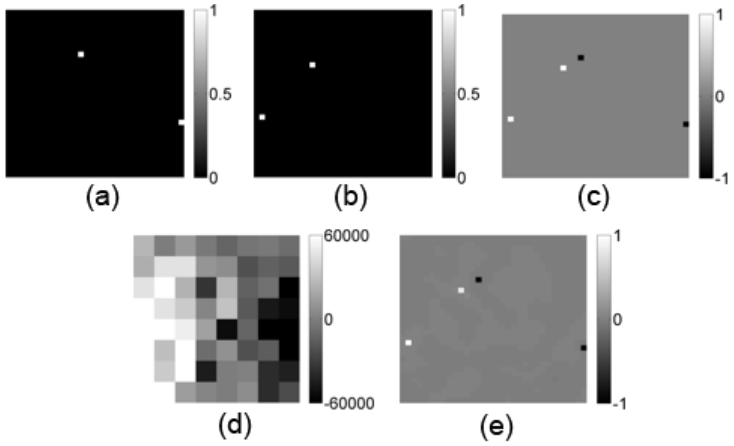


Figure B.1: Difference frame 1 of a sequence of one mover on a black background.

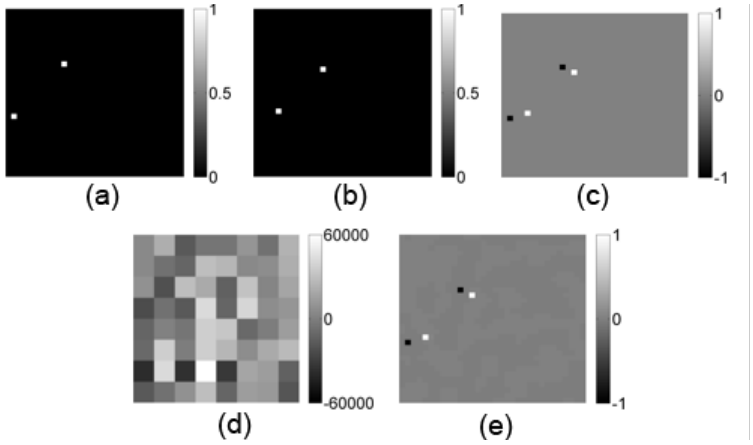


Figure B.2: Difference frame 2 of a sequence of one mover on a black background.

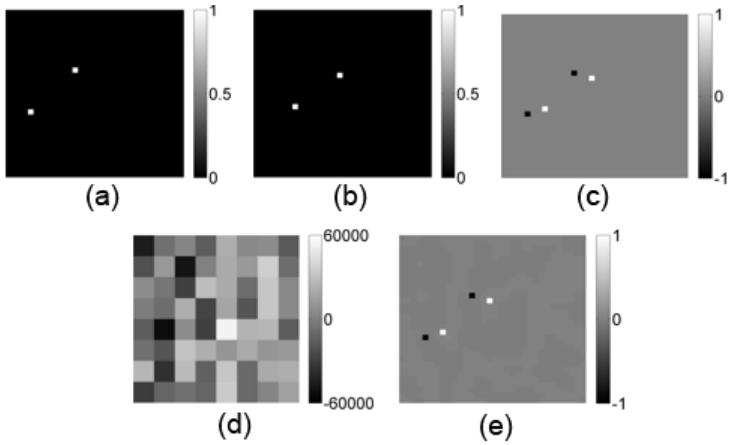


Figure B.3: Difference frame 3 of a sequence of one mover on a black background.

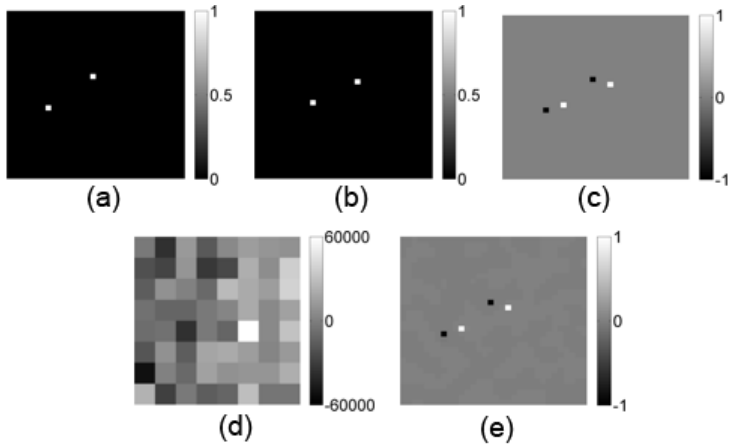


Figure B.4: Difference frame 4 of a sequence of one mover on a black background.

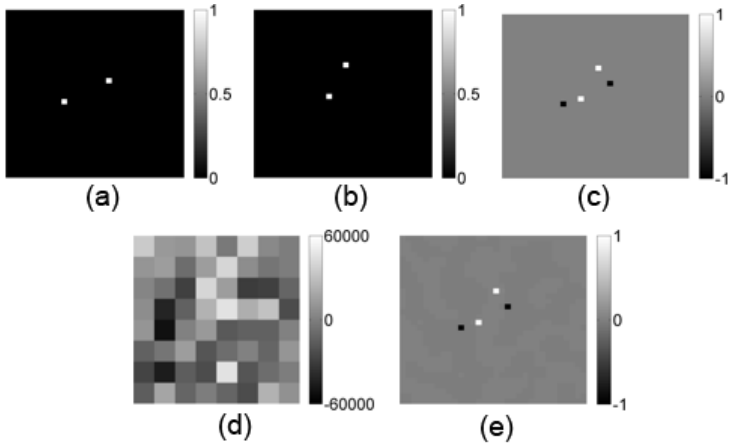


Figure B.5: Difference frame 5 of a sequence of one mover on a black background.

