

COHERENT IMAGING VIA TEMPORAL HETERODYNE
AND SPATIAL SHEARING METHODS

by

Ryan Moore Galloway

A thesis submitted in partial fulfillment
of the requirements for the degree

of

Master of Science

in

Optics and Photonics

MONTANA STATE UNIVERSITY
Bozeman, Montana

April, 2018

©COPYRIGHT

by

Ryan Moore Galloway

2018

All Rights Reserved

ACKNOWLEDGEMENTS

I would like acknowledge Dr. Zeb Barber, for openly sharing his knowledge and guidance throughout my Masters Degree. I would like to thank my mentors at AFRL, Dr. David Rabb and Dr. Jason Stafford, for providing such a fun and insightful summer of research.

Thank you to all the folks at Spectrum Lab for helping me along the way, I really appreciate all of your hard work.

Thank you to all the professors who helped steer me into the light, without you I would not have found myself in this exciting field of study. Thank you to my fellow students for your friendship and support.

Finally thank you to my parents, brother, girlfriend, and dogs for helping me along the way. I really could not have made it this far without you all.

Funding Acknowledgment

This work was funded by the AFRL RYMM Laser Radar Branch through Leidos ALTAR subcontract P010169067, and through an ATR Center internship.

TABLE OF CONTENTS

1. INTRODUCTION	1
Coherent Imaging Background.....	1
Motivation	2
Outline	3
2. FRAUNHOFER PROPAGATION	5
Light as a Field	5
Fresnel Approximation	6
3. DIGITAL HOLOGRAPHY.....	8
Spatial Heterodyne Digital Holography	8
Experimental Setup for Spatial Heterodyne Holography	10
Quadratic Focusing and Image Processing.....	13
Temporal Heterodyne Holography.....	15
How a Photonic Mixing Device Works	15
Using a PMD for Temporal Heterodyne	18
Hologram Processing	22
4. LINEAR FREQUENCY MODULATED LIDAR.....	26
Linear Frequency Modulated Lasers	26
LFM Lidar with a Flat or Point Target	27
Range and Cross Range Imaging with an LFM Lidar	29
Experimental Demonstration of Imaging with an LFM Lidar	32
5. SHEARED IMAGING THEORY	38
Sheared Coherent Interferometric Photography.....	38
A Better Way to Integrate Shears	40
6. SPATIAL SHEARING LIDAR SIMULATION	48
Creating a Synthetic Target	48
Simulated Standard Digital Holographic Output Image.....	50
Generating Synthetic Spatial Shears	51
Phase Closure Algorithm.....	56
Phase Solution via Maximum Likelihood Eigenvector Method	58
Amplitude Solution.....	61

TABLE OF CONTENTS – CONTINUED

Reconstructing an Image.....	64
Phase Closure Demonstration.....	66
This Reconstruction Algorithm is Very Sensitive.....	69
Incorrect Sign on Phase Closure Loops	70
Arbitrary Conjugate Shears	70
Incorrect Diagonal in the Covariance Matrix	71
Incorrect Shear Size.....	72
7. SPATIAL SHEARING LIDAR EXPERIMENT	73
Spatial Shearing Experimental Setup	75
Demultiplexing Spatial Shears using Frequency Offsets	76
Calibration of Frequency Peaks	80
Image Processing.....	82
Using the Calibration Data to Demultiplex the Spatial Shears.....	83
Compressing the Dataset from 3D to 2D Shears	84
First Try: Bar Target Data Processing	87
Link Budget Calculation.....	88
Low SNR Spatial Shearing Reconstruction Simulation	92
Mirror Target Data Processing.....	93
Confidence That the Shears Contain Image Information	94
Finding the Correct Phase Closure Loops	96
Zero Out DC Peaks in Shears	98
Removing Padding Around Shears	99
Finding the Correct Conjugations for the Shears	100
Validations of the Mirror Image.....	103
Test 1: Same Pattern over Multiple Datasets	103
Test 2: Geometry of the Target, and Spatial Interference Peaks	103
Test 3: Removing Shears from the Covariance Matrix	104
8. CONCLUSION	106
Is Spatial Shearing Lidar Practical?	106
Retroreflective AFBT at 3 meters.....	106
Diffuse AFBT at 3 meters.....	107
Diffuse AFBT at 1 km	108
Successful Experiments	109
Temporal Heterodyne Holography	109
Temporal Frequency Demultiplexing of Spatial Shears	110
Spatial Searing Lidar Verification	110

TABLE OF CONTENTS – CONTINUED

Future Work.....	111
Temporal Shearing Self-Referencing Lidar	111
Combining Shearing Methods.....	111
Fresnel Turbulence Simulation.....	112
Computing Digital Shears for Self-Referencing Lidar.....	112
REFERENCES CITED.....	114
APPENDICES	117
APPENDIX A : LFM Lidar Processing Script.....	118
APPENDIX B : Spatial Shearing Lidar Simulation.....	121
APPENDIX C : Spatial Shearing Lidar Experimental Data Processing	126
APPENDIX D : Matlab Functions.....	131

LIST OF FIGURES

Figure	Page
1.1 Standard Vs. Self-Referencing Lidar	2
3.1 Spatial Heterodyne Holography Schematic	10
3.2 Lab Demonstration of Spatial Heterodyne Holography	11
3.3 Digital Hologram	11
3.4 Fourier Transform of Hologram	12
3.5 Quadratic Phase Screen Correction	14
3.6 Photonic Mixing Device Phase Determination	16
3.7 Temporal Heterodyne Holography Schematic	19
3.8 Temporal Heterodyne Holography Experiment	19
3.9 Phase Measurement with a PMD	23
3.10 Intensity Measurement with a PMD	24
3.11 Temporal Heterodyne Holography Output Image	24
4.1 Linear Frequency Chirp	26
4.2 Flat Target Example	29
4.3 Target With Depth	30
4.4 LFM Lidar Schematic	35
4.5 LFM Lidar Spectrum	36
4.6 LFM Lidar Image	37
5.1 Shifted Speckle	39
5.2 Largest Eigenvalue for Correct Phase	46
6.1 Digital Bar Target	49
6.2 Simulated Digital Holographic Image	50
6.3 Definition of Shears	51
6.4 Diagram of One Spatial Shear	54

LIST OF FIGURES – CONTINUED

Figure	Page
6.5 Simulated Shear in x Direction	56
6.6 Phase Closure Loops	57
6.7 Rastering Shears	59
6.8 Amplitude Solution	61
6.9 Spokes and Connector Shears.....	63
6.10 Simulated Sheared Lidar Reconstruction.....	65
6.11 Failed Reconstruction	67
6.12 Phase Closure Comparison	68
6.13 Phase Corrected Reconstruction.....	69
6.14 Incorrect Phase Loop	70
7.1 Spatial Shearing Experimental Setup.....	73
7.2 Spatial Shearing Experiment Photo.....	74
7.3 Spatial Shear Transmitter.....	75
7.4 Time Delaying the Transmitters.....	77
7.5 Temporal Power Spectrum of Transmitters	78
7.6 Maximum Target Depth	80
7.7 Calibration Data.....	81
7.8 Calibration Peak Registration	82
7.9 Raw Data Format	83
7.10 Demultiplexing Spatial Shears.....	84
7.11 Diagram of Speckle Shifting.....	85
7.12 Registered Frames for Shear Compression	86
7.13 Compressed Shear	87
7.14 Reconstructed “Image” of an AFBT.....	88

LIST OF FIGURES – CONTINUED

Figure	Page
7.15 Reflection Divergence from Retro-Reflective Fabric.....	90
7.16 Simulating Reconstruction With Low SNR	92
7.17 Four Mirror Target Arrangement	94
7.18 Shear Spatial Frequency Content	95
7.19 Phase Closure With Simulated Data.....	97
7.20 Phase Closure With Real Data.....	98
7.21 Removing DC Peaks From Shears	99
7.22 Removing Padding from Shears.....	100
7.23 Output Mirror Images from Nine Different Shots	101
7.24 Four Mirror Reconstructed Image.....	102
7.25 Quadratic Phase Check	103
7.26 Removing Diagonal Shears from Covariance Matrix.....	104

ABSTRACT

Atmospheric turbulence rapidly decreases image quality at long ranges. Here multiple coherent imaging methods are discussed that lead to a new type of active imaging system, which may help mitigate the effects of atmospheric turbulence. This is accomplished via a self-referencing, linear frequency modulated laser signal, where the signal is both offset in transmitter location (spatial shearing), and is demultiplexed in the temporal frequency domain using unique time delays for each transmitter (temporal heterodyne). Spatial shearing allows one to capture a spatial derivative of the object's spatial frequency content, which if properly "integrated" can be used to reconstruct an atmospheric phase-aberration-corrected image of the object. The system is illustrated from the starting point of temporal digital holography methods, and builds up to the self-referencing scheme. Various coherent imaging methods and situational parameters are compared.

INTRODUCTION

Coherent Imaging Background

We interact with light every day, but the light we know is the time and wavelength averaged version of its true self. Coherent imaging is based off of the idea that light is a complex electromagnetic field, and that even though we can only measure intensity variations with detectors, we can still solve for the amplitude and phase of that light if we employ the tools of coherent imaging. In order to get the phase information from the light, we need to find a way to convert that information into intensity variations, in space or time, which we can then interpret to back out the phase of the light. [1,2] This is commonly done with what is called a Local Oscillator (LO), which is a well behaved reference beam with which we interfere with the light, the phase of which, we want to know.

This basic idea of interfering a reference beam with a beam that contains target information is common to all coherent imaging techniques. By measuring both the amplitude and phase of the light, one has much more freedom in post processing steps for converting the optical field from a target for imaging or other optical measurements. For example, if we have the phase, we can correct for different known aberrations, as well as focusing errors. We can even unwrap the phase to construct a three dimensional version of the object. Dennis Gabor was one of the first people to employ coherent imaging in the form of Holography, which will be outlined in Chapter 3. [1,3]

Motivation

The reason coherent imaging is useful is that it gives us access to the phase of the light scattered from an object. This phase information tells us exactly how the light (as a field) scattered off of the object, and one can reconstruct an image of that object from that information. [1]

Atmospheric turbulence can cause problems in long range imaging applications. The turbulent cells in the atmosphere scramble the optical phase of light scattered from an object, and at long ranges reaches a point where the phase is not representative of the object, and the images become blurred. The goal of this thesis is to introduce coherent imaging techniques, and then provide the theory and experimental validation of a new coherent imaging technique that has the potential to mitigate the effects of atmospheric turbulence on the phase of the light to be measured. [2, 4]

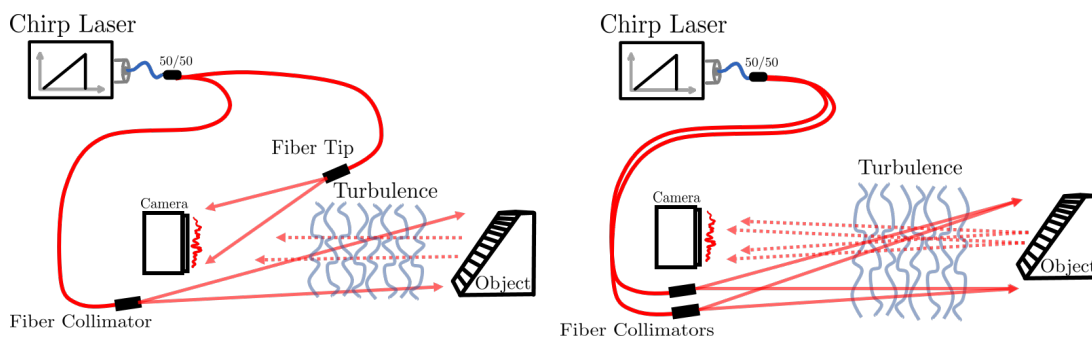


Figure 1.1: The left diagram is a “typical” lidar setup, where the transmit beam and returned light both encounter turbulence. This is then mixed with a local oscillator that did not encounter any turbulence. This can be compared to the self-referencing case on the right, where two transmitters are used and are spatially offset from each other. These two illumination beams see the same turbulence on the way to the object, and again on the way back to the detector. Since the returned signals are mixed with each other on the detector, the phase contributions of the turbulence will drop out yielding a signal that contains only object information.

Figure 1.1 highlights the difference between “standard” lidar and self-referencing lidar. The basic idea is that in self-referencing lidar, two transmitters are used to illuminate the object. The spatial shift between these transmitters leads to a spatial shift of the returned light on the detector. The main difference is that in self-referencing lidar, the scattered light is being interfered with a shifted copy of itself, which “sees” the same atmospheric phase aberrations as the other path. When two optical signals are mixed, their phases are subtracted from each other, and any common atmospheric phase term that is same on both beams will simply drop out. This does mean that the size of the shear sets smallest length scale of turbulence features that are canceled. So as long as the shears are small, the turbulence phase should mostly cancel in self-referencing lidar. This theory was originally proposed as a simpler alternative to adaptive optics, where atmospheric aberrations are corrected in the measurement process directly instead of with deformable mirrors. [5]

Outline

Chapter 2 will briefly cover free space diffraction theory from the perspective of Fourier optics. Chapter 3 will go over common methods of retrieving the phase of the light, and how the phase information is converted into intensity variations. Chapter 4 will introduce chirp ranging methods for lidar, where optical chirps are used to find the distance to objects, and reconstructing range-crossrange images of the target from this data. Chapter 5 will introduce the theory behind a new technique called self-referencing lidar, that may be useful in mitigating atmospheric phase turbulence in long range coherent imaging. [6] In Chapter 6, a simulation of self-referencing lidar is presented that employs spatial shears to understand the basics of processing this new type of coherent imaging data. Chapter 7 will go over the experimental demonstration of self-referencing lidar, with some slight processing modifications, as

well as validation of the output image. The last chapter will be a conclusion with a brief overview of what was learned, and a discussion of future areas of research.

FRAUNHOFER PROPAGATION

Most of the math presented in this thesis will be based off the foundations of Fourier optics. Fourier optics is built off of approximations of the full electromagnetic vector wave propagation, where the mathematics can be too complicated for general use. Linear system theory used for electronic system analysis utilizes the temporal frequency response of the system. Linear optical systems are analogous to linear electrical systems, where the spatial frequency response is used to characterize the system instead of the temporal frequency response. Fourier optics is a powerful tool for characterizing and designing optical systems based off of the desired spatial frequency response. [1]

Light as a Field

While there are many tools in Fourier optics, the one used the most in this thesis is the propagation of a light field from one plane to another plane. Start by defining a plane wave as a complex field in the x - y - z coordinate system.

$$U(x, y, z) = Ae^{i\vec{k} \cdot \vec{r}}, \quad (2.1)$$

where A is the amplitude of the plane wave, and

$$\vec{k} = \frac{2\pi}{\lambda}(k_x\hat{x} + k_y\hat{y} + k_z\hat{z}) \quad (2.2)$$

is the wavevector which defines the direction of propagation in terms of the direction cosines. The plane wave travels in a direction that is an angle $\cos^{-1}(k_x)$ from the x axis, $\cos^{-1}(k_y)$ from the y axis, and $\cos^{-1}(k_z)$ from the z axis. For the following it will be assumed that the plane wave is propagating mostly in the z direction.

Fresnel Approximation

The Fresnel diffraction integral is one approximate method by which light is propagated from one plane to another mathematically. It has the following approximations built into it already. The first is that the distance to the observation plane is much larger than the wavelength of the light being propagated. The second is that the detector plane is in the “near field” which is defined as being many wavelengths away from the aperture, but also not so far as to be in the far field, which is defined in terms of the Fraunhofer approximation. This integral also assumes that one is starting in the $\xi \eta$ plane, and is trying to find the optical field in the $x y$ plane that is a distance z away. [1]

$$U(x, y) = \frac{e^{ikz}}{i\lambda z} e^{i\frac{k}{2z}(x^2+y^2)} \iint_{-\infty}^{\infty} \left\{ U(\xi, \eta) e^{i\frac{k}{2z}(\xi^2+\eta^2)} \right\} e^{-i\frac{2\pi}{\lambda z}(x\xi+y\eta)} d\xi d\eta \quad (2.3)$$

One can recognize the integral portion of this to be the Fourier transform of the complex input field times a quadratic phase exponential term. If one wanted to find an approximation that would make the field at the detector plane look exactly like a Fourier transform of the aperture, they might look for a distance z that is so far away that the quadratic phase term looks like a flat phase across the aperture. That would be at a distance

$$z \gg \frac{k(\xi^2 + \eta^2)}{2}, \quad (2.4)$$

which would greatly simplify the Fresnel diffraction integral,

$$U(x, y) = \frac{e^{ikz}}{i\lambda z} e^{i\frac{k}{2z}(x^2+y^2)} \iint_{-\infty}^{\infty} \left\{ U(\xi, \eta) \cancel{e^{i\frac{k}{2z}(\xi^2+\eta^2)}} \right\} \overset{1}{e^{-i\frac{2\pi}{\lambda z}(x\xi+y\eta)}} d\xi d\eta. \quad (2.5)$$

This yields the Fraunhofer diffraction integral,

$$U(x, y) = \frac{e^{ikz} e^{i\frac{k}{2z}(x^2+y^2)}}{i\lambda z} \iint_{-\infty}^{\infty} U(\xi, \eta) e^{-i\frac{2\pi}{\lambda z}(x\xi+y\eta)} d\xi d\eta. \quad (2.6)$$

For this thesis I will define the following convention, when light is being propagated from one plane to another, the Fraunhofer operator will be used and is denoted as $\mathcal{F}$. This is mathematically identical to a Fourier transform denoted as $\mathcal{F}$, but with the correct multiplicative phases, and the following change of variables,

$$f_\xi = \frac{x}{\lambda z} \quad \text{and} \quad f_\eta = \frac{y}{\lambda z}. \quad (2.7)$$

This relationship is as follows,

$$U(x, y) = \frac{e^{ikz} e^{i\frac{k}{2z}(x^2+y^2)}}{i\lambda z} \mathcal{F}\{U(\xi, \eta)\} \Big|_{f_\xi, f_\eta} = \mathcal{F}\{U(\xi, \eta)\} \quad (2.8)$$

So the light that was in the initial plane, has undergone Fraunhofer propagation to the “far field”, where the quadratic phase term is unity across the entire observation aperture, and is now a representation of the two dimensional Fourier transform of the original optical field. That is to say that the field at the detector plane is the spatial frequency content of the input plane. These concepts will be used repeatedly in this thesis, and it is good to have a firm grasp on them before moving forward. The next chapter will introduce the basics of coherent imaging with digital holography, where these concepts will first be applied.