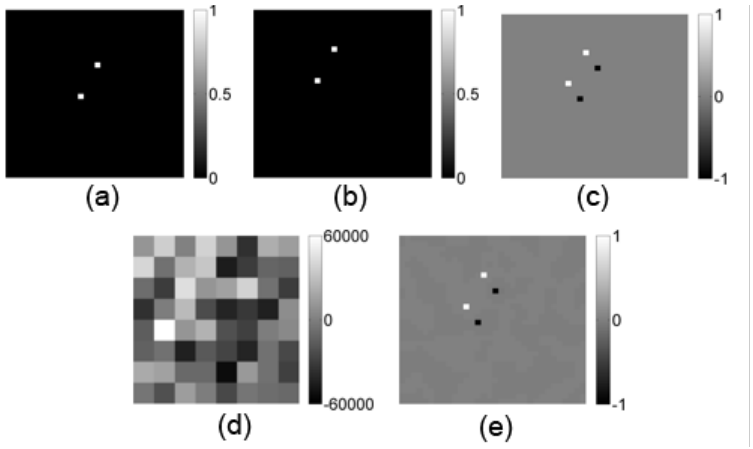


Figure B.6: Difference frame 6 of a sequence of one mover on a black background.

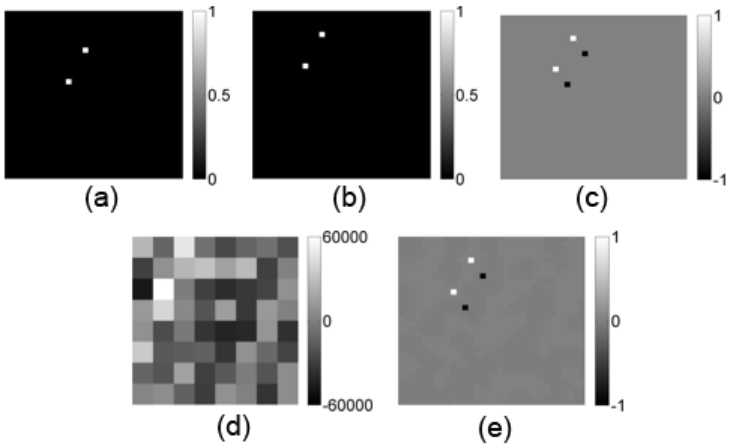


Figure B.7: Difference frame 7 of a sequence of one mover on a black background.

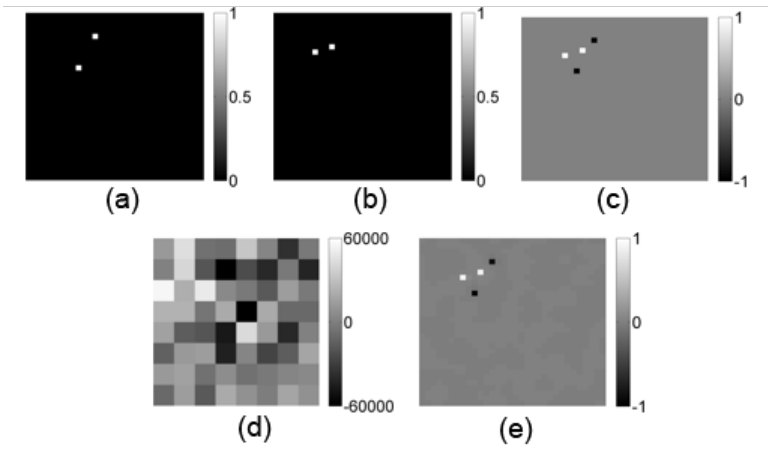


Figure B.8: Difference frame 8 of a sequence of one mover on a black background.

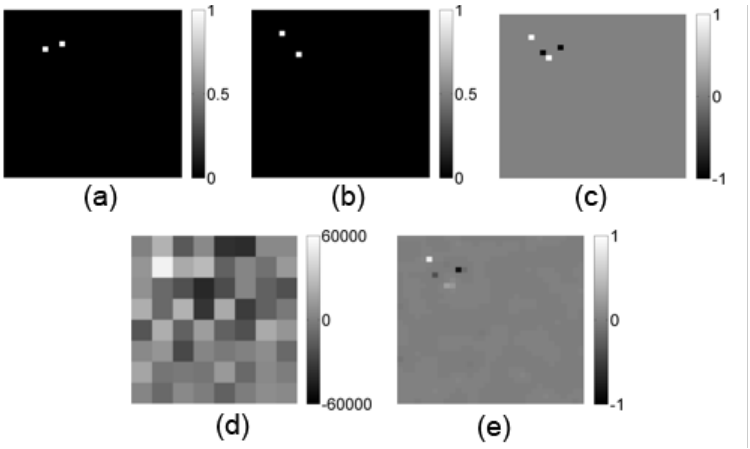


Figure B.9: Difference frame 9 of a sequence of one mover on a black background.

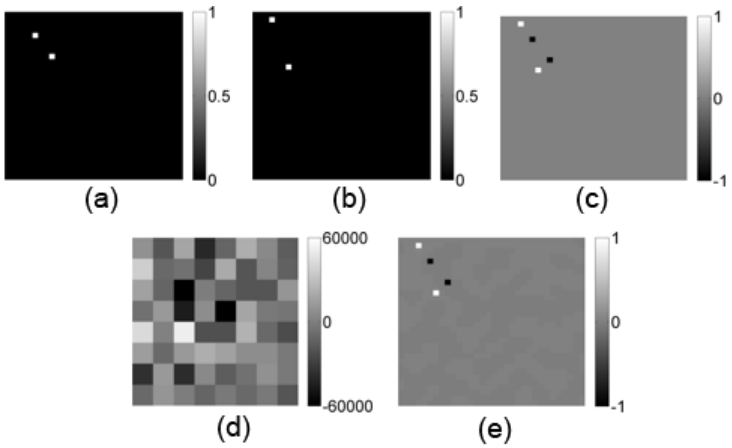


Figure B.10: Difference frame 10 of a sequence of one mover on a black background.

B.2 Non-Zero Background Difference Frames

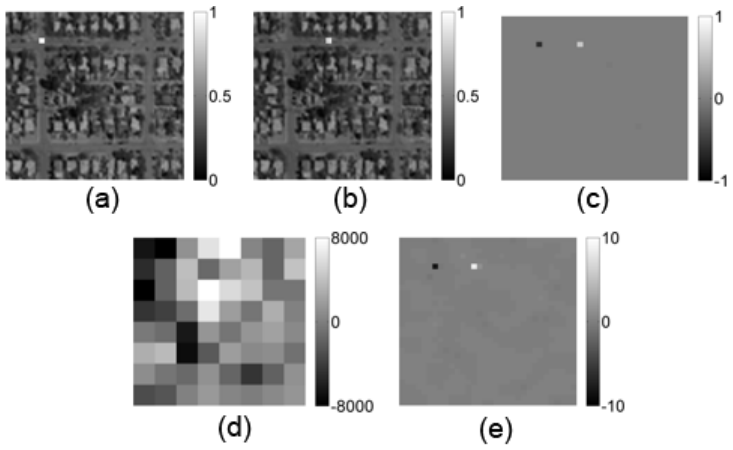


Figure B.11: Difference frame 1 of a sequence of one mover on a non-zero background.

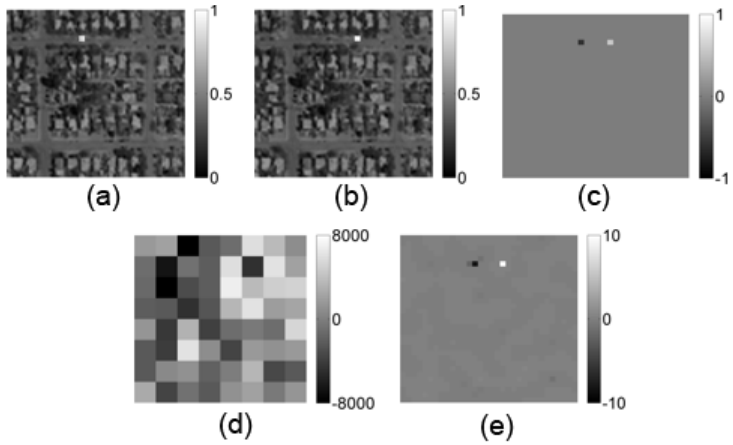


Figure B.12: Difference frame 2 of a sequence of one mover on a non-zero background.

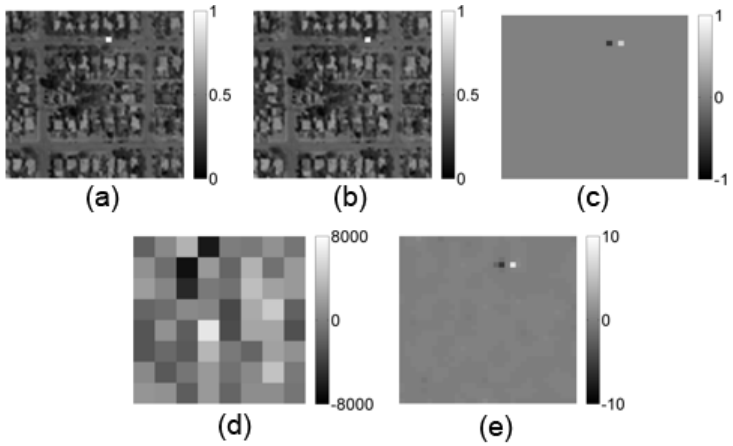


Figure B.13: Difference frame 3 of a sequence of one mover on a non-zero background.

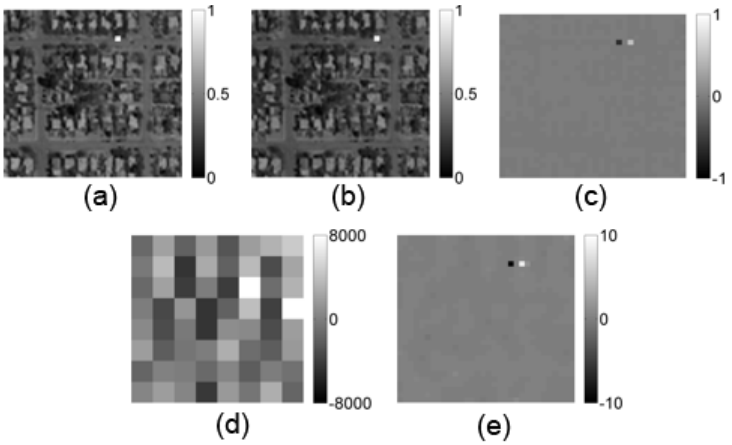


Figure B.14: Difference frame 4 of a sequence of one mover on a non-zero background.