DIGITAL HOLOGRAPHY

Spatial Heterodyne Digital Holography

Digital Holography is a coherent imaging technique that retrieves both intensity and phase information from the light that was scattered off an object. It accomplishes this by illuminating the object with a coherent light source (Transmit beam), and illuminating the sensor with a portion of the same coherent laser source (Local Oscillator or LO). When the scattered light from an object is mixed with the local oscillator on a CCD, an interference pattern is recorded. This interference pattern can be processed to get an image of the object, in both amplitude and phase. [1]

Optical detectors can only measure the intensity of light, so to retrieve both the amplitude and phase of an optical field, the phase must be somehow converted into an intensity variation for the sake of recording. This is commonly done with interference, and in the case of spatial heterodyne holography, an angular offset is used to demultiplex the object information from the rest of the interference terms. [1, 7] To measure the amplitude and phase of the light scattered from an object,

$$E_{\text{obj}}(x, y) = a(x, y)e^{i\phi(x, y)}, \quad (3.1)$$

one can interfere it with a known field,

$$E_{\text{Ref}}(x, y) = A(x, y)e^{i\psi(x, y)} \quad (3.2)$$

where the interference between them is an intensity, that is the sum of their fields

absolute value squared,

$$I(x, y) \propto |A(x, y)|^2 + |a(x, y)|^2 + 2|A(x, y)||a(x, y)| \cos[\psi(x, y) - \phi(x, y)]. \quad (3.3)$$

The first two terms of this expression contain information only about the intensity of the two fields, but the third term contains information about their relative phase, and thus all of the information about the object field can be recorded. The phase information is there, but how does one separate the first two terms from the mixing terms where the phase information is contained? That is where spatial heterodyne is utilized. We need to attach a spatial frequency to the mixing term that will allow it to be separated from the first two terms when examined in the spatial frequency domain. [1, 8] Spatial heterodyne holography utilizes a small angle between the LO and scattered light, which leads to interference fringes across the hologram. In practice, this is accomplished by illuminating the sensor with the LO at some off-axis angle that does not get in the way of the scattered light coming back from the object. We can think of this as a modification to the third term in the intensity equation,

$$I(x, y) = |A(x, y)|^2 + |a(x, y)|^2 + 2|A(x, y)||a(x, y)| \cos \left[\vec{k} \cdot \vec{r} + \psi(x, y) - \phi(x, y) \right] \quad (3.4)$$

where the $\vec{k} \cdot \vec{r}$ is due to the angle between the central propagation directions of the two fields. The object information is stored in the modulation of these fringes, and they act like a spatial carrier frequency, that moves the image information away from the low spatial frequencies. When the hologram is spatially Fourier transformed, it shows the distribution of spatial frequencies in the hologram. This usually contains a DC component in the center of the Fourier space, which is the first two terms in Equation 3.4, and an output image in two opposing quadrants, which are the

conjugate exponential terms one would get by expanding the cosine term in Equation 3.4. This can be seen in Figure 3.4. Because the images are complex conjugates of each other, only one image is needed to retrieve all of the information. All it takes is to then crop down to one of the two images and that is the output image. The reason this works, is because the light that was scattered off of the object had previously undergone a Fourier transform spatially as it propagated to the sensor plane as was discussed in the previous chapter. The raw hologram is essentially a measurement of the spatial frequency content of the target, and to get an image back, one only needs to take another Fourier transform. [1, 8]

Experimental Setup for Spatial Heterodyne Holography

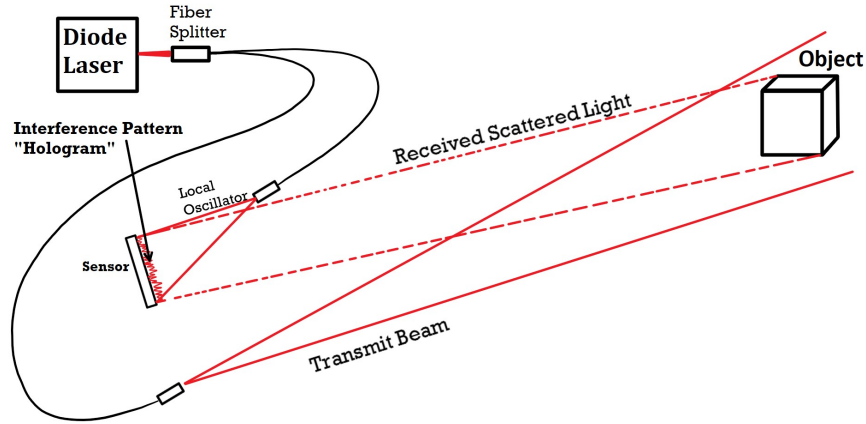


Figure 3.1: A sketch of a typical off-axis holography configuration. A laser beam is split, then half is used to illuminate the object (Transmit beam), and the other half (Local Oscillator), is used to illuminate the sensor. The scattered light from the object interferes on the sensor and because the two beams are at an angle with respect to each other, a fringe pattern appears, which contains the spatial frequency content of the image.

Figure 3.1 is a schematic of a standard off axis spatial heterodyne holography experiment. It should be noted that the angular difference that leads to a spatial offset frequency is due to the angle between the local oscillator and the returned

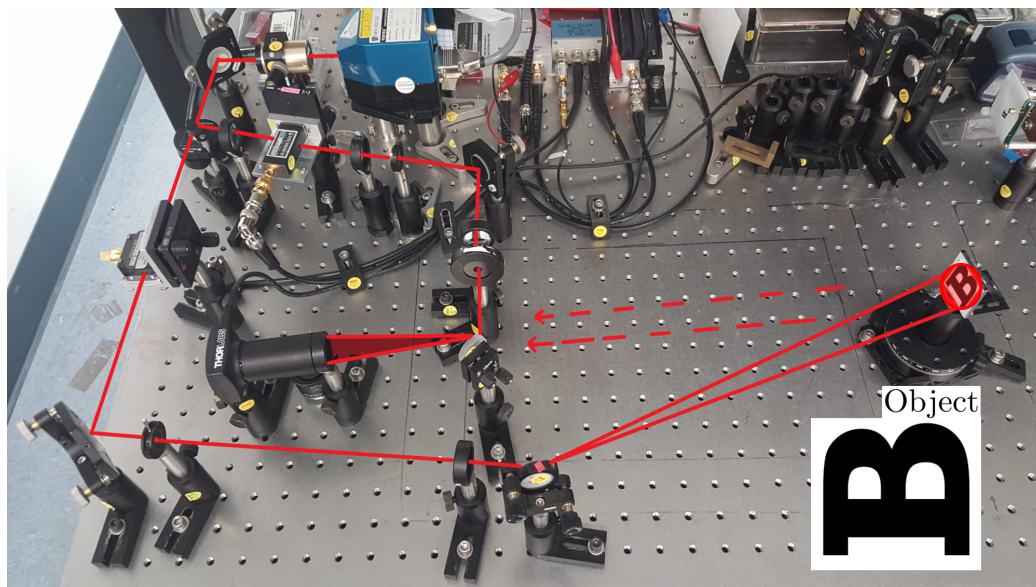


Figure 3.2: This is an image of the bench top setup that was used to demonstrate spatial heterodyne digital holography. It is arranged according to the schematic shown in Figure 3.1, with the only modifications being the use of an acousto-optic modulator (AOM) on each beam path, which will be needed in the next experiment.

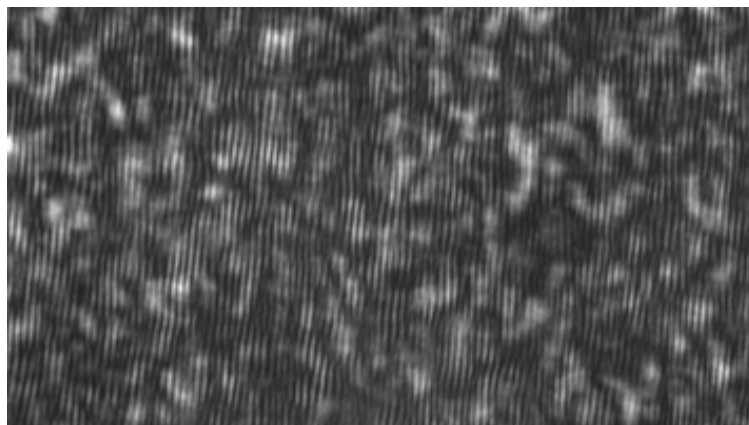


Figure 3.3: Digital hologram taken directly from the camera sensor in the off axis holography experiment depicted in Figure 3.1. The vertical fringes are quite small, and the speckle from the target is also very noticeable. It is clear that the speckle is modulated at a spatial frequency related to the angle between the LO and the scattered light from the target.

scattered light, not the illumination beam. In this arrangement, where no lens is used on the camera, a set of fringes are seen, with small variations that are the spatial frequency content of the object. In this case the fringes are vertical because the LO was displaced only in a horizontal direction relative to the scattered light from the object. Figure 3.3 is the intensity distribution that one would expect to

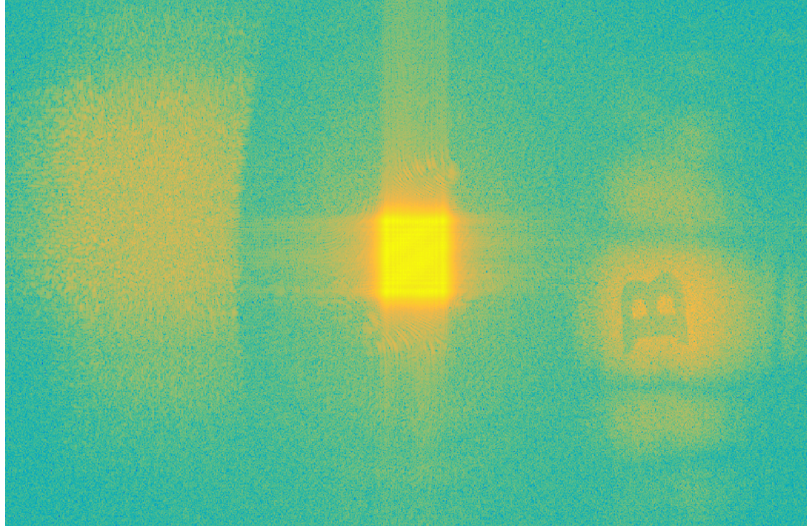


Figure 3.4: This is the result of a two dimensional Fourier transform performed on the intensity pattern shown in Figure 3.3. The DC component is the bright square in the middle, which is essentially the first two terms in from Equation 3.4, and the image can be seen in the bottom right corner, with its conjugate image in the top left. The top left image is blurry because the quadratic focusing step applies a phase correction that will help one of the conjugate images, and make the other even more unfocused.

see in a standard off axis holography configuration. The speckle (spatial frequency content of the target) is modulated by the small vertical fringes, which act as a spatial frequency offset so that the image can be separated out from the DC components of the hologram in the Fourier transform plane as seen in Figure 3.4. The images seen are complex conjugates of each other, so they contain the same information.

One more way to think about all of this, is that the raw hologram shown in

Figure 3.3 is just a two dimensional representation of the spatial frequencies of the object. These spatial frequencies (the speckle) are mixed with the higher frequency vertical fringes to offset the image from DC once it is spatially Fourier transformed.

Quadratic Focusing and Image Processing

The reason that the upper left “image” in Figure 3.4 does not look the same as the image in the bottom right is because of a post processing step that was used to focus the output image. In the experimental setup, the LO and the scattered light that are combined onto the sensor have different wavefront shapes. A pinhole was used to clean up the LO spatially, and as a result it had a simple spherical wavefront. When the spherical wavefront is mixed with the wavefront of the scattered light from the target, a set of concentric rings of light and dark are created. To extract a focused image, with just a 2D Fourier transform, the two wavefronts should be planes, as a flat wavefront Fourier transformed form perfect points in an image. [8] The defocus can be corrected in post processing by multiplying the hologram by a phase correction function with the opposite shape. [1, 2] If one expands the cosine interference term from Equation 3.4 in terms of exponentials,

$$2 \cos \left[\vec{k} \cdot \vec{r} + \psi(x, y) + \phi(x, y) \right] = e^{i(\vec{k} \cdot \vec{r} + \psi(x, y) + \phi(x, y))} + e^{-i(\vec{k} \cdot \vec{r} + \psi(x, y) + \phi(x, y))} \quad (3.5)$$

one sees that multiplying it by a quadratic focus term,

$$\left(e^{i(\vec{k} \cdot \vec{r} + \psi(x, y) + \phi(x, y))} + e^{-i(\vec{k} \cdot \vec{r} + \psi(x, y) + \phi(x, y))} \right) \cdot \left(e^{i \frac{k}{2\Delta z} (x^2 + y^2)} \right) \quad (3.6)$$

will effect the two terms in a different way, one focusing and the other defocusing. Here $k = 2\pi/\lambda$ and Δz is the distance that the spherical wave has propagated. The reason one is focused and the other is defocused, is because they are complex conjugates of

one another, which means that the complex correction function is helping one of the conjugate images, and making the other worse because of the sign difference between them. Because it is difficult to measure the distance between the pinhole and the camera sensor on the LO path precisely, I used an algorithm to iteratively try different amounts of focus correction while monitoring the sharpness of the output image until an ideal focus was found. This mostly matters for the LO path, because the target is far enough away, that the points in the target can be closely approximated to be plane waves. Figure 3.5 shows an example of the real part of the quadratic correction function that is multiplied against the raw hologram before the Fourier transform. Another way to think about how this phase is used to focus the image is by recognizing that it is the same phase a lens would impart onto a complex optical field. [1,2]

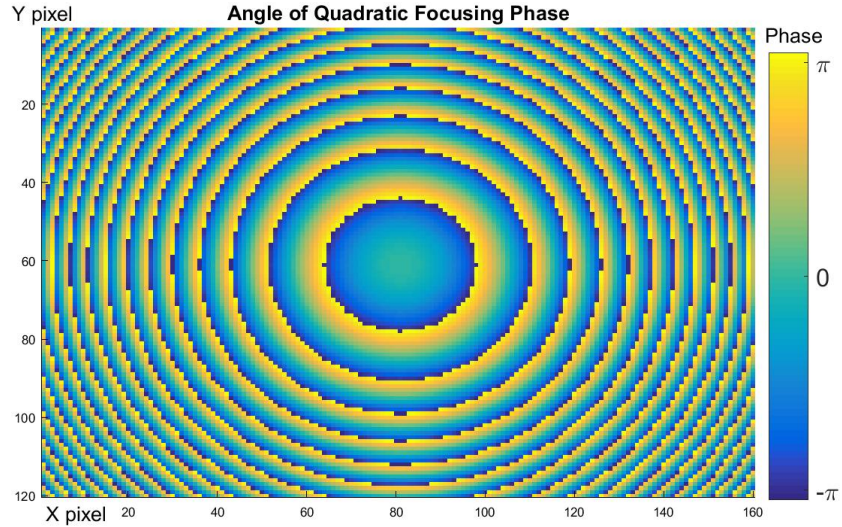


Figure 3.5: This is the phase correction that was multiplied point by point with the raw hologram. It corrects for both the curvature of a diverging spherical field on the LO, as well as for the quadratic term that is contained within the Fresnel propagation from the target. An algorithm is used to determine this phase screen by looking at the sharpness of the output image.

Spatial heterodyne holography was demonstrated to build a conceptual foundation for the modifications to come. It is important to remember that the image is offset from the DC component by using an angle between the LO field and the returned scatter field from the target. This leads to a spatial frequency offset of the image information, which is necessary for extracting the image in post processing. One major drawback to spatial heterodyne holography is that the effective resolution of the sensor is about a half in each dimension of what it would be in a normal imaging system, due to the cropping down to one quadrant after taking the Fourier transform. [8]

The next section will explore a new way to collect the complex field of the scattered light that does not involve a spatial frequency offset.

Temporal Heterodyne Holography

How a Photonic Mixing Device Works

Before diving into temporal heterodyne holography, I must first discuss how the sensor used in this experiment differs from a normal CCD array, and how it will allow the use of a frequency shift instead of a angle shift to get the image information away from the DC component in the temporal Fourier Domain.

A Photonic Mixing Device (PMD) is a special type of camera sensor where each element mixes the optical intensity signal with a reference electrical signal, and outputs the low pass integrated signal between the two. PMDs are used in active illumination 3D imaging systems, where each captured frame is a point cloud, representing the range of the objects in the scene. Their advantage is that they capture a whole frame at a time at very high frame rates, 160 fps, and have a very compact form factor. The main disadvantage is the limited range resolution and small maximum range. [9]

The time of flight (TOF) camera, for example using a PMD operates by illuminating the scene with LEDs amplitude modulated at 20 MHz. This light scatters off of the objects, and the returned light is imaged onto the PMD sensor with a lens. The square wave signal used to modulate the illumination light is also sent to each PMD pixel, as an electrical local oscillator (ELO). By electronically mixing the ELO signal with the photoelectrons generated by the returned optical signal, a phase difference is measured, which can be used to relate the time of flight of the light, to the depth of the scene. This mixing process can be seen schematically in Figure 3.6, where four quadrature measurements are made in order to extract the phase of the light. The voltage on each pixel of each frame on the PMD corresponds to the amplitude of the returned signal that is in phase with each of the quadrature ELO phases.

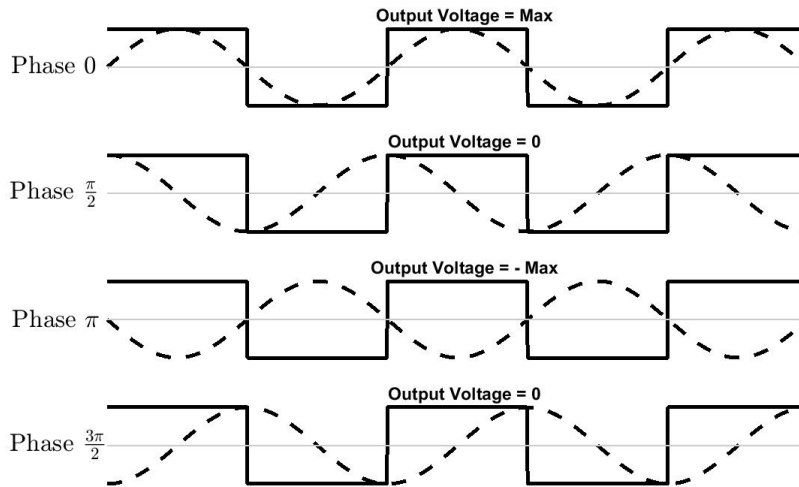


Figure 3.6: The square wave is the ELO, and the sinusoidal wave is the temporal beat note of the LO and scattered light being combined on the PMD pixel. The voltage output of a pixel on a PMD is a measure of how in phase the optical signal is with respect to different phases of the ELO.

The mixing eliminates any DC component of the optical field, and is a very efficient background light elimination method. So the only signal that the PMD pixel records, is the scattered light from the object that is amplitude modulated at 20 MHz. [9] The PMD measures a phase difference between the ELO and the amplitude modulated light that was scattered from an object and imaged onto the PMD array,

$$\Delta\phi = \phi_{\text{obj}} - \phi_{\text{ELO}}. \quad (3.7)$$

One can define a distance to that object in terms of the round trip time delay 2τ , the speed of light c , and the frequency of the amplitude modulations of the illumination beam ν_{ELO} .

$$d = \frac{c\tau}{2} = \frac{c\Delta\phi}{4\pi\nu_{\text{ELO}}} \quad (3.8)$$

This is the standard operating mode of the PMD camera, where it utilizes this measured phase difference to calculate a distance for each pixel in an array, for a 3D image.

The idea behind temporal heterodyne holography is that instead of using a spatial frequency to get away from noise at DC, a temporal frequency shift is used. This means that there is no spatial angle between the LO and the scattered light impinging on the PMD. Using an optical beat note at 20 MHz will get the information away from DC in the time domain, and the PMD camera will only image light at that modulation frequency, thus eliminating the spatial DC components. When the hologram is appropriately quadratically focused and Fourier transformed, an image of the object is formed. The advantage here is that the whole resolution of the camera can be utilized instead of about a quarter as with spatial heterodyne holography.

Using a PMD for Temporal Heterodyne

The experimental setup required the removal of the LEDs that are used for illumination with a PMD. These were replaced with a laser source. The reason for this was because a temporally and spatially coherent light source is necessary to get interference.

Figure 3.8 illustrates the temporal heterodyne setup that was used in this experiment. The laser beam is split using a beam splitting cube. The LO is frequency shifted by 200MHz using an AOM, and goes through a pinhole to clean it spatially, and is sent onto the sensor via another beam splitter. The transmit beam goes to an AOM that frequency shifts the laser by 200MHz plus the frequency of the ELO, which is gathered by the PMD, resulting in a shift of 220 MHz. When the LO and scattered light combine on the sensor, a beat note is generated at the difference frequency 20 MHz, which can be seen by the PMD because it was the source of the ELO modulation signal to the AOM modulating the transmit beam, keeping everything properly phased.