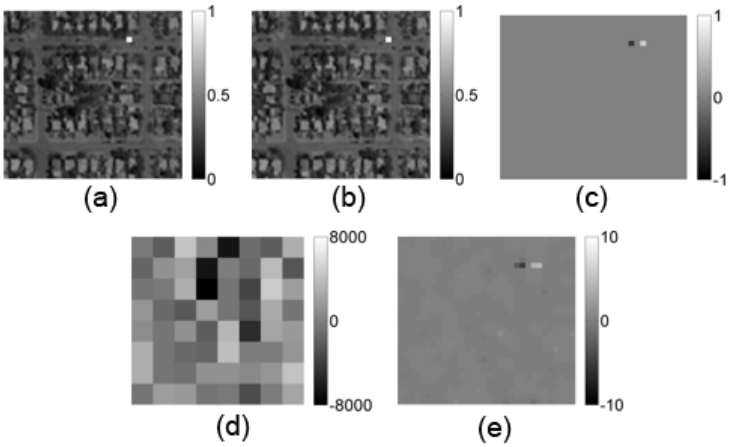


Figure B.15: Difference frame 5 of a sequence of one mover on a non-zero background.

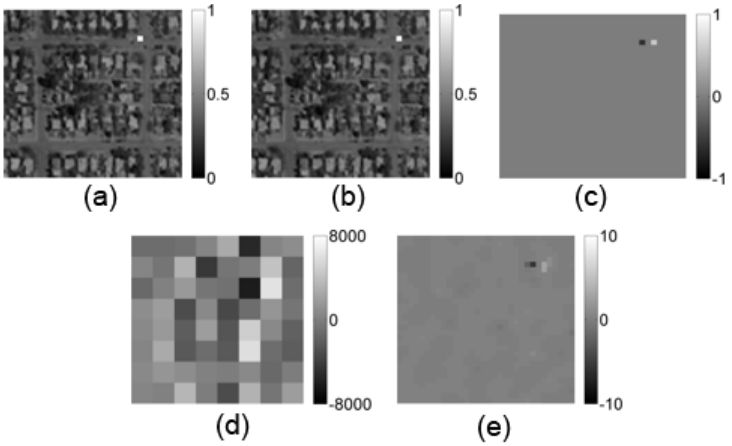


Figure B.16: Difference frame 6 of a sequence of one mover on a non-zero background.

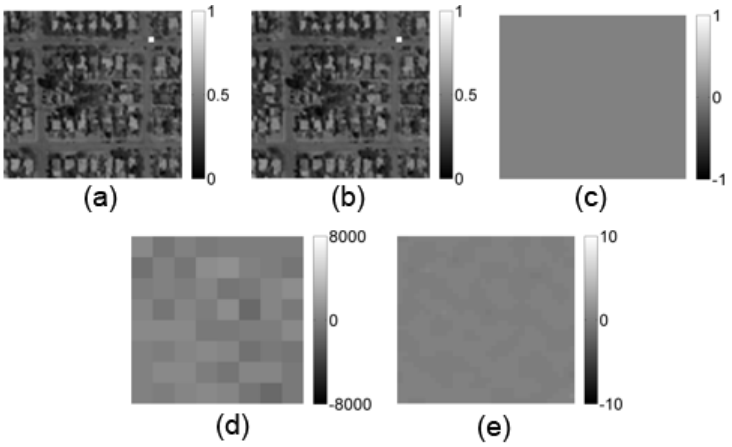


Figure B.17: Difference frame 7 of a sequence of one mover on a non-zero background.

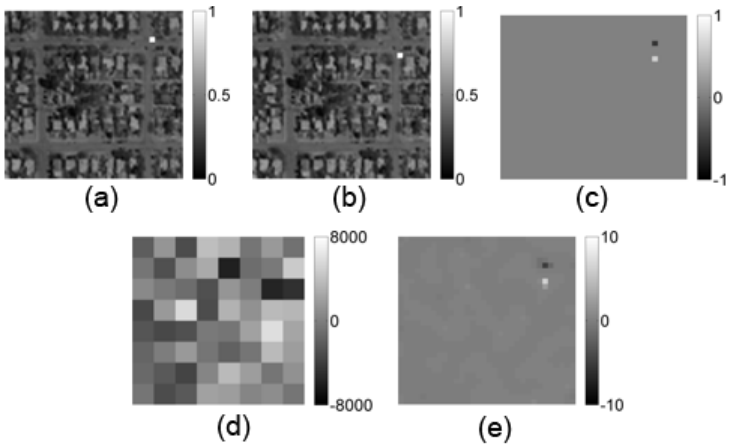


Figure B.18: Difference frame 8 of a sequence of one mover on a non-zero background.

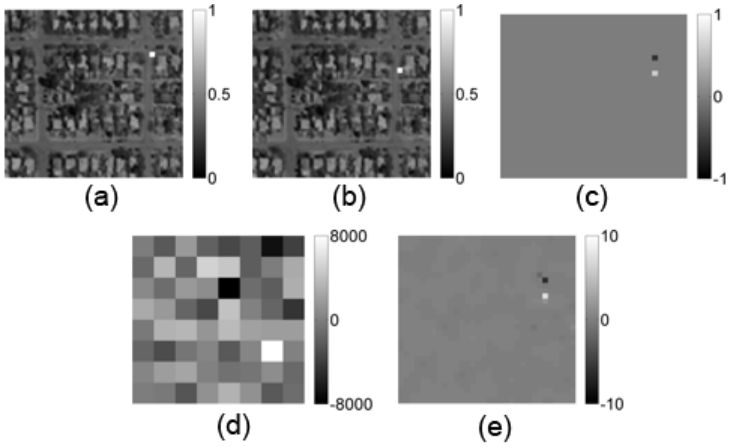


Figure B.19: Difference frame 9 of a sequence of one mover on a non-zero background.

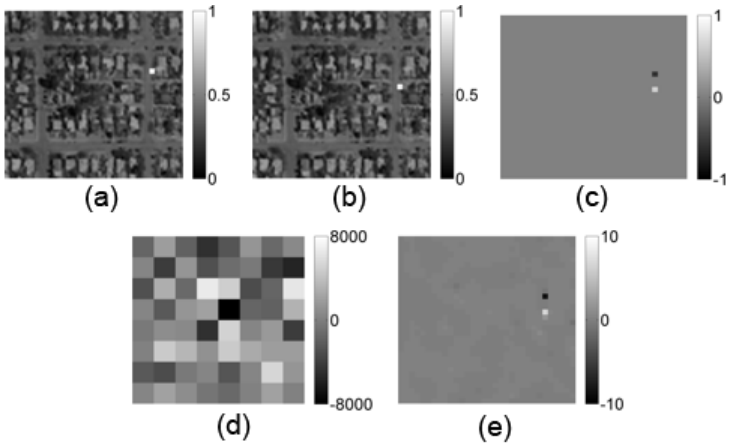


Figure B.20: Difference frame 10 of a sequence of one mover on a non-zero background.

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;

for i = 1:rand_trials

    % d1 is distance between Mask 1 and image plane of lens
    % here 2.04 mm is the spacing between camera sensor
    % image plane and Mask 1
    d1=defocus_value-2.04;

    % d2 is distance between Mask 2 and image plane of lens
    % here 6.35 mm is the spacing between
    d2=f-6.35;

    % calculating pitch and positions relative to detector
    % for locating Mask1 away from focus
    pos1 = (defocus_value -d1)/(f+defocus_value);

    % position of Mask 2
    pos2=(d2+defocus_value)/(f+defocus_value);

    %calculating pitch values of Masks on detector
    pd1=(p1*defocus_value)/d1; % for mask1
    pd2=(p2*defocus_value)/d2; % for mask2
    pitch1=pd1/p;
    pitch2=pd2/p;


    pitch = [pitch1 pitch2];
    pos=[pos1 pos2];

    % Mask leaakge based on experimental mask transmission
    % values
    leakage1=0.12;%masks open part leakage of 12%
    leakage=0.00;%masks black part no leakage

    % fill factor of mask1 and mask2 , 50%
    through=[0.5  0.5];

```

```
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_%.3g_pitch_%.3g_%.3g.mat',directory,pos(1),pos
            (2),through(1),through(2),pitch(1),pitch(2));
127
```

```
    %%choose blur type on masks
     psfblur='fourierpsf';
    %psfblur='false' for no blur
    %psfblur='fourierpsf' for adding blur based on fourier filter


     varargin={'upsample',upsamp,'mask_pos',pos,'throughput',through,'maskpitch', pitch,'defocus',
          lens_defocus,'psfblur',psfblur,'leakage',leakage,'leakage1',leakage1};

    N=32;%size of the scene (NxN)
    K=8;%size of detector (KxK)

    % call function to generate system matrix, with point by point calibration.
    %selecting display as 'false' to prevent figure display while generating matrix
    H(:,:,i) = struct_projection(N, 'display',false,...
         'compression', N/K, varargin{:});

    % system_mat_name=sprintf('%s_defocus_%d.mat',date,lens_defocus);% file
    % name to save system matrix 'H'

    %     system_mat_name = ['trial' num2str(i) '_default_matrix.mat']
     save(['condnum_df_' num2str(defocus_value) 'normalization_on.mat'],'H','rand_trials')

end
```

Listing D.2: struct_projection function

```
function H = struct_projection(N,varargin)

defopt.mask_pos = [.1  .9];
defopt.compression = 4;
defopt.throughput = 0.5;
defopt.defocus = 2;
defopt.display = true;
defopt.upsample = 4;

% Sets pitch equal to reconstruction resolution
defopt.maskpitch = 1;
defopt.psfblur='false';
defopt.leakage1=0;%mask clear part
defopt.leakage=0;%mask dark part

opt = matlab_options(varargin, defopt);

compression = opt.compression;
throughput = opt.throughput;
dd = opt.defocus;
leakage = opt.leakage;
leakage1 = opt.leakage1;

upsample = opt.upsample;  % linear upsample factor

display = opt.display;

psfblur=opt.psfblur; % read blur type

% all physical distances are in mm


N=32;% scene size NxN

% location pitch - desired image-space reconstruction
% resolution in mm
p  = 0.324;

f = 35; % lens focal length in mm
D=25.4; %lens diameter in mm

maskcount = length(opt.mask_pos);

```

```
%for locating Mask1 away from focus
d1 = dd - opt.mask_pos(1)*(f+dd);