This resulting hologram is the speckle pattern in both amplitude and phase, see Figures 3.10 and 3.9. This technique is different because it is measuring the phase by IQ demodulation of the amplitude modulated signal, which is the phase of the light returning from the object. This measured phase and intensity can then be processed to create an image of the object. This setup sacrifices the 3D ability of the camera, but utilizes the phase measuring capabilities for coherent imaging. This system subtracts background light automatically. As only light amplitude modulated at 20 MHz is sensed. [9]

By looking at the two optical fields interfering on the sensor, which are then mixed with the ELO, one can see how the PMD selects for only the light that was scattered off of the object. Here $I(x, y, t)$ is the optical field on the sensor of the PMD

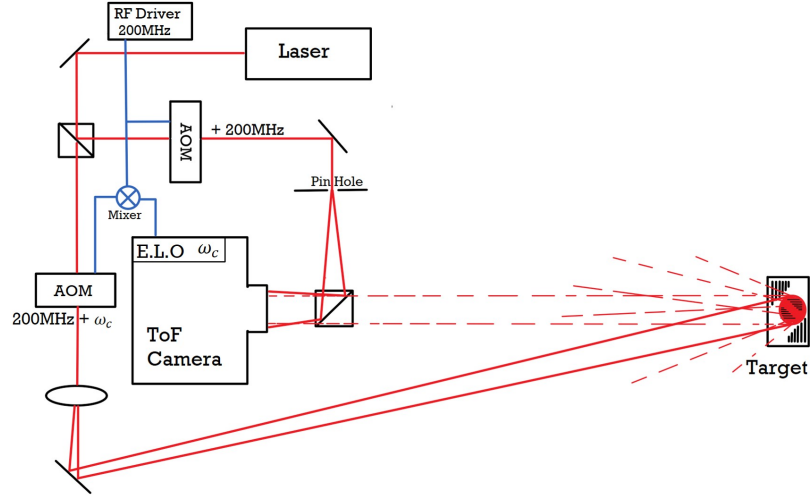


Figure 3.7: This schematic demonstrates temporal heterodyne holography. Two AOMs are used to frequency shift the beams. One beam is shifted by the AOM frequency, and the other by the AOM frequency plus the frequency of the ELO as dictated by a signal output from the PMD camera. One is used to illuminate the target, and the other is mixed with the scattered light on the PMD sensor creating a beat note at the difference frequency, 20 MHz. The phase of the beat note is the phase of the light that scattered off of the target.

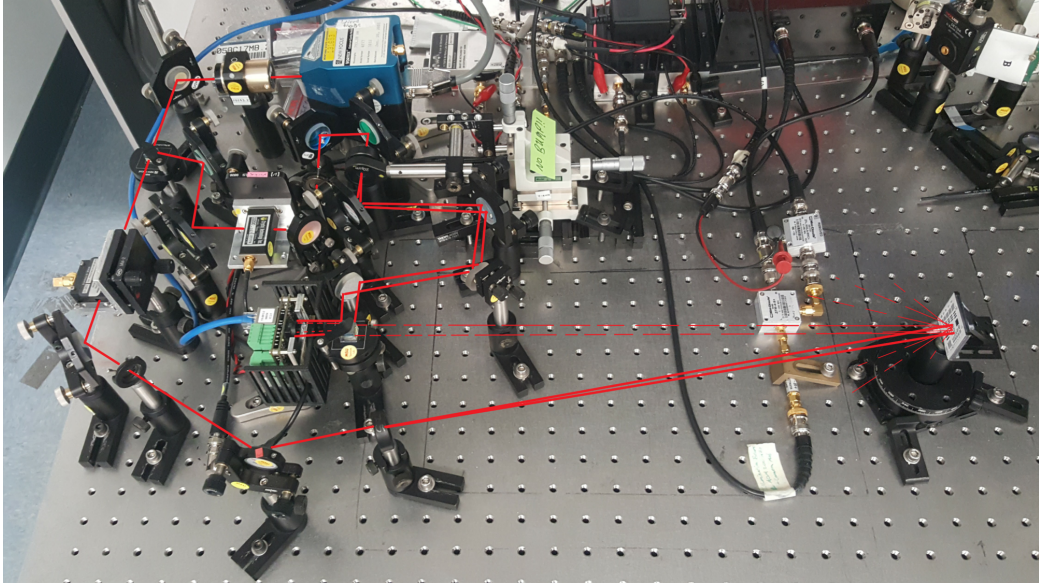


Figure 3.8: This is the bench top setup of temporal heterodyne holography using a TOF camera. One can see the traced out beam path, noting the use of the two AOMs, where one of the AOMs is driven with a signal that is the ELO frequency mixed with the AOM driving frequency. The beam splitter in front of the camera is used to make sure that both the scattered light and the LO are collinear.

which is the combination of the complex electric field of the optical local oscillator E_{LO} , and the complex electric field of the scattered light from the target E_{tar} , where ω_ℓ is the laser optical carrier frequency, ω_{aom} is the frequency contributed by the AOMs, and ω_{elo} the frequency of the ELO in the camera.

$$E_{\text{LO}}(x, y, t) = E_{\text{LO}}(x, y) e^{i(\omega_\ell + \omega_{\text{aom}})t}, \text{ and } E_{\text{tar}}(x, y, t) = E_{\text{tar}}(x, y) e^{i(\omega_\ell + \omega_{\text{aom}} + \omega_{\text{elo}})t}. \quad (3.9)$$

If one looks at the intensity of the combination of these two beams,

$$I(x, y, t) = |E_{\text{LO}} + E_{\text{tar}}|^2 = |E_{\text{LO}}|^2 + |E_{\text{tar}}|^2 + E_{\text{LO}} E_{\text{tar}}^* e^{-i\omega_{\text{elo}}t} + E_{\text{LO}}^* E_{\text{tar}} e^{i\omega_{\text{elo}}t} \quad (3.10)$$

One notes that the frequency contribution from both the AOM and the optical carrier (assuming no laser noise) drop out leaving only the modulation of the camera's ELO.

The following relationships will be used to simplify the equation after it has been separated into its sine and cosine components using Eulers formula.

$$\text{Re}\{E_{\text{LO}} E_{\text{tar}}^*\} = \text{Re}\{E_{\text{LO}}^* E_{\text{tar}}\}, \quad \text{Im}\{E_{\text{LO}} E_{\text{tar}}^*\} = -\text{Im}\{E_{\text{LO}}^* E_{\text{tar}}\} \quad (3.11)$$

$$I(x, y, t) = |E_{\text{LO}}|^2 + |E_{\text{tar}}|^2 + 2\text{Re}[E_{\text{LO}}^* E_{\text{tar}}] \cos(\omega_{\text{elo}}t) + 2\text{Im}[E_{\text{LO}}^* E_{\text{tar}}] \sin(\omega_{\text{elo}}t) \quad (3.12)$$

To calculate the phase of the return amplitude modulated signal, each pixel on the PMD will integrate this intensity against the ELO with four different phases. These correspond to the angles $\phi = 0$, $\phi = \frac{\pi}{2}$, $\phi = -\frac{\pi}{2}$ and, $\phi = -\pi$. Which is integrating against cosine, -cosine, sine, and -sine respectively. Figure 3.6 is a graphical way to look at how the phase of the light modulation is computed on the

PMD. By mixing the ELO square wave with the sinusoidal beat note. The interference leads to an amplitude modulation at the frequency that the PMD was expecting to see, and so integrating against a cosine over many periods of the same frequency will be non zero. In fact, it will measure the phase of the incoming signal. Integrating the signal against the ELO for 0 phase using cosine gives

$$\begin{aligned}
\text{Phase } 0^\circ &= \int_{t-\tau}^{t+\tau} I(x, y, t) \cos(\omega_{\text{elo}} t) dt = \\
&= \int_{t-\tau}^{t+\tau} \left[\cancel{|E_{\text{LO}}|^2}^0 + \cancel{|E_{\text{tar}}|^2}^0 + 2\text{Re}(E_{\text{LO}}^* E_{\text{tar}}) \cos(\omega_{\text{elo}} t) \right. \\
&\quad \left. + 2\text{Im}(E_{\text{LO}}^* E_{\text{tar}}) \sin(\omega_{\text{elo}} t) \right] \cos(\omega_{\text{elo}} t) dt = \text{Re}[E_{\text{LO}}^* E_{\text{tar}}], \quad (3.13)
\end{aligned}$$

where the integration time of the camera is defined as τ , which is much longer than the period of the amplitude modulation of the ELO. By integrating this intensity against a cosine over many periods, the DC terms as well as the sine terms all drop out leaving only the real part of the product of electric field amplitudes. Repeating this process for the other three phases, one gets

$$\text{Phase } 90^\circ = \int_{t-\tau}^{t+\tau} I(x, y, t) \sin(\omega_{\text{elo}} t) dt = \text{Im}[E_{\text{LO}}^* E_{\text{tar}}] \quad (3.14)$$

$$\text{Phase } 180^\circ = \int_{t-\tau}^{t+\tau} I(x, y, t) (-\cos(\omega_{\text{elo}} t)) dt = -\text{Re}[E_{\text{LO}}^* E_{\text{tar}}] \quad (3.15)$$

$$\text{Phase } 270^\circ = \int_{t-\tau}^{t+\tau} I(x, y, t) (-\sin(\omega_{\text{elo}} t)) dt = -\text{Im}[E_{\text{LO}}^* E_{\text{tar}}] \quad (3.16)$$

and the results of all four integrations are added in phase and in quadrature,

$$\text{Signal} = (\text{Phase } 0^\circ - \text{Phase } 180^\circ) + i(\text{Phase } 90^\circ - \text{Phase } 270^\circ) \quad (3.17)$$

$$\text{Signal} = 2E_{\text{LO}}^* E_{\text{tar}}. \quad (3.18)$$

The amplitude and phase can be retrieved from the signal by looking at the length and angle of the signal in the complex plane.

$$\text{Amplitude} = \sqrt{\text{Re}[E_{\text{LO}}^* E_{\text{tar}}]^2 + \text{Im}[E_{\text{LO}}^* E_{\text{tar}}]^2}, \quad \phi = \tan^{-1} \left(\frac{\text{Im}[E_{\text{LO}}^* E_{\text{tar}}]}{\text{Re}[E_{\text{LO}}^* E_{\text{tar}}]} \right) \quad (3.19)$$

When each pixel on a PMD computes this complex signal, the result is a digital hologram containing both the phase and the amplitude of the speckle field, assuming that the LO is a plane wave. This is the key difference between standard spatial heterodyne holography and temporal heterodyne holography. This method allows direct measurement of the phase and amplitude of the light coming from a target, without spatial heterodyne, demonstrating a new method of coherent imaging with PMD technology.

Hologram Processing

We now understand how the scattered light from a target is converted into a complex signal that a PMD can measure, and the next step is to extract an image from this information. It is very similar to the processing done for spatial heterodyne holography, with a couple minor differences.

The image processing used for spatial heterodyne holography involved taking an Fourier transform, and cropping down to one of two conjugate images. This is analogous to temporal heterodyne, where the conjugate images would have shown up in the temporal frequency domain at ± 20 MHz, but because IQ phase measurement methods were used, the two conjugate images have been combined to get the final signal, which is twice the amplitude one would expect.

To obtain an image from a temporal heterodyne digital hologram, Figures 3.9 and 3.10, one must first multiply it by a quadratic phase screen, seen in Figure 3.5. This compensates for any spherical wavefront curvature in the LO, as well as

the quadratic phase component in a Fresnel propagation. [2, 8] This product is then Fourier transformed to give an image of the object. Because the LO and scattered light are collinear as they reach the sensor, there is no spatial frequency offset, which puts the image in the center of the spatial frequency domain, which is okay because the background subtraction that is inherent to a PMD removed the DC component, and the IQ nature of the sensor combined the twin images. This is better than the spatial heterodyne case, because there is no need to crop down to one quadrant, and spatial resolution is not lost.

The final image shown in Figure 3.11 is a speckle averaged image. Here the object was rotated by a small angle and a hologram taken for each angle change. The rotation essentially shifts the speckle pattern in the detector plane, which was a way to sample more of the speckle field generated by the object. After each image was processed independently, the intensity images were averaged together to create a higher quality image. Speckle averaging is a type of incoherent averaging, because the

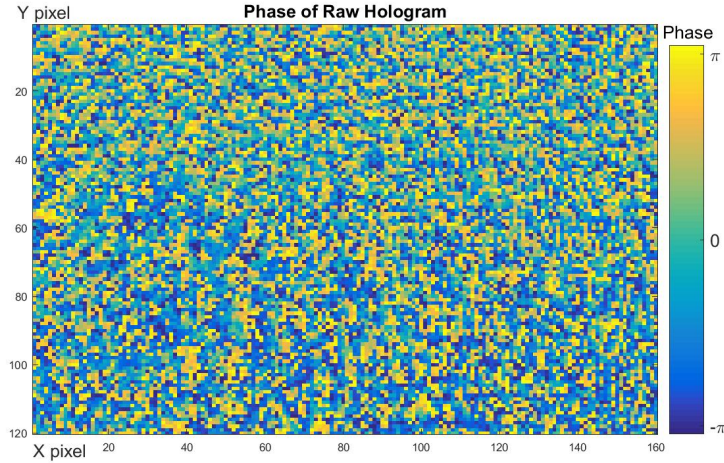


Figure 3.9: This is the phase measurement made by the PMD camera without any editing, and before multiplication with the quadratic phase correction. Here each pixel is displaying the phase as found by mixing the ELO with the combination of the LO and the scattered light, as seen in Equation 3.19.

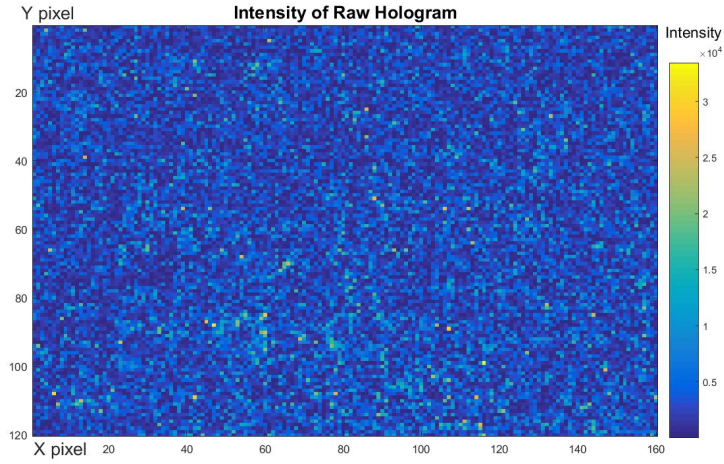


Figure 3.10: This is the intensity distribution on the PMD sensor. It is the speckle pattern from the object.

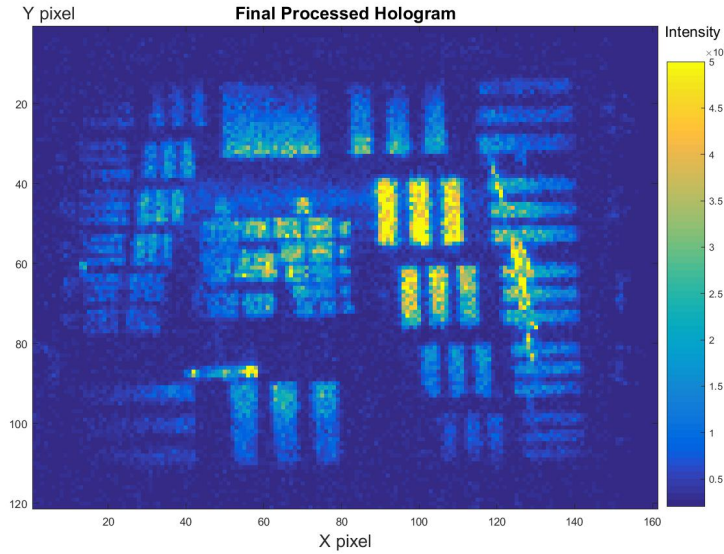


Figure 3.11: This is the resultant image if the complex valued hologram is multiplied by the quadratic phase screen, and then Fourier transformed. This image is speckle averaged, where the speckle from the object was sampled from multiple rotations, images were created, registered, then stacked to form this image.

image is formed, then added incoherently. A coherent averaging process is similar to synthetic aperture, where it is the effective size of your aperture that is changing. [10]

Temporal heterodyne digital holography is a novel holographic imaging technique with a few key differences from spatial heterodyne. The ability to use the ELO mixing process to measure light away from the DC temporal frequencies while also removing any background light that is not oscillating at the frequency of the ELO is a powerful new tool. Also, the fact that there is no spatial frequency offset, allows for placing the image in the center of the spatial frequency domain. This helps because it is no longer necessary to crop the image down to one quadrant like in spatial heterodyne, which effectively causes the loss of a quarter of the resolution. One drawback that should be fixed with time is since PMDs are still a relatively new technology, the number of pixels on a sensor is still low compared to current CCDs. [9]

Future work with this new method could include using multiple optical frequencies with multiple matching ELO frequencies in an attempt to regain some coarse range measurements, where each frequency would relate to a distance in range. [11] One other key enhancement that this new system has over the original intended uses for PMDs is the opportunity to extend the range of the 3D measurements because a laser is used to illuminate the scene instead of LEDs, which can stay collimated longer to illuminate targets at longer distances. This could become particularly useful for many active illumination scenarios.

LINEAR FREQUENCY MODULATED LIDAR

Linear Frequency Modulated Lasers

Linear frequency modulation, referred to colloquially as a chirp, is a signal that changes its frequency in a linear manner as a function of time. For all of the experiments presented in this thesis, these frequency shifts are from low to high frequency¹, as illustrated in Figure 4.1. This signal is then repeated to form a sawtooth waveform. Chirp lasers are often employed in a frequency modulated continuous wave (FMCW) laser ranging configuration as an alternative to pulsed laser systems. Pulsed lidar systems require very high bandwidth detectors, to correctly time the return of a pulse, whereas chirped laser ranging requires much lower bandwidth detector. The particular laser that was used in this experiment was a Bridger Photonics SLM LIM series laser, which had a center wavelength of 1550 nm, a bandwidth of about 50 GHz, and chirp repetition rate of 20 Hz. [12]

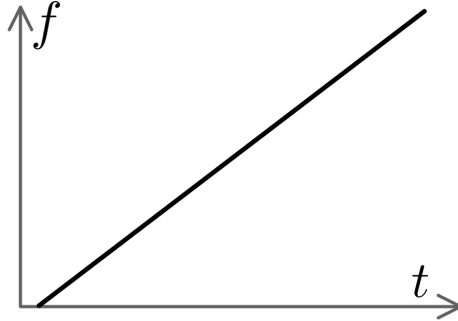


Figure 4.1: The optical frequency as a function of time for a Linear Frequency Modulated laser. This is a generic representation of what is commonly referred to as a chirp signal. The bandwidth and duration of the chirp are often modified to suit the needs of the lidar system.

¹Chirp lasers are often illustrated as being from low to high frequency, in reality it is often not truly known, but in terms of typical experimental usage, the results are the same.

LFM Lidar with a Flat or Point Target

This section will build up some of the mathematical tools that will be crucial for understanding the interference of two chirp signals, and how that can be used to get range information from a scene. Let $p(t)$ be the chirped laser signal that will illuminate a target,

$$p(t) = e^{i\pi\gamma t^2} e^{i2\pi f_\ell t} \quad (4.1)$$

where γ is the chirp rate of the frequency modulation in units of frequency per second, and f_ℓ is the optical frequency. The simplest lidar system determines the range of a flat plate or point target that only has one range delay. Essentially this will be taking a chirp signal and mixing it with a delayed copy of itself. This simulates having a local oscillator being mixed with a chirp that was used to illuminate a flat target at some range proportional to the time delay τ . The resultant electric field is,

$$E(t) = e^{i\pi\gamma t^2} e^{i2\pi f_\ell t} + e^{i\pi\gamma(t+\tau)^2} e^{i2\pi f_\ell(t+\tau)} \quad (4.2)$$

Since all optical detectors measure intensity, take the absolute square of the electric field,

$$I(t) = \left| e^{i\pi\gamma t^2} e^{i2\pi f_\ell t} + e^{i\pi\gamma(t+\tau)^2} e^{i2\pi f_\ell(t+\tau)} \right|^2 \quad (4.3)$$

$$= \cancel{e^{i(\pi\gamma t^2 - \pi\gamma t^2 + 2\pi f_\ell t - 2\pi f_\ell t)}} + e^{-i\pi\gamma(2t\tau + \tau^2) - i2\pi f_\ell \tau} + e^{i\pi\gamma(2t\tau + \tau^2) + i2\pi f_\ell \tau} + \cancel{e^{i(\pi\gamma(t+\tau)^2 - \pi\gamma(t+\tau)^2 + 2\pi f_\ell(t+\tau) - 2\pi f_\ell(t+\tau))}} \quad (4.4)$$

$$= 2 \cos (\pi \gamma (2t\tau + \tau^2) + 2\pi f_\ell \tau) + 2 \quad (4.5)$$

$$= 2 \cos (\omega t + \phi) + 2, \quad \omega \equiv 2\pi \gamma \tau, \quad \phi \equiv \pi \gamma \tau^2 + 2\pi f_\ell \tau. \quad (4.6)$$

So the intensity beat note of the light is oscillating at an angular frequency ω which is proportional to the time delay between the two chirps τ . The next step is to take the Fourier transform of the intensity in the time domain. This produces a delta peak at that frequency. The next step is to relate the frequency to a distance from the transmitter to the target and back to the receiver. This is the round trip distance which adds a multiple of two into the distance. Hence,

$$\tau = \frac{2R}{c}, \quad (4.7)$$

which is the expression relating time delay to distance. Using this to solve for the range in terms of the chirp rate γ and frequency ω gives,

$$R = \frac{\omega c}{4\pi \gamma} \quad (4.8)$$

Figure 4.2 shows how this system works schematically if one starts with two chirps offset by a time τ that interfere with each other to create a beat note at an angular frequency ω . Fourier transforming this signal one can precisely locate that frequency, and back out a range measurement using Equation 4.8. Which is assuming that the chirp rate γ is known. [12]

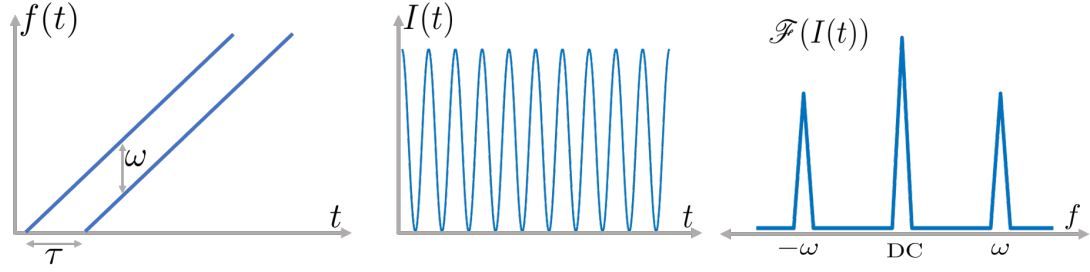


Figure 4.2: The leftmost plot shows the frequency modulation curves of the two signals, one being a delayed copy of the other. The middle plot shows the resulting intensity beat note due to the interference between the two signals. The right plot shows the frequency domain of the intensity pattern, which has a DC component due to the fact that intensity is a positive quantity, and a peak at plus and minus ω which is the signal that one can recover a range measurement from.

Range and Cross Range Imaging with an LFM Lidar

The previous section explored how to extract range information from the interference between a chirp and a delayed copy of itself. This section will generalize this to a target that has more range depth than just a point target. I will also generalize to an array of detectors, instead of a single detector as in the previous section. The switch to a target with more depth will lead to a range profile in the Fourier domain instead of just a single peak at the frequency of the beat note. The switch to a detector array, will allow for gathering information in the spatial frequency domain of the target. By measuring both a range profile for each pixel in the array, as well as the relative intensities between pixels, one can reconstruct an image that is very similar to what one would see when doing synthetic aperture radar or lidar. Namely it is an image that is range in one dimension and cross range, sometimes called azimuth, in the other. [10]

Figure 4.3 shows how to extract a range profile from the intensity measurement

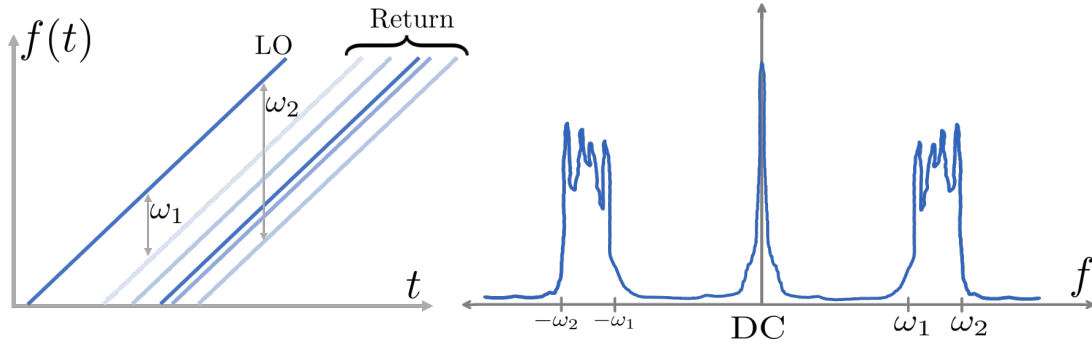


Figure 4.3: Here is an intuitive way to think about a target with depth to it. The local oscillator is the “clean” chirp on the left of the first figure, and it is mixed with a chirp that has interacted with the target. The multiple lines on the return indicate reflections from different range location in the target. In this example, ω_1 is the lower frequency beat note, and shows up in the Fourier domain closer to DC. Similarly for ω_2 which is a higher frequency, corresponds to a larger distance.

of a single pixel. In this thought experiment, a chirped LO

$$E_{\text{LO}} = e^{i2\pi f_\ell t} e^{i\pi\gamma t^2} \quad (4.9)$$

mixes with the target return,

$$E_{\text{tar}} = \sum_n r_n e^{i2\pi f_\ell (t-n\Delta t)} e^{i\pi\gamma (t-n\Delta t)^2}. \quad (4.10)$$

Where f_ℓ is the optical frequency, r_n is the discrete range profile which corresponds to the target complex reflectivity at different depths, and the $n\Delta t$ is the time delays due to the target depth. Another way to think about r_n , is that it is the amount of light reflected at $n\Delta t$ which is a sum of all the reflectivities at distances $n\Delta t c$ away from the detector. The target return (as seen by the detector array) is represented by multiple discrete time delayed chirps, hence the sum over n , of varying intensities

corresponding to different range returns in the target. These two fields are mixed,

$$I = |E_{\text{LO}} + E_{\text{tar}}|^2 \quad (4.11)$$

$$= \left| e^{i2\pi f_\ell t + i\pi\gamma t^2} + \sum_n r_n e^{i2\pi f_\ell(t-n\Delta t) + i\pi\gamma(t-n\Delta t)^2} \right|^2 \quad (4.12)$$

Each range return chirp has a unique beat note when it interferes with the LO, which can be seen in the Fourier domain as many spikes on top of each other. Expanding the intensity equation and canceling terms,

$$I_n(t) = \sum_n r_n \sum_n r_n^* + \sum_n r_n \underbrace{e^{-i2\pi f_\ell n\Delta t} e^{i\pi\gamma(n\Delta t)^2}}_{\text{ignore these phases}} \underbrace{e^{-i2\pi\gamma t n\Delta t}}_{\text{Fourier Kernel}} + \text{c.c.} + 1 \quad (4.13)$$

where the “c.c.” is the complex conjugate of the second term. The ignored phases are constants, because they do not depend on t , and are commonly ignored in chirped ranging, and the third phase term in the mixing term looks like the discrete Fourier transform kernel. With this in mind one can think of the intensity equation in the following terms, where the subscript denotes time steps,

$$I_n(t) = r_n \star r_n + \mathcal{F}\{r_n\} + \mathcal{F}^{-1}\{r_n\} + 1. \quad (4.14)$$

The first term is the autocorrelation of the range profile, the mixing terms are the temporal Fourier transforms of the range profiles, and the last term is the autocorrelation of the LO. If the intensity is fluctuating in time according to the Fourier transform of the range profile, one only has to isolate a mixing term and take a Fourier transform to retrieve the range profile of the target.

$$\text{Range Profile} = \mathcal{F}\{\mathcal{F}^{-1}\{r_n\}\} \quad (4.15)$$

Similar to the flat target, I am directly measuring the range spatial frequency, and the Fourier transform of this gives me a range profile for the target as measured by one pixel on the detector array. Typically the way one would isolate one of the mixing terms is via a time delay on the local oscillator path. This will offset the mixing terms in the temporal frequency domain, which allows one to distinguish them from each other and the DC terms. That is how the range is measured, and that corresponds to one dimension of the image. [10]

By combining the pixels in a 1D horizontal array, information can be obtained about the azimuth dimension². If this measurement is taken without any imaging optics, the intensity distribution on the pixels represents the spatial frequency content of the target due to free space diffraction. To synthesize an image from this, one has to Fourier transform along the azimuth (pixel) dimension. Combined with the range processing there will be an image of the target where range is on the vertical axis, and azimuth on the horizontal axis. If imaging a flat target tilted away from the detector plane, then it will look like a “normal” image of the target, because the elevation and range are proportional. This is the basis for the first experiment using LFM lidar described in the next section.

Experimental Demonstration of Imaging with an LFM Lidar

This section will explain the setup for range and azimuth imaging with an LFM lidar system.

To start, I will go over the constraints of the system as dictated by things like wavelength, detector bandwidth in both space and time, target size, LFM chirp rate, and so forth. All of these variables are dependent upon one another, and it is a little

²Synthetic Aperture Radar literature commonly refers to this dimension as “cross range”, but I believe it is more accurate in my case to refer to it as azimuth.

tricky making sure they all stay within reasonable experimental constraints. [10]

The spatial sampling requirements are fairly straightforward. Namely, one needs to sample the light scattered from the target at twice the spatial frequency of the speckle that one would expect to see. This is a Nyquist sampling constraint, which is basically saying that if the signal is sampled at twice the maximum frequency, one can recover the signal without aliasing. [13] The speckle pattern may seem arbitrary, but it is a mixture of all of the spatial frequencies of the target, and as I chose the target, I can set its size and distance from the detector array to match the sampling bandwidth of the detector. Using an analogy to Young's double slit experiment, one can see that the interference between two points on the target separated by a distance $d = 13.5$ cm, and $z = 3.5$ m from the detector array, at a wavelength of $\lambda = 1550$ nm, will give me a minimum spatial period l of

$$l = \frac{\lambda z}{d} = 40.2 \mu\text{m} \quad (4.16)$$

at the detector plane. This needs to be at least twice $20 \mu\text{m}$, the pixel size on the detector, to fully capture all of the spatial frequency content of the speckle from the target. In order to calculate the azimuth resolution of the target, one can think about the detector array in terms of a synthetic aperture. In this experiment the array is 80×16 pixels, and because no lens is used, the resolution for each spatial dimension depend on the size of the aperture in that dimension. Where the square pixels on the sensor are $L_{\text{pix}} = 20 \mu\text{m}$. [1, 10]

$$X_{\text{res}} = \frac{\lambda z}{80(L_{\text{pix}})} = 3.39 \text{ mm}, \quad Y_{\text{res}} = \frac{\lambda z}{16(L_{\text{pix}})} = 17.0 \text{ mm} \quad (4.17)$$

The better resolution is in the dimension with more samples (azimuth) but the elevation dimension also contains some information, just with worse resolution. For

now I will ignore the fact that there is any information in the elevation dimension, and treat the camera as a line camera. The elevation information will be revisited during the processing steps. The constraints here are necessary to fully capture object information in the spatial (azimuth) domain, as well as in the range domain.

The sampling requirements in the time domain are contingent upon the relationship between the target depth, the chirp rate of the laser, and the temporal bandwidth of the detector array. The high speed detector array used in this experiment has a maximum frame rate (bandwidth) that depends on the integration time, and the windowing of the array. For this first experiment, an integration time of $20\ \mu\text{s}$ was used, with a window of 80×16 pixels. The chirp rate of the laser was set to $\gamma = 1.179\ \text{THz/s}$. The camera will sample as many times as it can throughout the duration of a single chirp. The stable chirp length in time is $t_{\text{chirp}} = 43\ \text{ms}$, and the maximum frame rate of the camera is $f_{\text{cam}} = 34\ \text{kHz}$ so the number of samples N taken during a single chirp is

$$N = f_{\text{cam}} t_{\text{chirp}} = 1462\ \text{samples}. \quad (4.18)$$

The bandwidth of the chirp, how much frequency it sweeps over, is $B_{\text{chirp}} = 50.7\ \text{GHz}$, which in conjunction with the sampling rate, allows one to define a maximum depth, and minimum feature size (in range) of the target. The minimum feature size is equal to the range resolution

$$Z_{\text{res}} = \frac{c}{2B_{\text{chirp}}} = 2.96\ \text{mm}. \quad (4.19)$$

The maximum range depth is equal to the range ambiguity, which is a way to think about how deep the target can be before it starts to wrap back around in range depth. It depends on the optical frequency change between samples in time $\Delta\nu \equiv \frac{B_{\text{chirp}}}{N}$, and

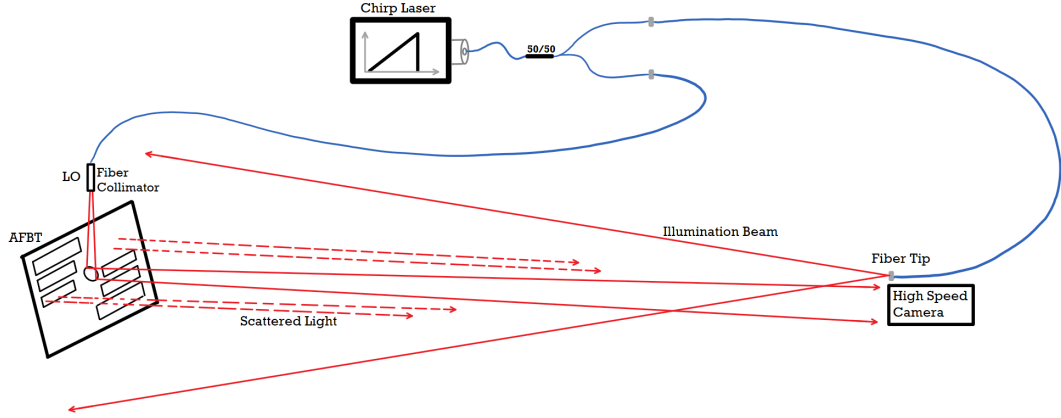


Figure 4.4: This is the schematic for the LFM lidar experiment outlined in this chapter. A fiber coupled chirped laser, is split into two paths. The LO path is reflected from a mirror in the scene directly onto the camera, and the transmit path is mounted on top of the camera to illuminate the scene, and scatter back to the detector array. The optical path length difference between the two paths was chosen so that the interference frequency would be within the bandwidth of the camera.

is defined to be

$$Z_{\text{amb}} = \frac{c}{2\Delta\nu} = 4.3 \text{ m} = N Z_{\text{res}}. \quad (4.20)$$

Now some constraints on what the target size can be, based off of things like the chirp rate of the laser, the bandwidth of the camera, and the pixel size on the array have been determined. The next step is to choose a fiber delay length for the LO so that the interference between the LO and the scattered light from the target is within the bandwidth of the detector, but away from the DC components. Figure 4.4 illustrates the overall setup. The relationship between the chirp rate, time delay, path length difference, and resulting beat note, will be used to choose the correct fiber delay between the two paths.

$$\gamma = \frac{f}{\tau} = \frac{fc}{n_{\text{f}}d} \quad (4.21)$$

If one thinks back to Figure 4.2, it becomes fairly clear that this is how one calculates a frequency f that corresponds to the beat note between two chirped signals. Since the detector has a sample rate of 34 kHz, then the highest frequency beat note that can be detected would be at the Nyquist frequency, half of the bandwidth, at 17 kHz. [3] Because the target has some depth to it, one needs to choose a fiber delay length d such that the highest frequency beat note due to the largest path length difference is less than 17 kHz. As it turns out, a path length difference of about one meter of fiber worked well. Figure 4.5 shows the Fourier transform of the recorded intensity

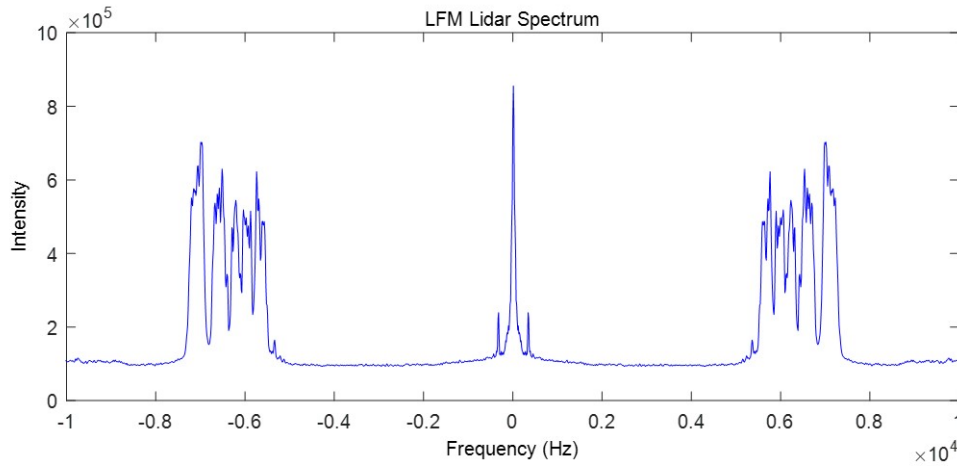


Figure 4.5: This is the Fourier transform of the intensity recieved by one pixel on the high speed camera as shown in figure 4.4. One can see that there are multiple peaks in the plot corresponding to different range returns from the target.

variations due to the interference between the LO and the scattered light from the scene. It is evident that there are many peaks as was expected.

The next step is to crop this “spectrum” down to just the region around the signal. This is actually the range profile as measured by a single pixel on the array. This is done for each pixel in order to isolate the range dimension of the output image. The next step is to Fourier transform along the spatial dimension. This is necessary because lensless imaging is measuring spatial frequencies of the azimuth

information. This needs to be Fourier transformed in order to get that azimuth spatial information back. This process can be seen in more detail in the LFM lidar Matlab script in Appendix A.