% use the statement below for locating Mask1 before the
% focus (i.e. on the same side of focus as Mask 2)

%%% 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
maskpitch = [maskpitch ; ones(3 - length(maskpitch),1)];

% make mask pitch (pd) equal to reconstruction pitch (p)
pd = maskpitch * p; % mask pitch in sensor plane

%% perform geometrical calculations

p1 = pd(1)*d1/dd; % physical mask pitch---- mm real mask
p2 = pd(2)*d2/dd;
p3 = pd(3)*d3/dd;

% this is the pitch of the points in the aperture,
% a larger defocus (dd) gives a smaller pitch
pL = p*f/dd;

fprintf('mask 1: d1 = %.1f, p1 = %.3f\n', d1, p1);
fprintf('mask 2: d2 = %.1f, p2 = %.3f\n', d2, p2);
fprintf('mask 3: d3 = %.1f, p3 = %.3f\n', d3, p3);

Sf = p * (N-1) / (1 + dd/f);  % full "image" width at focus (mm)
Sd = Sf * (1 + dd/f); % image width at sensor (at z = dd)
S1 = Sf * (1 + d1/f); % at mask 1
S2 = Sf * (1 + d2/f); % at mask 2

S3 = Sf * (1 - d3/f); % at mask 3

Bd = D * dd/f; % blur width at sensor
B1 = D * d1/f;
B2 = D * d2/f;
B3 = D * d3/f;

%% define mask structures
```

```
m0.z = f; % aperture
m0.x = [-D/2:pL:D/2];
m0.y = m0.x;
[X,Y] = meshgrid(m0.x, m0.y);

%%## adding gaussian blur to model lens defocus

% 'blursize' indicates the size of the blur filter which is
% computed as per the defocus level above(bigger blur for a larger defocus).
blursize=size(X);

% the standard deviation of 12 is chosen to model lens
% defocus with Gaussian blur
m0.v=fspecial('gaussian',blursize(1),12);

m1.z = d1; % first mask
m1.x = [-(B1+S1)/2:p1:(B1+S1)/2];
m1.y = m1.x;
m1.v = double(rand(length(m1.x)) <= throughput(1));

% including leakage of dark part
leak = leakage*ones(length(m1.v));

% including leakage of clear part
leak1 = leakage1*ones(length(m1.v));
m1.v = max(m1.v.*(1-leak1),leak);
m2.z = d2; % second mask
m2.x = [-(B2+S2)/2:p2:(B2+S2)/2];
m2.y = m2.x;
m2.v = double(rand(length(m2.x)) <= throughput(2));

% including leakage of dark part
leak = leakage*ones(length(m2.v));

% including leakage of clear part
leak1 = leakage1*ones(length(m2.v));
m2.v = max(m2.v.*(1-leak1),leak);
m3.z = d3; % third mask
m3.x = [-(B3+S3)/2:p3:(B3+S3)/2];
m3.y = m3.x;
m3.v = double(rand(length(m3.x)) <= throughput(3));
% including leakage of dark part
leak = leakage*ones(length(m3.v));
% including leakage of clear part
leak1 = leakage1*ones(length(m3.v));
m3.v = max(m3.v.*(1-leak1),leak);

%% perform calculations
H = []; Hcrop=[];
% range and sampling for masks in sensor plane
xx = [-Sd/2:pd/upsample:Sd/2];
% range of point locations (mapped to image space)
xlist = [-Sd/2:p:Sd/2];
progress(1, length(xlist)^2, 1, 1);
ind = 1;

countval=length(xlist);

for xcount=1:countval
    for ycount=1:countval
        x=xlist(xcount);
        y=xlist(ycount);
        progress(ind); ind = ind + 1;
        [mout, ml] = overlay_masks({m0, m1, m2}, ...
            [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);
```

```
186
187         if display
188             if gcf ~= 2; figure(2); end
189             subplot(1,2,1);
190             imagesc(high_res); axis image; colormap gray;
191             subplot(1,2,2);
192             imagesc(low_res); axis image; colormap gray;
193             drawnow;
194         end
195
196         H = [H low_res(:)]; % build the system matrix
197
198     end
199 end
200 if display
201   figure(3);
202   imagesc(H);
203   colormap jet;
204   colorbar;
205 end
```

Listing D.3: overlay_masks function

```
1  function [m_out, ml] = overlay_masks(mask_list, origin, ...
2      z, x, y,B1,B2,Bd,psfblur,D)
3  % project a list of masks to given z plane and overlay them
4  %   mask_list - list of mask structures
5  %   origin - [x, y] coordinates of projection origin
6  %   z      - z-plane to project to
7  %   x, y   - x and y coordinate lists to interpolate onto
8
9  if nargin < 4
10     x = ml{1}.x;
11     y = ml{1}.y;
12 end
13
14 % project the masks to common z plane
15 if nargin > 1
16     for ind = 1:length(mask_list)
17         ml{ind} = project_mask(mask_list{ind}, origin, z);
18     end
19 else
20     ml = mask_list;
21 end
22
23 % build the combined mask structure
24 m_out = ml{1};
25 m_out.x = x;
26 m_out.y = y;
27 [X,Y] = meshgrid(x,y); % gridded coordinates for output mask
28 m_out.v = ones(size(X));
29
30 for ind = 1:length(ml)
31     [Xm, Ym] = meshgrid(ml{ind}.x, ml{ind}.y); % native projected coordinates
32     if 0
33         fprintf('ind = %i\n', ind);
34         fprintf('size(Xm) = [%i, %i]\n', size(Xm,1), ...
35             size(Xm,2));
36         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
```