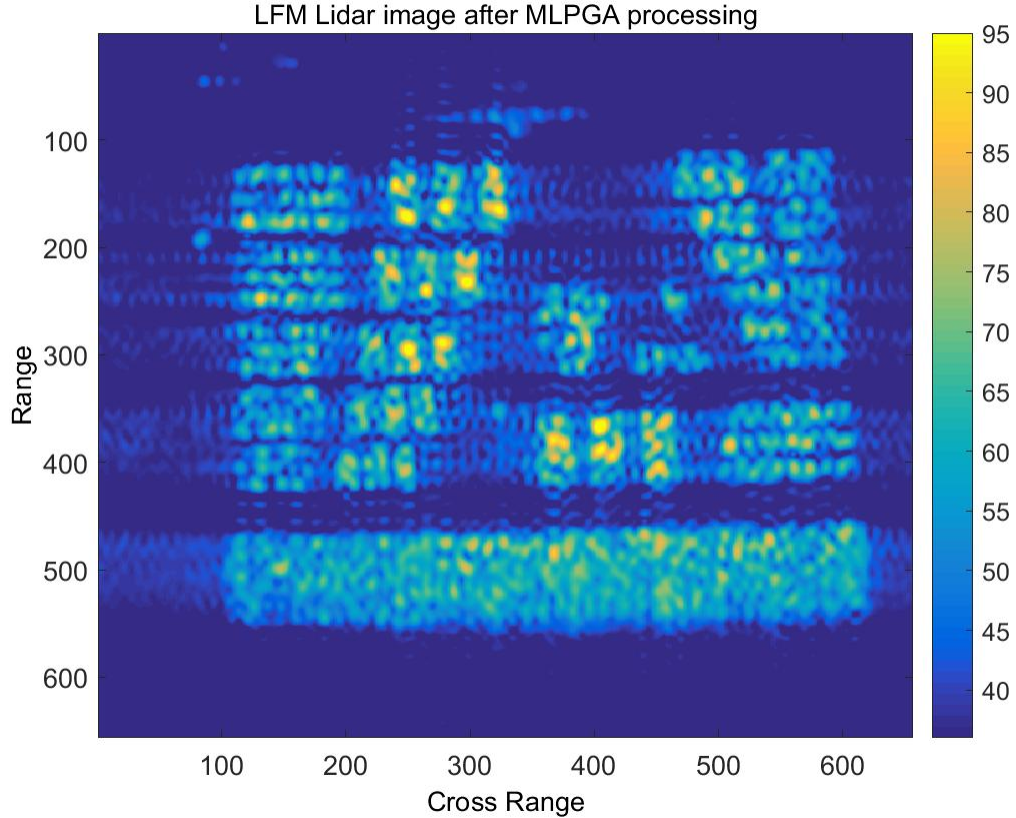


Figure 4.6: Here is the output image from a LFM lidar imaging system. It is similar to synthetic aperture imaging techniques because the final image is in the range and cross range (azimuth) dimensions. This image was corrected using an MLPGA algorithm. [14]

This type of ranging has been done before, [10,14] but usually not with an array or with this fast of a detector array. This is very similar to a synthetic aperture lidar, but instead of moving a single detector along a path to synthesize an aperture, I am just taking the values of the whole array and “digitally synthesizing” the aperture. [10,14,15]

SHEARED IMAGING THEORY

Sheared Imaging was first proposed in 1983 by Richard Hutchin. [4] It was to be an alternative to adaptive optical techniques used in astronomical telescopes. Both adaptive optics and Sheared Coherent Interferometric Photography (SCIP) seek to eliminate distortion due to atmospheric turbulence. SCIP is an active imaging system that is lensless, while adaptive optics is passive and employs deformable mirrors to correct atmospheric aberrations. SCIP is a coherent imaging technique that seeks to obtain the amplitude and phase of an optical field.

The optical field that it aims to measure is the “speckle” from the target of interest, which is coherent light that has scattered off of the target, and undergone free space diffraction on the way back to the detector. Fourier optics describes free space propagation as essentially the same thing as performing a Fourier transform of the target’s spatial information. [1] The speckle is identical to the spatial frequency content of the target. If one can obtain the amplitude and phase of the speckle, an image can be reconstructed with a simple spatial Fourier transform. [4]

Sheared Coherent Interferometric Photography

When light propagates through weak to moderate atmospheric turbulence, it scrambles the phase information of the speckle field, but leaves the intensity mostly unaltered. What SCIP does is encode the phase information of the speckle field in such a way that it is immune to the atmospheric turbulence. The way this was first proposed was to illuminate a target with multiple spatially separated coherent sources. The speckle behaves in a manner similar to light reflecting off of a mirror, where if the object is illuminated from one location, a speckle pattern will appear in the far field, but if illuminated from a slightly different location, the same speckle pattern will be

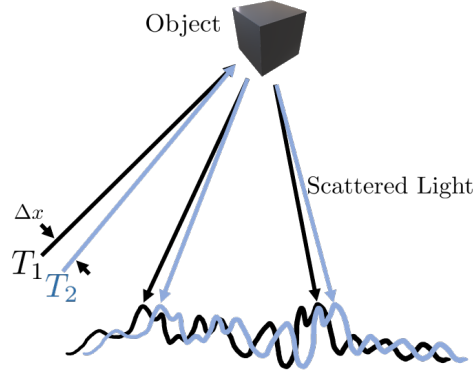


Figure 5.1: Illustrating how an object illuminated from two spatially shifted transmitters T_1 and T_2 lead to translated copies of the speckle in the far field. The spatial shift is defined as Δx . Notice that in the object plane, the spatial shift has changed into a phase difference thanks to the Fourier Shift Theorem.

translated slightly in the far field. The goal behind this type of illumination pattern is that the two beams will experience nearly the same atmospheric turbulence, affecting the phase of the beams in the same way. This is to say that the phase relationship between the two beams is largely unaltered. If an optical frequency shift is used to create a beat note between the two beams, the phase of the intensity modulations is exactly the phase difference between the two speckle patterns. This measurement of the phase difference is not enough information by itself to reconstruct an image of the target, but it can be used with other measurements of phase differences to reconstruct the phase of the speckle of the target, and thus an image of the target can be reconstructed.

SCIP was first demonstrated using three frequency shifted laser beams where one was spatially shifted slightly in the x direction and the other shifted slightly in the y direction relative to the first beam. These two “sheared” beams would generate two different measurements of phase differences on the detector plane. The first would represent the phase differences of the speckle pattern in the x direction,

and the second would represent the phase differences in the y direction. These are sometimes referred to as “gradients” because if the spatial shifts are small enough, it would look like a phase derivative of the speckle in that direction. This is a nice way to think about what is happening later in the processing where the different shear measurements are “integrated” to get an estimate of the phase of the speckle of the target. [4]

The way Hutchin approached the problem of reconstructing the phase of the target’s speckle was essentially an iterative two dimensional integration using the shears in x and y as the inputs. This method integrated multiple different paths in the grid to attempt to estimate the phase of the speckle. While this method does work, it will not give the best phase estimate, as all possible paths are not integrated over.

Another more recent sheared imaging technique invented by Trex Enterprises Corp, uses spatially sheared illumination beams to reconstruct a three dimensional output image. [16] This invention is somewhat related to the experiment I will demonstrate in chapter 7. All of these sheared imaging techniques have one common processing step, which is finding a way to reconstruct the phase and amplitude of the speckle pattern. There are many ways to do this, and in the next section, I will explore a different way one can “integrate” these sheared measurements to find a more robust estimation of the phase of the speckle.

A Better Way to Integrate Shears

During my summer at AFRL, Dave Rabb and Jason Stafford introduced me to a better way to integrate multiple shear measurements simultaneously using a

“covariance”¹ matrix method. I used this method for all of my simulations and experimental data processing without truly understanding why it worked. Zeb Barber and I found a more intuitive way to explain this method, which we base loosely off of ideas found in Jakowatz’s paper, [17] and lay this argument out in the rest of this chapter.

Charles Jakowatz developed an eigenvector method to find the maximum-likelihood estimation of phase errors in synthetic aperture radar imagery. [17] I believe that similar ideas can apply to the phase reconstruction problem outlined in the previous section. Jakowatz used this method to solve for phase aberrations in his data that were to be removed, but I want to use this method to find the phase in the first place. To introduce this problem I will define a vector $\vec{S}$ which has N complex components,

$$S_n = a_n e^{i\phi_n} \quad (5.1)$$

where n denotes the n th element in the vector, a_n is the amplitude, and ϕ_n is the phase. This vector can represent the speckle field of the object that I want to find. If all I can measure is the interferences between $\vec{S}$ and spatially shifted copies of itself, what would this look like in matrix form? For this particular case it would look like

¹The term “covariance” is borrowed from statistics, and is usually an averaged quantity. In this case the matrix looks more like a measurement of the correlation of the speckle field, but covariance is used commonly in the literature, so I will use that nomenclature to avoid confusion.

the product $\vec{S}\vec{S}^\dagger$, where the dagger denotes a complex conjugation transpose operator.

$$\vec{S}\vec{S}^\dagger = \begin{bmatrix} a_1^2 & a_1 a_2 e^{i(\phi_1 - \phi_2)} & a_1 a_3 e^{i(\phi_1 - \phi_3)} & \dots & a_1 a_N e^{i(\phi_1 - \phi_N)} \\ a_2 a_1 e^{i(\phi_2 - \phi_1)} & a_2^2 & a_2 a_3 e^{i(\phi_2 - \phi_3)} & & a_2 a_N e^{i(\phi_2 - \phi_N)} \\ a_3 a_1 e^{i(\phi_3 - \phi_1)} & a_3 a_2 e^{i(\phi_3 - \phi_2)} & a_3^2 & \ddots & \vdots \\ \vdots & & \ddots & \ddots & a_{N-1} a_N e^{i(\phi_{N-1} - \phi_N)} \\ a_N a_1 e^{i(\phi_N - \phi_1)} & a_N a_2 e^{i(\phi_N - \phi_2)} & \dots & a_N a_{N-1} e^{i(\phi_N - \phi_{N-1})} & a_N^2 \end{bmatrix} \quad (5.2)$$

An off-diagonal of this matrix is exactly the same as a shear measurement, where one would get the product of amplitudes and phase differences between shifted copies of the speckle field. One may notice that the lower diagonals are complex conjugates of the upper diagonals.

What would happen if one multiplies this matrix by the original vector $\vec{S}$? Since $\vec{S}^\dagger \vec{S}$ is just a number, it can be moved to the front of $\vec{S}$, and replaced with a coefficient λ which just looks like a scaled version of the original vector $\vec{S}$.

$$\vec{S}\vec{S}^\dagger \vec{S} = \vec{S}(\vec{S}^\dagger \vec{S}) = \vec{S}\lambda = \lambda\vec{S} \quad (5.3)$$

This can be recognized as an eigenvector problem, where the eigenvector solution is the vector $\vec{S}$ that I want to retrieve, but the only measurements I have are elements of $\vec{S}\vec{S}^\dagger$. Rewriting Equation 5.3 on gets,

$$[\vec{S}\vec{S}^\dagger - \lambda\mathbf{I}]\vec{S} = 0. \quad (5.4)$$

It is only possible to recover the vector $\vec{S}$ in its entirety if the matrix $\vec{S}\vec{S}^\dagger$ is fully

populated. Which is to say, every possible shear is measured and put into this matrix.

To be a little bit more concrete, the right hand side of Equation 5.3 would look like,

$$\vec{S}\vec{S}^\dagger\vec{S} = \begin{bmatrix} a_1^3 e^{i(\phi_1 - \phi_1 + \phi_1)} + a_1 a_2^2 e^{i(\phi_1 - \phi_2 + \phi_2)} + a_1 a_3^2 e^{i(\phi_1 - \phi_3 + \phi_3)} + \dots + a_1 a_N^2 e^{i(\phi_1 - \phi_N + \phi_N)} \\ a_2 a_1^2 e^{i(\phi_2 - \phi_1 + \phi_1)} + a_2^3 e^{i(\phi_2 - \phi_2 + \phi_2)} + a_2 a_3^2 e^{i(\phi_2 - \phi_3 + \phi_3)} + \dots + a_2 a_N^2 e^{i(\phi_2 - \phi_N + \phi_N)} \\ \vdots \\ a_N a_1^2 e^{i(\phi_N - \phi_1 + \phi_1)} + a_N a_2^2 e^{i(\phi_N - \phi_2 + \phi_2)} + \dots + a_N a_{N-1}^2 e^{i(\phi_N - \phi_{N-1} + \phi_{N-1})} + a_N^3 e^{i(\phi_N - \phi_N + \phi_N)} \end{bmatrix} \quad (5.5)$$

Where it is clear that each element of the vector in Equation 5.5 can have the amplitude and phase of $\vec{S}$ factored out.

$$\vec{S}\vec{S}^\dagger\vec{S} = \begin{bmatrix} (a_1^2 + a_2^2 + a_3^2 + \dots + a_N^2) a_1 e^{i\phi_1} \\ (a_1^2 + a_2^2 + a_3^2 + \dots + a_N^2) a_2 e^{i\phi_2} \\ \vdots \\ (a_1^2 + a_2^2 + a_3^2 + \dots + a_N^2) a_N e^{i\phi_N} \end{bmatrix} = (a_1^2 + a_2^2 + a_3^2 + \dots + a_N^2) \vec{S} = \lambda \vec{S} \quad (5.6)$$

Which is clearly an eigenvalue problem where the eigenvalue is the sum of the squares of the amplitudes, and the eigenvector is the amplitude and phase solution of the speckle field $\vec{S}$.

In an experiment, it would be unreasonable to attempt to measure every possible shear, so the following math will be representative of a situation where only two shears are measured. I will show that the resulting eigenvector is no longer an exact solution of the amplitude and phase, but only a phase estimate of the speckle field $\vec{S}$.

Here I will define a new matrix that will contain only two shear measurements. This is a sparse matrix because most of the elements will be zeros, and only the ± 1 and ± 2 off diagonals will be filled. This will be defined as $\widetilde{\vec{S}\vec{S}^\dagger}$ which is a sparse version of $\vec{S}\vec{S}^\dagger$

$$\widetilde{\vec{S}}\vec{S}^\dagger = \begin{bmatrix} 0 & a_1a_2e^{i(\phi_1-\phi_2)} & a_1a_3e^{i(\phi_1-\phi_3)} & 0 & \dots & 0 \\ a_2a_1e^{i(\phi_2-\phi_1)} & 0 & a_2a_3e^{i(\phi_2-\phi_3)} & a_2a_4e^{i(\phi_2-\phi_4)} & \ddots & \vdots \\ a_3a_1e^{i(\phi_3-\phi_1)} & a_3a_2e^{i(\phi_3-\phi_2)} & 0 & a_3a_4e^{i(\phi_3-\phi_4)} & \ddots & 0 \\ 0 & a_4a_2e^{i(\phi_4-\phi_2)} & a_4a_3e^{i(\phi_4-\phi_3)} & 0 & \ddots & a_{N-2}a_Ne^{i(\phi_{N-2}-\phi_N)} \\ \vdots & \ddots & \ddots & \ddots & \ddots & a_{N-1}a_Ne^{i(\phi_{N-1}-\phi_N)} \\ 0 & \dots & 0 & a_Na_{N-2}e^{i(\phi_N-\phi_{N-2})} & a_Na_{N-1}e^{i(\phi_N-\phi_{N-1})} & 0 \end{bmatrix} \quad (5.7)$$

To extract an estimate of the speckle phase from the shears that generated this matrix, I will next multiply this matrix $\widetilde{\vec{S}}\vec{S}^\dagger$ by a vector $\vec{\nu}$, where the amplitudes and phases of its elements are unknown.

$$\nu_n = b_n e^{i\psi_n} \quad (5.8)$$

One can see that the result shown in Equation 5.9 is a column vector where each element is a sum of products of amplitudes with phase differences. Where since a total of four off diagonals of the matrix are populated, the first element in Equation 5.9 will have two terms, the second three, most of the rest four, then three and two again in the last two terms of the vector.

$$\widetilde{\vec{S}\vec{S}^\dagger}\vec{\nu} = \begin{bmatrix} a_1a_2b_2e^{i(\phi_1-\phi_2+\psi_2)} + a_1a_3b_3e^{i(\phi_1-\phi_3+\psi_3)} \\ a_2a_1b_1e^{i(\phi_2-\phi_1+\psi_1)} + a_2a_3b_3e^{i(\phi_2-\phi_3+\psi_3)} + a_2a_4b_4e^{i(\phi_2-\phi_4+\psi_4)} \\ a_3a_1b_1e^{i(\phi_3-\phi_1+\psi_1)} + a_3a_2b_2e^{i(\phi_3-\phi_2+\psi_2)} + a_3a_4b_4e^{i(\phi_3-\phi_4+\psi_4)} + a_3a_5b_5e^{i(\phi_3-\phi_5+\psi_5)} \\ a_4a_2b_2e^{i(\phi_4-\phi_2+\psi_2)} + a_4a_3b_3e^{i(\phi_4-\phi_3+\psi_3)} + a_4a_5b_5e^{i(\phi_4-\phi_5+\psi_5)} + a_4a_6b_6e^{i(\phi_4-\phi_6+\psi_6)} \\ \vdots \\ a_Na_{N-2}b_{N-2}e^{i(\phi_N-\phi_{N-2}+\psi_{N-2})} + a_Na_{N-1}b_{N-1}e^{i(\phi_N-\phi_{N-1}+\psi_{N-1})} \end{bmatrix} \quad (5.9)$$

Taking a closer look at Equation 5.9, if $\phi_n = \psi_n$, then the phase on each of the terms reduce to $e^{i\phi_n}$ for the n th vector element. If that is the case, that $\phi_n = \psi_n$, then that element's phase, and its related amplitude, can be factored out of all the terms component by component.

$$\widetilde{\vec{S}\vec{S}^\dagger}\vec{\nu} = \begin{bmatrix} (a_2b_2 + a_3b_3)a_1e^{i\phi_1} \\ (a_1b_1 + a_3b_3 + a_4b_4)a_2e^{i\phi_2} \\ (a_1b_1 + a_2b_2 + a_4b_4 + a_5b_5)a_3e^{i\phi_3} \\ (a_2b_2 + a_3b_3 + a_5b_5 + a_6b_6)a_4e^{i\phi_4} \\ \vdots \\ (a_{N-2}b_{N-2} + a_{N-1}b_{N-1})a_Ne^{i\phi_N} \end{bmatrix} \quad (5.10)$$

Equation 5.10, is the ideal outcome, where each term had the correct phase to cancel out the phase of the vector $\vec{\nu}$. Each element of this vector has different amplitudes, which leads to an incorrect measurement of the amplitude. This loss of amplitude is the reason why this method is used to solve for an estimate of the phase of the speckle only. Next we need to see what happens when ϕ_n is not exactly equal to ψ_n .

It is helpful to think of each term in one element of the vector on the right side of Equation 5.9 as phasors, where if they all have the same phase, they add together to form one long phasor in the direction of the phase to be recovered.

This is illustrated in Figure 5.2, where the left most image is a case in which $\phi_n \neq \psi_n$, and the length of the sum of all of the phasors is small. In the middle image, $\phi_n \approx \psi_n$, showing that the length of the phasor is longer, and somewhat in the correct direction. The best phase estimation is shown in the right image, in which $\phi_n \cong \psi_n$, where the resultant phasor is the longest, this indicates that the phase is closest to the phase we want to recover. [17]

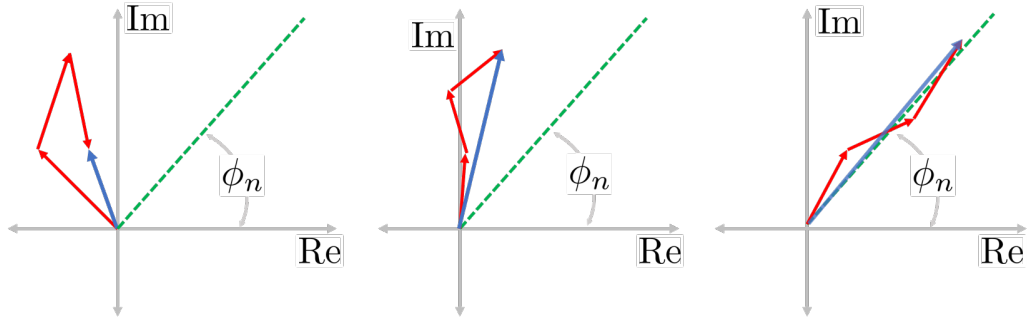


Figure 5.2: This illustration shows three cases of the sum of the terms in an element of the right hand side of Equation 5.9. The terms being added are shown in red, and the resultant phasor in blue. The green dashed line is the angle of the correct phase (that I am trying to find). In the leftmost image one can see that the phasors have incorrect angles, and add in such a way that the resultant phasor is short. The middle figure shows phasors with slightly closer matches to the correct phase, and the resultant phasor is slightly longer. The right image shows phasors with nearly correct phases, that all add up to a longer resultant phasor that is in the correct direction.

In this way, the length of the phasor is analogous to the scaling factor on the vector $\vec{\nu}$, where the larger the scaling factor the more closely the vector represents the phase of the unknown speckle field that I started with. Indeed one can recognize this whole problem as the following eigenvalue problem, where $\widetilde{\vec{S}\vec{S}^\dagger}$ is defined to be the covariance matrix $\mathbf{C}$, the vector $\vec{\nu}$ is the eigenvector, and the scaling factor, or length of the summed phasors, is the eigenvalue λ .

$$\mathbf{C}\vec{\nu} = \lambda\vec{\nu} \quad (5.11)$$

Where if this equation is solved for the eigenvector that corresponds to the largest eigenvalue, that particular eigenvector will be the most correct estimation of the phase of the original speckle field $\vec{S}$. Previously I talked about how SCIP was measuring two different shears, one in x and the other in y . These shears could populate the diagonals of the covariance matrix I presented. With only two diagonals of information, the covariance matrix is very sparse, but the method outlined above can still be used to reconstruct an estimation of the phase of the speckle of the target. This leads to the realization that the more shears one can measure, the better the phase estimate will be.

In the next chapter I will examine a simulation of SCIP, in which I will refer to it as spatial shearing lidar. It will involve more of the specifics on how to correctly implement the phase reconstruction technique that was outlined in this chapter. I still have yet to explain how to recover the amplitude information from the speckle field, as the method outlined above only provides a phase estimate, this too will be explained in the next chapter.

SPATIAL SHEARING LIDAR SIMULATION

This chapter will go through the process of building a simulation of the spatial shearing imaging technique that uses the covariance matrix approach. The purpose of this simulation is to help motivate and understand the complicated processing algorithm that this imaging technique employs. The Matlab code for this is in Appendix B, and each section of this chapter corresponds to a cell or set of cells in the code.

Creating a Synthetic Target

To start the simulation, it would make sense to create an object to image. To do this, I took a digital CAD model of an Air Force bar target, and added simulated roughness and gave the target a range profile. The roughness is used to create an object that will behave like a diffuse reflector. The range profile creates a three dimensional target for the sheared imaging. Both are accomplished by creating a meshgrid in Matlab for the azimuth and elevation (spatial) dimensions, which gives the target its size. To give the target a range profile (i.e. z -grid), I took a copy of the elevation (y -grid), made it negative, simulating a 45° backtilted planar target, and added a small random number to each point to make it rough. This is then turned into a phase by multiplying by the wavenumber and then the object by that. See Figure 6.1b. For a single wavelength the phase is equivalent to a depth, and I am using a single wavelength not a chirp for this simulation. It is also easier to work with a complex two dimensional object than a three dimensional object. [5] This operation is

$$\text{Complex Object} = \text{Real Object } e^{i2k(-\tilde{y}\tan(\theta)+\text{rand}\times 10^{-5})} = \text{Real Object } e^{i2k(-\tilde{z}+\text{rand}\times 10^{-5})}, \quad (6.1)$$

where k is defined as the wavenumber of the light $\frac{2\pi}{\lambda}$. The vector $\vec{y}$ was used because it is identical to the necessary z -grid, since the target was tilted by 45° ,

$$z = y \tan(\theta) = y. \quad (6.2)$$

In Figure 6.1a, one can see the real original object, and in Figure 6.1b, one can see the real part of the complex bar target that now contains the same phase that a target planar tilted away from the detector by 45° would have. One other caveat here is that if this was a real target that was tilted away from the detector plane, it would appear to shrink in the vertical direction by $\sqrt{2}$. That shrinking is not important to keep track of in a simulation where the phase of the target is really what matters.

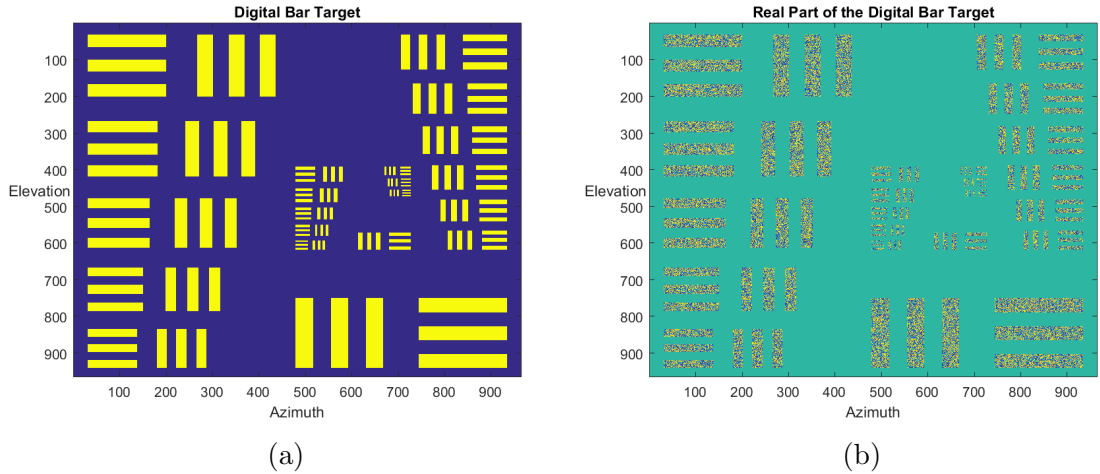


Figure 6.1: The left image is the image that I started with when creating a synthetic object. It has pixel values of one or zero, and is purely real. The right image is the real part of the complex bar target. The phase that was added accomplished two things. First it makes it appear as if the top of the bar target is tilted away from the observer by 45° , and secondly it adds a small random phase that makes it appear rough and so will scatter light diffusely.

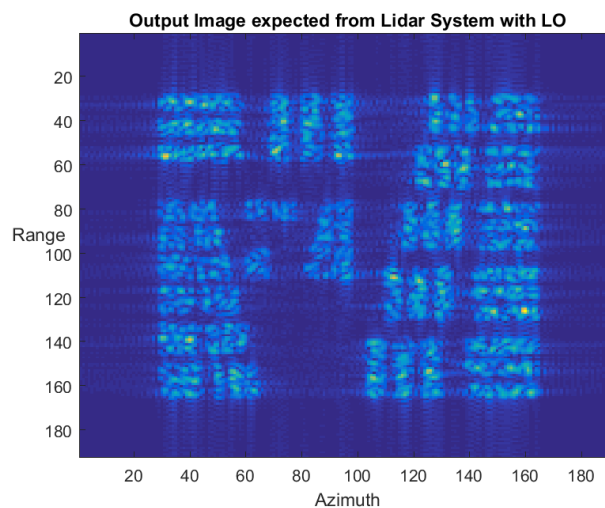


Figure 6.2: This is the result one would expect to see when imaging the target with a digital holographic method. It is just the 2D spatial Fourier transform of the downsampled spatial frequency content of the target. One can see the effect of the object speckle in this simulated image.

Simulated Standard Digital Holographic Output Image

Here I will show what a simulated standard digital holographic image would look like when imaging the object that was generated in the previous section. This is accomplished by taking a two dimensional spatial Fourier transform of the object, downsampling to the number of pixels which will be used in the actual experiment later, then taking an inverse Fourier transform. This will serve as a reference for the best a coherent imaging system can be, as one can assume that the spatial shearing reconstruction to be demonstrated will be of equal quality or slightly worse than a standard holographic image. [1]

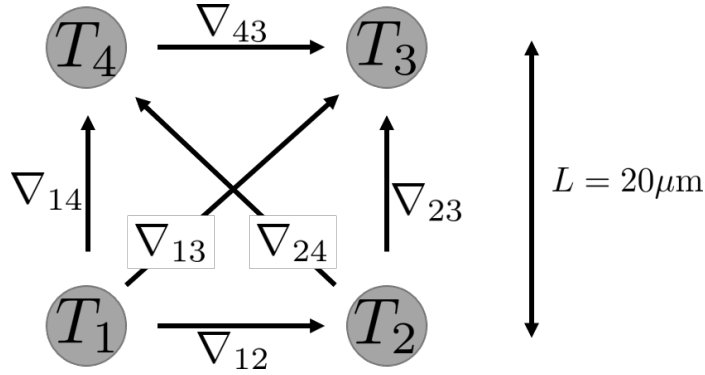


Figure 6.3: This is an illustration of the transmitter pattern where T_n is a transmitter, and ∇_{jk} is an indication of which shear is which, and in what direction.

Generating Synthetic Spatial Shears

To understand how to generate the synthetic spatial shears of the object's spatial frequency content, let us do a quick review of what information I am trying to capture from the target. The basic setup for spatial shearing is as follows. The target is an Air Force bar target, with the top tilted away from the transmitters and camera by 45° . The illumination beam is split into four beams, arranged in a square pattern with $20\mu\text{m}$ to a side. This is illustrated in Figure 6.3. More generally, a shift in space of one transmitter relative to another is referred to as a “shear”. If this spacing is chosen to be exactly the same size as a pixel width on the receiver array, it will be mathematically identical to a discrete phase derivative, along the direction of the shift, of the returned speckle from the object. So if each transmitter illuminates the target with the same spatial beam profile, but offset from each other spatially by one pixel, the return scattered light from the target will be four copies of the same speckle pattern on top of each other. These will then interfere with one another at the detector array, and represent measurements of the shears of the speckle in multiple different directions as indicated in Figure 6.3. To be somewhat more concrete, lets

think about one shear in particular, e.g. the one generated between T_1 and T_2 . Since the target will be illuminated with bare fiber tips, one can approximate the transmitters as point sources, so in the transmitter/detector plane the transmitters have the following definition. [5]

$$T_1(x, y, 0) \equiv \delta(x, y) \quad (6.3)$$

$$T_2(x, y, 0) \equiv \delta(x + \Delta x, y) \quad (6.4)$$

Where one can clearly see that T_2 is shifted relative to T_1 by Δx . Assuming that the target is in the far field, then the transmit beams will be plane waves at the target, i.e. their amplitude is varying slowly enough to ignore, but they have different phases. If the object is at a distance R from the detector plane the fields are

$$T_1(x, y, R) = 1 \quad (6.5)$$

$$T_2(x, y, R) = e^{i\frac{\Delta x}{\lambda R}x}. \quad (6.6)$$

Each of these fields interact with the target independently, by multiplication with the object's complex reflectivity function $O(x, y, R)$. If this product is back propagated to the detector plane, again under the Fraunhofer approximation, one can see that at the detector plane is the spatial frequency content of the target. The spatial frequency content of the target is what I refer to as "speckle" and for both transmitter cases above results in

$$S_1(x, y, 0) = \mathcal{F} \{T_1(x, y, R)O(x, y)\} = \mathcal{F} \{O(x, y)\} \quad (6.7)$$

$$S_2(x, y, 0) = \mathcal{F} \{T_2(x, y, R)O(x, y)\} = \mathcal{F} \left\{ e^{i\frac{\Delta x}{\lambda R}x} O(x, y) \right\} = S_1(x + \Delta x, y). \quad (6.8)$$

Where now the linear phase term that was acquired by the shifted transmitter has led to a shifted version of the speckle by that same amount. When the modulus square of the sum of these two fields is examined, the mixing term of interest can be seen.

$$I_{12}(x, y, 0) = |S_1 + S_2|^2 = |S_1|^2 + |S_2|^2 + S_1 S_2^* + S_1^* S_2 \quad (6.9)$$

$$= |S_1|^2 + |S_2|^2 + \nabla_{12} + \nabla_{12}^* \quad (6.10)$$

This is illustrated in Figure 6.4. The mixing terms can be recognized as a discrete measurement of a phase derivative along the direction of the shift. This mixing term is precisely what a diagonal of the covariance matrix is defined as. [17]

$$\begin{aligned} \nabla_{12} &= S_1(x, y) S_1^*(x + \Delta x, y) = a_1(x, y) e^{i\phi_1(x, y)} a_1^*(x + \Delta x, y) e^{-i\phi_1(x + \Delta x, y)} \\ &= a_1(x, y) a_1^*(x + \Delta x, y) e^{i(\phi_1(x, y) - \phi_1(x + \Delta x, y))} \end{aligned} \quad (6.11)$$

Which as was discussed in the previous chapter, can be used to reconstruct the phase of the original speckle. This mixing term can be isolated if a temporal frequency shift is used in conjunction with the spatial shift to push each of the mixing terms away from DC when examined in the temporal Fourier domain. This will be done experimentally using fiber delays and a frequency chirped laser, but for the sake of the simulation, one can just assume that each of the six shear terms have been correctly isolated.

In the simulation, the goal is to generate a shear of the speckle field for each of the mixing terms between all four transmitters. This leads to six shears, where the x and y shears are measured twice but from different beams so they are shifted relative to the other. These shears (and their assumed directions) are shown in Figure 6.3.

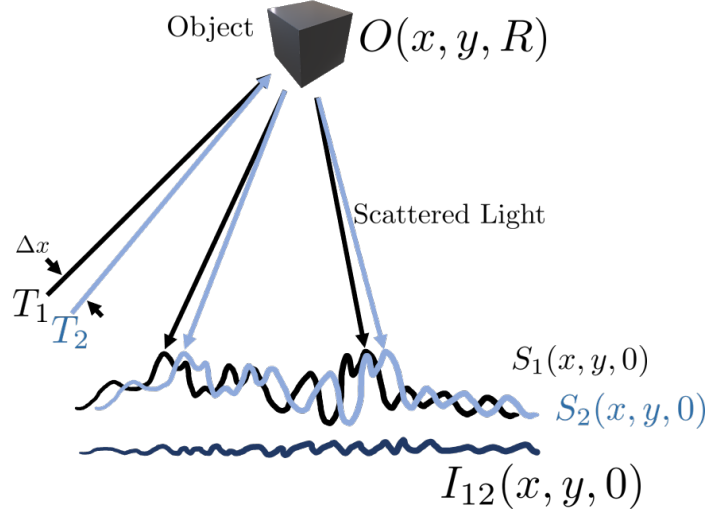


Figure 6.4: This is demonstrating how two spatially separated co-linear sources lead to overlapping copies of the scattered light on the detector plane, which when interfered with each other create four terms, one of which is the shear measurement in the x direction. Note that the spatial shift became a linear phase term at the object plane thanks to the Fourier Shift Theorem.

To create these in the simulation, one starts by taking the Fourier transform of the complex object. This gives the speckle at the detector plane, which is identical to the spatial frequency content of the target. Then use the Fourier shift theorem to shift a copy of it by one pixel (Δx and Δy) relative to the unshifted speckle. Then take the complex conjugate of the shifted speckle, and multiply it by the unshifted speckle point by point. This is then downsampled to match the number of pixels used in the experiment for ease of comparison later.

$$\nabla_{12} = S_1 S_2^* = S(x, y) S^*(x + \Delta x, y) \quad (6.12)$$

$$\nabla_{43} = S_4 S_3^* = S(x, y + \Delta y) S^*(x + \Delta x, y + \Delta y) \quad (6.13)$$

$$\nabla_{14} = S_1 S_4^* = S(x, y) S^*(x, y + \Delta y) \quad (6.14)$$

$$\nabla_{23} = S_2 S_3^* = S(x + \Delta x, y) S^*(x + \Delta x, y + \Delta y) \quad (6.15)$$

$$\nabla_{13} = S_1 S_3^* = S(x, y) S^*(x + \Delta x, y + \Delta y) \quad (6.16)$$

$$\nabla_{24} = S_2 S_4^* = S(x + \Delta x, y) S^*(x, y + \Delta y) \quad (6.17)$$

In the simulation, I chose to make the second speckle field the conjugate as a standard, as indicated by the directions in Figure 6.3. In practice the “direction” of the shear is unknown, so this is only a convention in the simulation. In the experiment I will need to know the “direction” of the shears otherwise the reconstruction will fail. This problem will be approached in the next chapter.

Simulated shears of the speckle field have now been generated. The next step would be to attempt to reconstruct the phase of the speckle using these shears. One problem in the experiment is that there is always a random phase offset on each shear relative to each other due to small distance differences between the paths. This will cause the reconstruction to fail. In the next section, a method for removing this unwanted phase offset from each shear will be discussed.

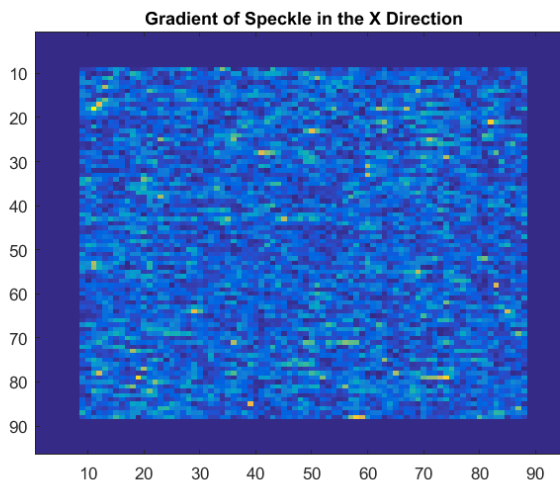


Figure 6.5: This is a simulated shear, where the speckle field of the target was added to a conjugated copy of the speckle field that was shifted in the x direction by Δx . This looks very similar to the speckle of the object, but it is not possible to recover an image from this alone, as it is a discrete derivative of the speckle phase.

Phase Closure Algorithm

The premise behind phase closure, is that the speckle field is a continuous and smooth complex function that contains no poles. That means that if one integrates over some path that starts and ends in the same place, there should be zero phase difference between the start and the end, regardless of the path taken. This is a technique that was first utilized in solving for phase corrections in astronomical interferometry experiments [18]. Phase closure can be used to solve for the phase offsets of the shears. The far left triangle in Figure 6.6 will be the first example. Here one starts at the bottom left, uses a shear in the x direction to move to the right by one unit, then a shear in the y direction to move up by one unit, and then the negative diagonal shear to get back to the beginning of the loop. The diagonal shear is negative because the step is in the opposite direction that it was defined in. This

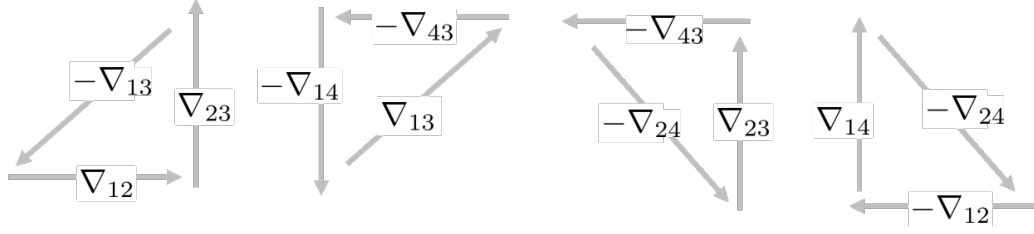


Figure 6.6: These are four possible loops that can be built to calculate the phase offset of each shear. The leftover phase after going around the loop is used to calculate the phase offset of each shear that would yield those leftover phases.

process can be written as,

$$\angle \nabla_{12} + \angle \nabla_{23} - \angle \nabla_{13} = P_{\text{close}_1} \quad (6.18)$$

where the angle symbols indicate taking the phase of the shear, and P_{close_1} is the resulting “leftover” phase from this particular loop. If the shears had the proper phases without errors, P_{close_1} would be equal to zero. This process is repeated for the four loops shown in Figure 6.6, which will yield a system of equations, which can be put into a matrix equation.

$$\mathbf{A}p = P \quad (6.19)$$

$$\begin{bmatrix} 1 & 0 & -1 & 0 & 0 & 1 \\ 0 & -1 & 1 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 & -1 & 1 \\ -1 & 1 & 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} \Delta\phi_{12} \\ \Delta\phi_{14} \\ \Delta\phi_{13} \\ \Delta\phi_{24} \\ \Delta\phi_{43} \\ \Delta\phi_{23} \end{bmatrix} = \begin{bmatrix} P_{\text{close}_1} \\ P_{\text{close}_2} \\ P_{\text{close}_3} \\ P_{\text{close}_4} \end{bmatrix} \quad (6.20)$$

Where all of the $\Delta\phi_{jk}$ terms are the unknown phase errors that are being solved for. To find a vector containing the phase offsets, the *mldivide* command in Matlab is used, which solves for a solution to this matrix equation. The *mldivide* command is used because this system of equations is under-determined, which means that it will not perfectly correct the phases, but get all of the shears to the same offset phase.

$$p = \mathbf{A} \backslash P \quad (6.21)$$

Now that I have solved for the phase offsets of the original shears, they can be subtracted from the shears. In the simulation, I put random phases on the shears and show that the reconstruction fails unless the phase closure algorithm is used. This algorithm will be demonstrated later in the chapter after the reconstruction algorithm is better understood.