```
    switch (psfblur)

        case 'fourierpsf' % adding blur on masks
            switch (ind)
                case 1
                    m_out.v = m_out.v .* interp2(Xm, Ym, ...
                        ml{ind}.v, X, Y, '*linear', 0);


                case 2 %blur on mask 1
                    [sz1,sz2]=size((interp2(Xm, Ym, ...
                        ml{ind}.v, X, Y, '*linear', 0)));

                    %create filter in fourier domain
                    blur_radius1=(sz1*((D-B1)/D))/2;
                    % compute blur radius at chosen defocus
                    % level
                    [rr cc] = meshgrid(1:sz1);
                    blurpsf1 = fftshift(sqrt((rr-round(sz1/2)).^2+(cc-round(sz1/2)).^2)<=
                        blur_radius1);
                    blur_filter1 = abs((ifft2(blurpsf1)));
                    %apply filter

                    filtered_mask =  ...
                                                real(ifft2(blurpsf1.*fft2(interp2(Xm, Ym, ml{ind
                                                    }.v, X, Y, '*linear', 0))));

                    %normalized the filtered mask to it can't amplify
                    filtered_mask = filtered_mask / max(filtered_mask(:));

                    m_out.v = m_out.v .* filtered_mask;
                    m_out.v(m_out.v<0)=0;

                case 3 %blur on mask 2

                    [sz1,sz2]=size((interp2(Xm, Ym, ml{ind}.v, X, Y, '*linear', 0)));

                    %create filter in fourier domain
                    blur_radius2=(sz1*((D-B2)/D))/2; % compute blur radius at chosen defocus
                        level
                    [rr cc] = meshgrid(1:sz1);
                    blurpsf2 = fftshift(sqrt((rr-round(sz1/2)).^2+(cc-round(sz1/2)).^2)<=
                        blur_radius2);
                    %apply filter
                    blur_filter2 = abs(ifft2(blurpsf2));

                    filtered_mask = ...
                                                real(ifft2(blurpsf2.*fft2(interp2(Xm, Ym, ml{ind
                                                    }.v, X, Y, '*linear', 0))));

                    filtered_mask = filtered_mask / max(filtered_mask(:));
                    m_out.v = m_out.v .* filtered_mask;
                    m_out.v(m_out.v<0)=0;
            end

        case 'false' %no blur
            m_out.v = m_out.v .* interp2(Xm, Ym, ml{ind}.v, X, Y, '*linear', 0);

    end

end
```

Listing D.4: project_mask function

```

function m_out = project_mask(m_in, origin, new_z)
% project the input mask to a new z plane.  The projection is performed
% via rays emanating from the provided x-y origin in the z=0 plane

% mask attributes: z, v, x, y (optional)
```

```
% if y is provided, v should be 2d, otherwise 1d

m_out.z = new_z;
m_out.v = m_in.v;
x0 = origin(1);
m_out.x = (m_in.x - x0) * (m_out.z/m_in.z) + x0;
if length(origin) > 1
    y0 = origin(2);
    m_out.y = (m_in.y - y0) * (m_out.z/m_in.z) + y0;
end

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

 % call function to center PSF
[m_out.x m_out.y]=centering_atlocation(origin,m_out.x,m_out.y);