Phase Solution via Maximum Likelihood Eigenvector Method

At this point, the shears that are being simulated contain random phase offsets that are expected in the experiment, the next step is to plug them into a covariance matrix to find the phase solution for the spatial frequency content of the object. The mixing terms between each transmitter will still be placed into diagonals of the covariance matrix, but now their locations need to be determined because the two dimensional shear measurements have to be turned into vectors. [17] This can be seen in Figure 6.7, where if the goal is to compare pixel number 2 to pixel number 97, one would need to put it 95 diagonals away and also in the right place in a covariance matrix. It is crucial to get the correct shear into the correct diagonal of the covariance matrix, because the covariance matrix defines the relationships between each shear measurement, and as a whole gives the phase of the speckle that I want to retrieve.

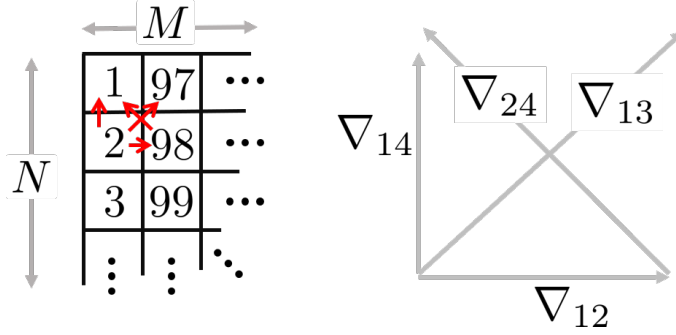


Figure 6.7: This is how Matlab turns a matrix into a vector, by taking each column and stacking them. The red arrows show the direction of the shears as defined by Equations 6.17.

Thinking about Figure 6.7, one can write a simple formula to help determine which diagonal to put what shear in. Where $N \times M$ is the size of the two dimensional shear. The way Matlab linearizes matrices is by “stacking” the columns on top of each other. That is what is being illustrated with the numbers in the boxes in Figure 6.7, where the N dimension’s size is 96.

$$\text{Diag}(\nabla_{14}) = 2 - 1 \Rightarrow \text{Diag}(\nabla_{14}) = \pm 1 \quad (6.22)$$

$$\text{Diag}(\nabla_{12}) = 98 - 2 = 96 \Rightarrow \text{Diag}(\nabla_{12}) = \pm N \quad (6.23)$$

$$\text{Diag}(\nabla_{13}) = 97 - 2 = 95 \Rightarrow \text{Diag}(\nabla_{13}) = \pm(N - 1) \quad (6.24)$$

$$\text{Diag}(\nabla_{24}) = 98 - 1 = 97 \Rightarrow \text{Diag}(\nabla_{24}) = \pm(N + 1) \quad (6.25)$$

All of the simulations and experiments used square field regions, so the covariance matrix is also a square matrix that is N^2 by N^2 . To construct the covariance matrix, take each two dimensional shear, turn it into a vector, then put it into its corresponding diagonal in a sparse covariance matrix. Where the upper diagonal is the original shear, and the lower diagonal is the complex conjugate of the shear. This

would look something like this,

$$\mathbf{C} = \begin{matrix} & \begin{matrix} 1 & 2 & \dots & 94 & 95 & 96 & 97 & \dots & \dots & N^2 \end{matrix} \\ \begin{matrix} -1 \\ -2 \\ \vdots \\ -94 \\ -95 \\ -96 \\ -97 \\ \vdots \\ \vdots \\ -N^2 \end{matrix} & \left[\begin{array}{cccccccccc} 0 & \nabla_{14}^{(1)} & 0 & \dots & 0 & \nabla_{13}^{(1)} & \nabla_{12}^{(1)} & \nabla_{24}^{(1)} & 0 & 0 \\ \nabla_{14}^{*(1)} & 0 & \nabla_{14}^{(2)} & 0 & \dots & 0 & \nabla_{13}^{(2)} & \nabla_{12}^{(2)} & \ddots & 0 \\ 0 & \nabla_{14}^{*(2)} & 0 & \nabla_{14}^{(3)} & 0 & \dots & 0 & \nabla_{13}^{(3)} & \ddots & \nabla_{24}^{(N^2)} \\ \vdots & 0 & \nabla_{14}^{*(3)} & 0 & \nabla_{14}^{(4)} & 0 & \dots & 0 & \ddots & \nabla_{12}^{(N^2)} \\ 0 & \vdots & 0 & \nabla_{14}^{*(4)} & 0 & \nabla_{14}^{(5)} & 0 & \dots & 0 & \nabla_{13}^{(N^2)} \\ \nabla_{13}^{*(1)} & 0 & \vdots & 0 & \nabla_{14}^{*(5)} & 0 & \nabla_{14}^{(6)} & 0 & \dots & 0 \\ \nabla_{12}^{*(1)} & \nabla_{13}^{*(2)} & 0 & \vdots & 0 & \nabla_{14}^{*(6)} & 0 & \nabla_{14}^{(7)} & 0 & \vdots \\ \vdots & \nabla_{24}^{*(1)} & \nabla_{12}^{*(2)} & \nabla_{13}^{*(3)} & 0 & \vdots & 0 & \nabla_{14}^{*(7)} & 0 & \ddots \\ 0 & \ddots & \ddots & \ddots & 0 & \vdots & 0 & \ddots & 0 & \nabla_{14}^{(N^2)} \\ 0 & 0 & \nabla_{24}^{*(N^2)} & \nabla_{12}^{*(N^2)} & \nabla_{13}^{*(N^2)} & 0 & \dots & 0 & \nabla_{14}^{*(N^2)} & 0 \end{array} \right] \end{matrix} \quad (6.26)$$

Where the superscripts indicate which element of the vectorized shear is in that location of the matrix, and the labels on the outside of the matrix correspond to a diagonal of the covariance matrix. This is accomplished in Matlab using a sparse matrix and the *spdiags* command to place the vectorized shears into the sparse matrix. Once this covariance matrix is constructed, the next step is to solve for the eigenvector that corresponds to the largest eigenvalue. [6, 17] As was discussed in the previous chapter, this eigenvector will be the one most closely related to the phase of the speckle that created the shears. Defining the covariance matrix as $\mathbf{C}$ the eigenvalue equation is,

$$\mathbf{C}\vec{\nu} = \lambda\vec{\nu} \quad (6.27)$$

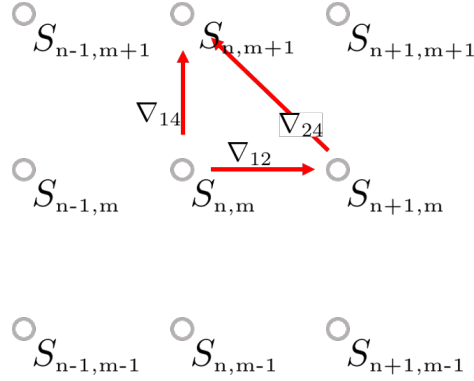


Figure 6.8: This is the grid of points in the two dimensional shears. I wish to know the amplitude of the point in the middle, but all I have are mixing terms that contain the amplitude of the middle point and one other point.

where if the eigenvector $\vec{\nu}$ is changed back to a two dimensional matrix, with grid point locations defined by the subscripts n and m , one gets ν_{nm} . Then take the angle to get the phase solution.

$$\text{Speckle Phase} = e^{i\angle \nu_{nm}} \quad (6.28)$$

To reconstruct an image, the phase and the amplitude of the speckle is needed. So the next step is to solve for the amplitude information. This will be explained in the next section.

Amplitude Solution

This section will demonstrate how to calculate the amplitude of the speckle from multiple shifted versions of the four shears used in the previous section for reconstructing the phase. To begin to understand how the amplitude solution works, I will start with a simple case of three shears, and then use that pattern to create a more robust amplitude solution. [6] In Figure 6.8, one sees a grid of points, where

each point is the speckle field that we are seeking to recover. It is easier to think about solving for the amplitude of one point, then later just interpret the result over the whole grid of points. So the fundamental problem to solve is that the shears only contain measurements of mixing terms, where each measurement contains information about the product of one amplitude in one location and the amplitude of the nearest neighbor. Since we are seeking only the amplitude of the speckle field, all of the math done here will be done with the absolute value of the shears.

$$|\nabla_{12}| = |S_{n,m} S_{n,m+1}^*| = |a_{n,m} e^{i\phi_1} a_{n,m+1} e^{-i\phi_2}| = a_{n,m} a_{n,m+1} \quad (6.29)$$

$$|\nabla_{14}| = |S_{n,m} S_{n+1,m}^*| = |a_{n,m} e^{i\phi_1} a_{n+1,m} e^{-i\phi_4}| = a_{n,m} a_{n+1,m} \quad (6.30)$$

$$|\nabla_{24}| = |S_{n+1,m} S_{n,m+1}^*| = |a_{n+1,m} e^{i\phi_2} a_{n,m+1} e^{-i\phi_4}| = a_{n+1,m} a_{n,m+1} \quad (6.31)$$

$$|\nabla_{13}| = |S_{n,m} S_{n+1,m+1}^*| = |a_{n,m} e^{i\phi_1} a_{n+1,m+1} e^{-i\phi_3}| = a_{n,m} a_{n+1,m+1} \quad (6.32)$$

By inspection, here is an equation that can be used to solve for the amplitude of the middle point.

$$\sqrt{\frac{|\nabla_{12}| |\nabla_{14}|}{|\nabla_{24}|}} = \sqrt{\frac{a_{n,m} a_{n,m+1} a_{n,m} a_{n+1,m}}{a_{n+1,m} a_{n,m+1}}} = a_{n,m} \quad (6.33)$$

That is take the shear in the x direction, multiply it by a shear in the y direction, and then divide by the diagonal shear that connects the two, as shown in Figure 6.8. This cancels any amplitude terms that are not the middle point.

Following the pattern of multiplying two shears that are touching the middle, and then dividing them by a shear that connects them, one can come up with a solution for the amplitude that takes into account all of the possible combinations of the four shears used for the reconstruction. In Figure 6.9, one can see that the “spokes” are the diagonal, horizontal and vertical shears (that may or may not be

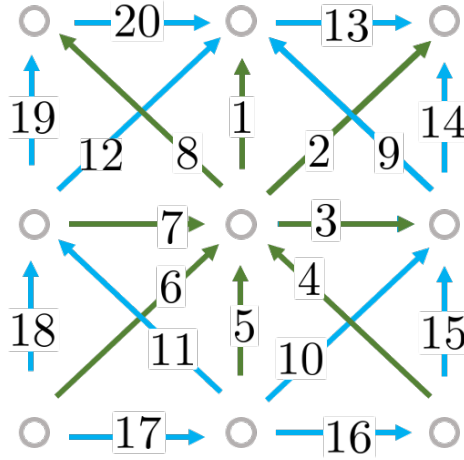


Figure 6.9: This is a visual way to think about how to use all twenty of these shears to solve for the amplitude of the middle grid point. The “spokes” are shown in green and are numbered one through eight. The “connectors” are shown in blue, and connect one end of a spoke to another end of a spoke, numbered nine through twenty.

shifted) which all have one end touching the central field point. These are shown in green, and numbered one through eight. The “connectors” are the shears that are not touching the middle point, but connect two spokes, and are numbered nine through twenty. If Equation 6.33 is generalized to include all possible combinations of spokes and connectors, the following expression is found. This equation solves for the amplitude of the field at the central location of the grid.

$$a_{nm} = \sqrt{\frac{g_1(g_8 + g_2 + g_3) + g_3(g_2 + g_4 + g_5) + g_5(g_4 + g_6 + g_7) + g_7(g_6 + g_8 + g_1)}{\sum_{i=9}^{n=20} g_n}} \quad (6.34)$$

Where for example,

$$g_7 = |\nabla_{12}(n-1, m)| \quad (6.35)$$

is a shifted version of $\nabla_{12}(n, m)$ to the left by one pixel. The rest of g_n are the

absolute values of ∇_{12} , ∇_{14} , ∇_{13} , and ∇_{24} shifted to the locations seen in Figure 6.9. Now that there is a solution for the amplitude of one point, it is easy to see how it can be generalized to a situation where each of the shears is a 2D array of shear measurements. By using this method one can solve for the amplitude of the unknown speckle field, even though only mixed terms, which are products of amplitudes, are measured.

Reconstructing an Image

Just as a reminder, the whole point of this is to solve for the amplitude and phase of a speckle field of an object that has propagated to the far field. Once the amplitude and phase have been solved for, all it takes to retrieve an image of the object is a two dimensional Fourier transform, and a variable substitution. The purpose of measuring the shears of the speckle instead of the speckle directly is because atmospheric turbulence will scramble the phase of a light field. Shears measure phase differences between copies of the light field that all went through the same turbulence. Those phase differences can then be used to back out the actual amplitude and phase of the target, but now without the phase aberrations due to the atmosphere which is common mode. [6] The previous two sections discussed how to find the phase and amplitude of this field.

$$\text{Amplitude} = a_{nm} \tag{6.36}$$

$$\text{Phase} = e^{i\angle\nu_{nm}} \tag{6.37}$$

Now all that is left to do to synthesize the image is to combine the two, and “back propagate” it to the object domain using the Fraunhofer propagation operator.

$$\text{Image} = \mathcal{F}^{-1}\{a_{nm}e^{i\angle\nu_{nm}}\} = \mathcal{F}^{-1}\{\mathcal{F}\{\text{Object}\}\} \quad (6.38)$$

This is similar to a Fourier transform because the detector is in the far field relative to the target, and the free space propagation of light is identical to a Fourier transform with a change of variables, under the Fraunhofer approximation. [1] When combined and Fraunhofer propagated, the final image can be seen in Figure 6.10. Where the

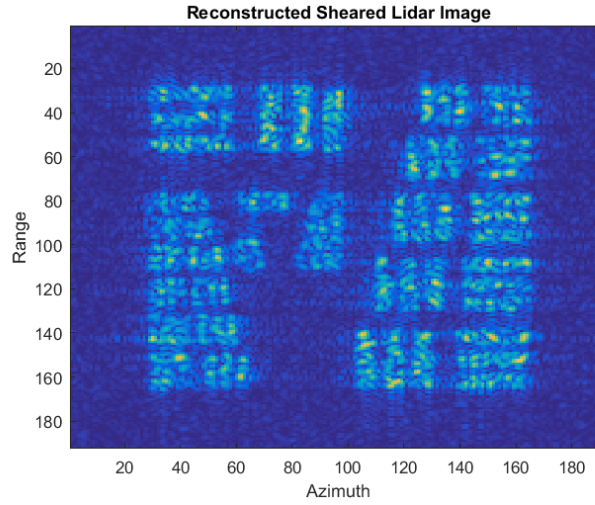


Figure 6.10: This is the result of multiplying the amplitude solution with the phase solution and then back propagating to the detector plane, which is identical to taking the Fourier transform with a change of variables, as seen in Equation 6.38.

amplitude and phase have been combined before taking the Fourier transform. This simulated output image is nearly identical to the simulated digital holography image in Figure 6.2. This is to demonstrate that in the simulation spatial shearing lidar is qualitatively just as good as another more common type of coherent imaging. This is true as long as the SNR is high and all of the shears are measured perfectly.

Phase Closure Demonstration

So the simulation can reconstruct an image from spatial shears, the next step is to test its ability to reconstruct if a random global phase is added to each shear. The idea here is to validate the phase closure algorithm, as it will be necessary in the experimental data. The shears are generated in the same way as was discussed earlier in the chapter. Take the speckle, shift it, complex conjugate it, then multiply it with an unshifted version of the speckle. This will be expressed as some field,

$$\nabla_{jk} = b_{nm}e^{i\phi_{nm}}. \quad (6.39)$$

When these shears are used as the input to the phase and amplitude reconstruction algorithms outlined earlier, the reconstructed image is identical to Figure 6.10. So clearly the reconstruction works if there are no random phases added to the shears.

Now add a random global piston phase to each simulated shear, $\Delta\phi_{jk}$, between 0 and 2π ,

$$\nabla_{jk} = b_{nm}e^{i(\phi_{nm}+\Delta\phi_{jk})}. \quad (6.40)$$

Where the subscripts n and m indicate individual locations within a two dimensional object, and subscripts jk indicate how this particular spatial shear is defined. So in this context, ϕ_{jk} is a global phase that is applied over all of the elements of ∇_{jk} , but is unique to that particular shear, and different for the other shears. Figure 6.11 shows the reconstructed image with random global offsets added to each shear, when nothing was done to attempt to correct for the random phase offsets. One can see this in action in Figure 6.12, where P_{close_1} was plotted for each point. These plots show the results of Equation 6.18, where I looked at the sum of the phases around a

loop, and used the “leftover” phase to calculate the phase offsets on the shears. The left side shows what each point gives as a result for the “leftover” phase around a triangular path. One can see that the phase is overall zero, which means that there is no phase offset on the shears, which is true by design. The right image is the “leftover” phase around the same loop, but now the shears that were used to create the loop each had a global piston phase offset, which results in a global phase shift of the “leftover” phase. To define this “leftover” phase, simply take an average of the phase in Figure 6.12b, and use that as an input to Equation 6.20. Once an average result for each of the phase closure loops is measured, one can solve equation 6.20 for the phase offsets on the shears. The last step is to just subtract these calculated phase offsets from the shears.

$$\nabla_{jk} = b_{nm} e^{i(\phi_{nm} + \Delta\phi_{jk} - \phi_{\text{correction}_{jk}})} \quad (6.41)$$

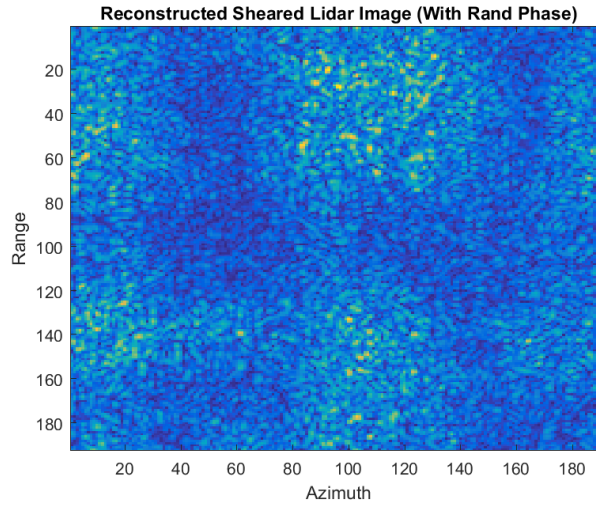
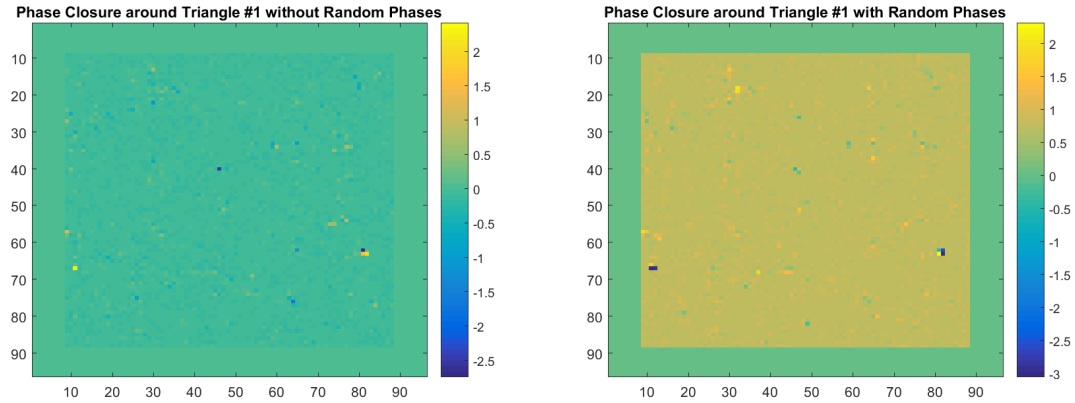


Figure 6.11: This is the reconstructed image when there are random global phase offsets on the shears that were input into the phase and amplitude reconstruction algorithms. Clearly these random phases have resulted in a failed reconstruction, and this motivates the need to find a way to remove the random phase offsets from the shears.