end
```

Listing D.5: centering_atlocation function

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

%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];
        M = subsref(im, S);
        S.subs{d} = i;
        imout = subsasgn(imout, S, sum(M, d));
    end
    if d < length(s) % only need to reassign if we're not done yet
        im = imout;
    end
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.

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.

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.

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.

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. $P\left(g \mid \theta\right)$

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.

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. $P(\theta)$

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.

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.

Acronyms

ADC analog-to-digital converter

AFSS Adaptive Feature Specific Spectrometer

AFSSI-C Adaptive Feature Specific Spectral Imaging-Classifier

CACTI Coded Aperture Compressive Temporal Imaging

CASSI Coded Aperture Snapshot Spectral Imaging

CCD Charge-Coupled Device

CMOS Complementary Metal–Oxide–Semiconductor

CT Computed Tomography

DISP Duke Imaging and Spectroscopy Program

DMD Digital Micro-Mirror Display

FPA focal-plane array

FTS Fourier Transform Spectrometer

LAR Least Angle Regression

LCOS Liquid Crystal on Silicon

LENS Laboratory for Engineering Non-Traditional Sensors

LS least squares

MAP Maximum A Posteriori

MRI Magnetic Resonance Imaging

MSE Mean Squared Error

OSA Optical Society of America

PCA Principal Component Analysis

PET Positron Emission Tomography

PSF point-spread function

RIP restricted isometry property

SAR Synthetic Aperture Radar

SCOUT Static Computational Optical Undersampled Tracker

SLM Spatial Light Modulator

SNR signal-to-noise ratio

SPECT Single-Photon Emission Computed Tomography

SWAP-C size, weight and power-cost

Symbols

$\mathbf{A}$ The product of the sensing matrix and the representation matrix $\mathbf{A} = \mathbf{H\Psi}$

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

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] T. M. Cover and J. A. Thomas, *Elements of information theory.* John Wiley & Sons, 2012.

[4] 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.

[5] I. Noor, O. Furxhi, and E. L. Jacobs, "Compressive sensing for a sub-millimeter-wave single pixel imager," in *SPIE Defense, Security, and Sensing.* International Society for Optics and Photonics, 2011, pp. 80 220K–80 220K.

[6] 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.

[7] M. J. Golay, "Multi-slit spectrometry," *JOSA*, vol. 39, no. 6, pp. 437–444, 1949.

[8] E. E. Fenimore and T. Cannon, "Coded aperture imaging with uniformly redundant arrays," *Applied optics*, vol. 17, no. 3, pp. 337–347, 1978.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] T.-H. Tsai and D. J. Brady, "Coded aperture snapshot spectral polarization imaging," *Applied optics*, vol. 52, no. 10, pp. 2153–2161, 2013.

[14] 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.

[15] 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.

[16] H. H. Barrett and K. J. Myers, *Foundations of Image Science*. John Wiley & Sons, 2013.

[17] J. Radon, "1.1 über die bestimmung von funktionen durch ihre integralwerte längs gewisser mannigfaltigkeiten," *Classic papers in modern diagnostic radiology*, vol. 5, 2005.

[18] "The Nobel Prize in Physiology or Medicine, 1979," https://www.nobelprize.org/nobel_prizes/medicine/laureates/1979/perspectives.html, accessed: 2016-08-22.

[19] 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.

[20] C. Chen and J. Huang, "Compressive sensing mri with wavelet tree sparsity," in *Advances in neural information processing systems*, 2012, pp. 1115–1123.

[21] W. S. Boyle and G. E. Smith, "Charge coupled semiconductor devices," *Bell System Technical Journal*, vol. 49, no. 4, pp. 587–593, 1970.

[22] 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

[23] 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.

[24] K. L. Moore, "Spectrometer with electronic readout," Mar. 27 1979, uS Patent 4,146,332.

[25] 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.

[26] 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.

[27] 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.

[28] 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.

[29] 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.

[30] D. L. Donoho, "Compressed sensing," *IEEE Transactions on information theory*, vol. 52, no. 4, pp. 1289–1306, 2006.

[31] 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.

[32] 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.

[33] 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.

[34] "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.

[35] S. Foucart and H. Rauhut, *A mathematical introduction to compressive sensing*. Springer, 2013, vol. 1, no. 3.

[36] M. Gehm, "Calibration–an open challenge in creating practical computational- and compressive-sensing systems," 2013.

[37] E. J. Candès and M. B. Wakin, "An introduction to compressive sampling," *IEEE signal processing magazine*, vol. 25, no. 2, pp. 21–30, 2008.

[38] 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.

[39] M. Harwit and N. J. Sloane, *Hadamard transform optics*. Elsevier, 2012.

[40] J. W. Goodman, *Introduction to Fourier optics*. Roberts and Company Publishers, 2005.

[41] 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.

[42] S. P. Davis, M. C. Abrams, and J. W. Brault, *Fourier transform spectrometry*. Academic Press, 2001.

[43] J. W. Goodman, *Statistical optics*. John Wiley & Sons, 2015.

[44] M. H. Tai and M. Harwit, "Fourier and hadamard transform spectrometers: a limited comparison," *Applied optics*, vol. 15, no. 11, pp. 2664–2666, 1976.

[45] I. Jolliffe, *Principal component analysis*. Wiley Online Library, 2002.

[46] J. Shlens, "A tutorial on principal component analysis," *arXiv preprint arXiv:1404.1100*, 2014.

[47] P. K. Poon, W.-R. Ng, and V. Sridharan, "Image denoising with singular value decompositon and principal component analysis," December 2009.

[48] 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.

[49] 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.

[50] C. E. Shannon, "Communication in the presence of noise," *Proceedings of the IRE*, vol. 37, no. 1, pp. 10–21, 1949.

[51] J. G. Proakis and D. G. Manolakis, *Introduction to digital signal processing*. Prentice Hall Professional Technical Reference, 1988.

[52] 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.

[53] 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.

[54] 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.

[55] C. C. Aggarwal, *Data streams: models and algorithms*. Springer Science & Business Media, 2007, vol. 31.

[56] 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.

[57] 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.

[58] 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.

[59] A. Neumaier, "Solving ill-conditioned and singular linear systems: A tutorial on regularization," *SIAM review*, vol. 40, no. 3, pp. 636–666, 1998.

[60] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 267–288, 1996.

[61] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An introduction to statistical learning*. Springer, 2013, vol. 6.

[62] 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.

[63] S. Evladov, O. Levi, and A. Stern, "Progressive compressive imaging from radon projections," *Optics express*, vol. 20, no. 4, pp. 4260–4271, 2012.

[64] 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.

[65] Y. Kashter, O. Levi, and A. Stern, "Optical compressive change and motion detection," *Applied optics*, vol. 51, no. 13, pp. 2491–2496, 2012.

[66] 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.

[67] 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.

[68] H. Rauhut, "Circulant and toeplitz matrices in compressed sensing," *arXiv preprint arXiv:0902.4394*, 2009.

[69] J. Romberg, "Compressive sensing by random convolution," *SIAM Journal on Imaging Sciences*, vol. 2, no. 4, pp. 1098–1128, 2009.

[70] 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.

[71] 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.