(a) Phase closure without random phases. (b) Phase closure with random phases

Figure 6.12: The image on the left is what the phase closure around the first triangular loop looks like without any random phase offsets on the shears. It is evident that there is not any “leftover” phase when there are no random phase offsets. The image on the right is the same phase closure, but where each shear had a different piston phase offset, which led to an overall phase shift by the “leftover” phase.

Once the correction phases are subtracted from each shear, one can put the shears into the amplitude and phase reconstruction algorithms. The resulting reconstruction of the corrected simulated image can be seen in Figure 6.13. One can see in Figure 6.13 that the image looks as expected, and is at least as good of quality as the first reconstruction, Figure 6.10, that did not have any random phases added to it. The image is circularly shifted in the image domain. The primary reason for this shift is that when I correct the phase, I am correcting the overall phase of the shears relative to each other. This means that after the correction all of the shears can still contain a common mode global piston phase shift. This leads to a shift because the shears have constant phases. The shears are put into an algorithm that essentially “integrates” them turning this constant phase term into a linear phase term. The last step is to take a Fourier transform of this integrated thing to get an image, which, by the Fourier Shift Theorem, leads to a shifted image. Unfortunately, I found no simple

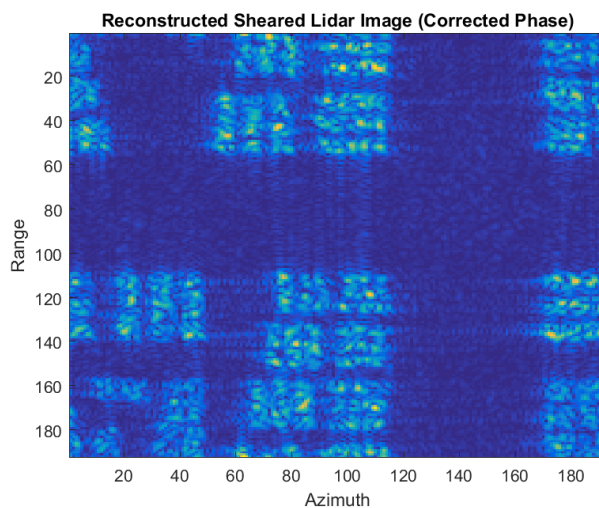


Figure 6.13: Here one can see a successful image reconstruction, where the shears used to reconstruct this image were given random phase offsets, then the phase closure algorithm was used to remove them. The only issue here is that the final image is shifted circularly in the image domain, which is a result of the fact that the phase closure algorithm can only correct for global phase differences between the shears, which will leave all the shears with the same phase offset instead of different offsets.

way to remove this shift, as it is random for each random set of phases that were added to the shears, and the main way to correct it is to manually shift it after image formation.

This Reconstruction Algorithm is Very Sensitive

This section will explore the myriad of ways that reconstruction can fail, which is ultimately the reason for creating this simulation, as it was necessary to know all of the possible ways that the experimental data may need to be manipulated in order to reconstruct an image. This section will talk about simulated shears that contain random global phase offsets, and so will require the implementation of the phase closure algorithm. If the experimental SNR is also modeled, this simulation most closely resembles the experimental data.

Incorrect Sign on Phase Closure Loops

In the simulation, I chose the “direction” of the shears. I defined that the starting point is not conjugated, and the ending point is conjugated. This is how I decided what the sign needed to be on the phase closure loops. But supposing that I do not actually know the “direction” of the shears, I would have to guess the sign of each shear before I can attempt to create continuous loops. If I use the loop illustrated in Figure 6.14, it would yield an incorrect value for the “leftover” phase that is needed to calculate the correction phases on the shears. If even one of the loops has an element with an incorrect sign, the reconstruction fails because the correction phases are calculated from incorrect closure loops. This can yield an image similar to Figure 6.11. This is one of the easiest places to cause the reconstruction to fail, and I will go into slightly more detail on how to overcome this challenge in the experimental data in the next chapter.

Arbitrary Conjugate Shears

When I put the shears into the covariance matrix to solve for the phase solution, I put the shears in the upper diagonals, and the complex conjugates of those same

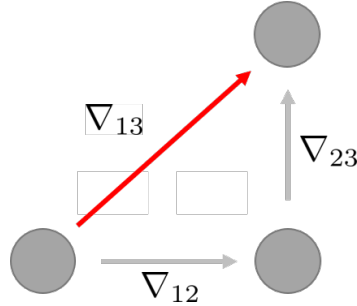


Figure 6.14: This is the first loop used in the phase closure algorithm, but with an incorrect sign on the diagonal shear, in red, which illustrates the idea of having the wrong sign on the phase of the shear leading to an incorrect phase closure value.

shears in the lower diagonals. In the experiment I do not know the “direction” of the shears, so I do not know if the shear I measured should go into the upper or lower diagonal. This is tested by conjugating the shears prior to putting them into the covariance matrix construction code. So if I am using four shears, I have sixteen possible conjugate combinations to try the reconstruction with. If I get one incorrect the reconstruction fails, and usually will come out looking like completely random noise, or at best will look similar to Figure 6.11. So I have to test all sixteen of these conjugate combinations in the experimental data to look for the image. The good news, is that once the correct conjugation combination has been found for a particular experiment, it will not need to be solved for again.

Incorrect Diagonal in the Covariance Matrix

The covariance matrix method is unique because it compares a shear measurement in one location, to all other shear measurements, and it is this unique relationship that is used to define the speckle field that would have generated these shears. If a shear is put in the wrong diagonal, that matrix describes a different system, where the shears are related in some other way than in the experiment. So if ∇_{12} is placed into the 97th diagonal, instead of the 96th, the reconstruction fails. In that example one would see something similar to Figure 6.11, where one might be able to make out a very blurry version of the image, it would still be a far cry from a correct image like Figure 6.10. This one is a little bit easier to determine because it is possible to use knowledge of the geometry of the experiment to derive where the diagonals should be put into the covariance matrix, just as was outlined in Section 6, Equation 6.20.

Incorrect Shear Size

Incorrect shear size is fundamental to the experiment. In the simulation it is simple to define the shears as having exact shifts of Δx and/or Δy , where both are defined to be exactly one pixel in size. However, since the Fourier Shift Theorem was used to perform the shifting, the simulation can explore a non integer pixel shifts. If this shift is chosen to be one and a half pixels, even in one dimension, the reconstruction fails. This is similar reasoning to the previous section, where if I wanted to use a shear other than one pixel, I would need to put the shears into different diagonals. Shears of size two could potentially work, but having a shear of one and a half pixels would require putting the shear “between” two diagonals of the covariance matrix, which one cannot do, so it results in a failed reconstruction. The implication of this is that the experiment must have the spatial shears just right, as in a single pixel shear, for the reconstruction to work correctly. There is a method that can resample the shears to make them have a shear of one pixel, but only if you know the actual size of the real shear.

In the next chapter I will introduce the experiment that was performed to validate spatial shearing lidar, the challenges that were overcome, and some slight additions to the processing algorithm that were implemented.

SPATIAL SHEARING LIDAR EXPERIMENT

This chapter will describe the experimental setup that was used to validate the spatial shearing lidar technique. I will begin with an overview, and each section will go through different portions of the setup or processing.

Figure 7.1 shows the overall experimental setup. An optically chirped fiber coupled laser was used as the light source. This was a Bridger Photonics laser with a chirp bandwidth $BW = 50 \text{ GHz}$, and a chirp rate of $\gamma = 1 \text{ THz/s}$. It was first split into four paths using three fiber splitters. Each path was time delayed using a different fiber length. The reason for these time delays is to demultiplex the spatial shears, but I will go more into the details of that in a later section. Each path was then transmitted to the target from a different location in a square pattern (the same one that was used for the simulation in Figure 6.3). [4, 5]

Each transmitter flood illuminated the object, creating four overlapping speckle

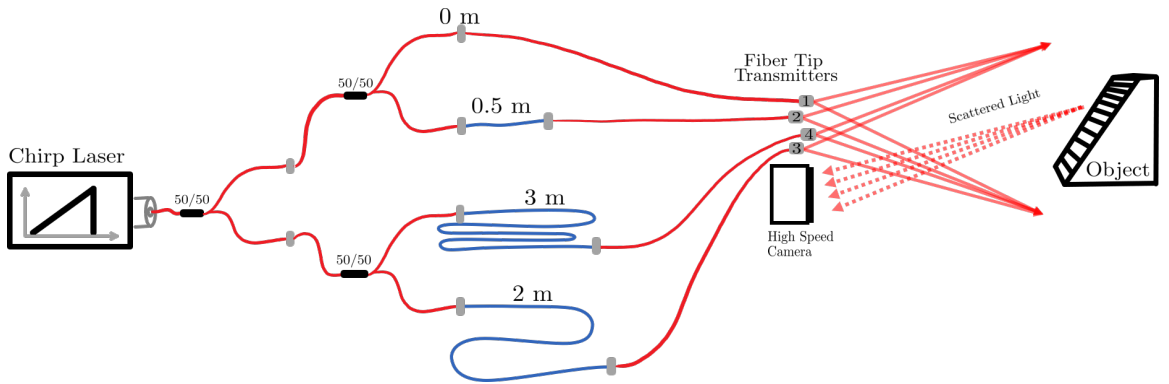


Figure 7.1: Starting with a fiber coupled chirped laser, the laser is split into four paths. Each path gets a unique fiber delay and then exits a fiber tip to flood illuminate the object. The transmitters are arranged in a square pattern that is in the plane of the detector. The scattered light contains four copies of the objects speckle, shifted by the same amounts as the transmitters. The spatial-temporal interference between the four speckle patterns is then recorded on a high speed camera.

patterns at the detector plane. The same speckle pattern is generated for each transmitter, but each is shifted by the location of that respective transmitter. These shifted copies of the speckle interfere with each other on the camera detector. The camera captured at 39,840 frames per second, and the region of interest was 16x80 pixels. Each pixel was $20\text{ }\mu\text{m}$ square, and the integration time per frame was $10\text{ }\mu\text{s}$. The raw data output from one “shot” is a datacube of 16x80 pixels, by 1714 frames in time. The limitation on the number of frames, taken was due to the finite length of the chirp in time (43 ms), which is what is meant by a “shot”.

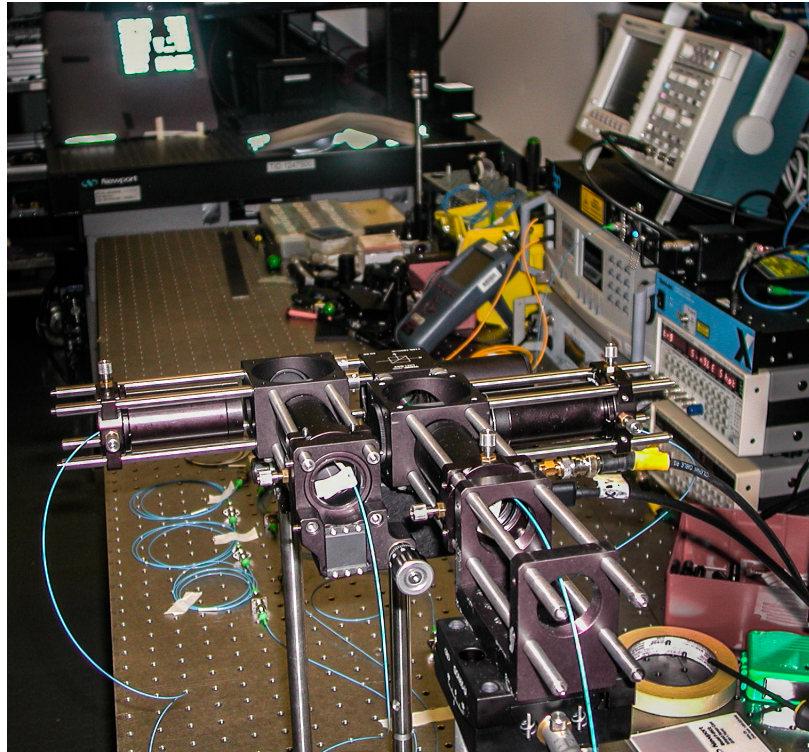


Figure 7.2: Here is a picture of the experimental setup. The object is the bright retro tape Air Force bar target on the top left, which is bright because of the flash on the camera. The spatial shearing setup is at the bottom in the cage mounts. The high speed camera is directly underneath the spatial shearing setup.

Spatial Shearing Experimental Setup

To achieve the precise spatial shifting of the transmitters was tricky. It involved three cage-mounted, non-polarizing beam-splitting cubes, and four cage-mounted fiber-tip holders each with x , y , and z translation capabilities. Figure 7.3 shows a schematic of how this was accomplished. For this alignment a non chirped CW laser was used, to see a static interference pattern between pairs of transmitters.

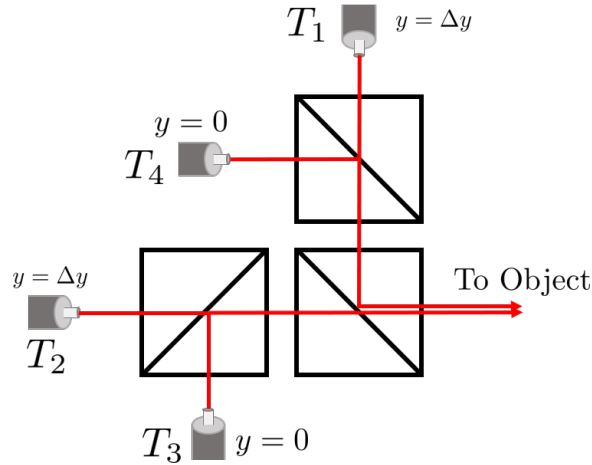


Figure 7.3: Three NPBS cubes were used in cage-mounts with the fiber tips as shown, which were mounted on xyz cage-mounts, to create the precise transmitter arrangement that is required for spatial shearing lidar. This is a three dimensional configuration, so to illustrate the different elevations of transmitters 1 and 2, they are shown to be at a location Δy above transmitters 3 and 4.

The process of getting the fiber tips into the correct location started by centering each fiber tip in its cage mount with the x - y stages, then estimating the z location for each. I then placed a large flat sheet of retro fabric down range and looked at the spatial interference pattern between transmitters. I looked at pairs of transmitters while changed their x , y and z locations until I saw a single “fringe” across the whole target, and iterated this until all four of the transmitters were in the “same” x , y , and z locations. I then put strips of black tape on the retro sheet spaced

by the half wavelength of the spatial fringe pattern one would expect to see when two transmitters were shifted by $20\mu\text{m}$ in x and in y . I then blocked two of the transmitters, and shifted one of the remaining two vertically until I saw the fringe period I wanted on the target. Then I blocked one of those two and unblocked one of the other two, and repeated the procedure in x and y until all four transmitters were where they should be, based on the far field interference pattern. This was a fairly meticulous process that involved many double checks to make sure that the transmitters were within $\pm 2\mu\text{m}$ of the correct shear size of $20\mu\text{m}$.

Demultiplexing Spatial Shears using Frequency Offsets

This section will go over how to simultaneously measure all six spatial shears with a high speed camera. The section title says that frequency offsets were used, which is true, but really this is because of a special property of linear chirps. The chirp allows for the use of time delays as frequency shifts, with this equation: [10]

$$\Delta f = \gamma\tau, \tag{7.1}$$

where γ is the chirp rate (think slope of the chirp) and Δf is the frequency shift, which is due to a time delay τ . If one interferes two chirps together with one time delayed relative to the other by τ , it manifests as a temporal beat note at the corresponding frequency Δf . This is the property used to demultiplex the spatial shears from each other, using the temporal offsets built into the transmitters. In Figure 7.1, one sees that T_1 has a fiber delay of 0 m (i.e. none), T_2 has a delay of 0.5 m, T_3 a delay of 2 m, and T_4 a delay of 3 m. These shift the chirps in time relative to each other. Keeping track of the shifts of transmitters 2 through 4 relative to transmitter 1, produces the time delays τ_{12} , τ_{13} , and τ_{14} , where τ_{12} is the delay between the first and second chirp. This can be seen in Figure 7.4. The time delays were chosen specifically to yield

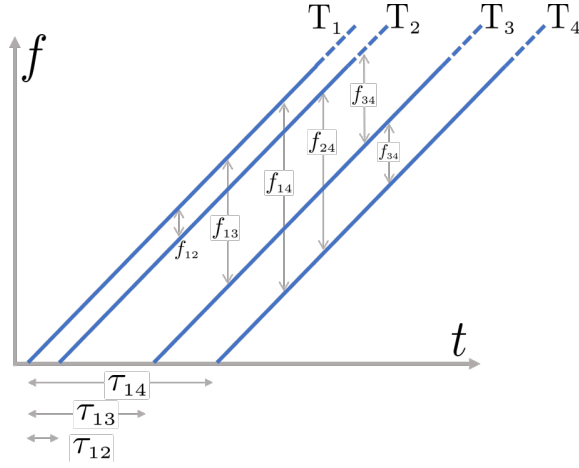


Figure 7.4: Here one can see the four transmit chirps, shifted relative to each other in time, which leads to six interference beat notes between them, shown as f_{jk} .

non-redundant beat notes between the chirps when they interfere with each other. This is to say that the temporal beat note between T_1 and T_2 is different than any other combination of two transmitters. The following equation is useful for keeping track of which time delay corresponds to what beat note frequency.

$$f_{jk} = \tau_{jk}\gamma \quad (7.2)$$

Where again, j and k correspond to interference of the j th chirp with the k th chirp. The other caveat is that these temporal beat notes are within the bandwidth of the camera, so the largest interference needs to be less than the Nyquist sampling frequency of the camera.

$$f_{14} < F_{\text{nyq}} = \frac{39,840}{2} \quad (7.3)$$

Using this as the upper bound on the frequency of the beat note, in conjunction with the chirp rate finds the fiber delay that will generate f_{14} as

$$\text{Delay Length}_{\max} \equiv d = \frac{f_{14}c}{n\gamma} = 3 \text{ meters}, \quad (7.4)$$

where $n = 1.5$ is the index of refraction of the fiber delay, c is the speed of light, and γ is the chirp rate. To be fair, I wanted to end up with fiber delays that were integer values of meters (or half integers) so that I did not have to order or make custom fiber lengths, so I chose the chirp rate to give me a fiber delay length that would utilize as much of the bandwidth of the camera as possible while still giving half meter fiber delay lengths.

Figure 7.5 illustrates the temporal frequency domain of the signal. This is the Fourier transform of the interference of the four transmitters with the fiber delays stated earlier. Six equally spaced peaks in the temporal frequency domain are seen on either side of the DC peak. Another reason to have the peaks spread out in the

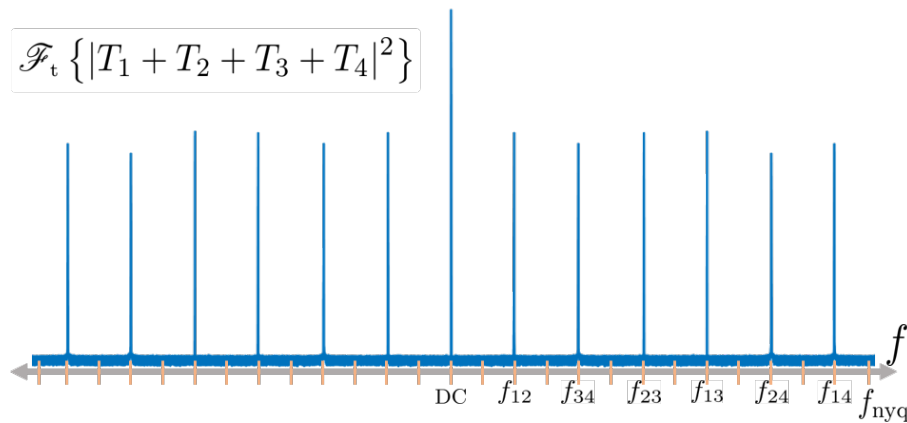


Figure 7.5: This is the temporal Fourier transform of the intensity distribution measured from the four transmitters. The fiber delays cause temporal beat notes between each pair of transmitters, which leads to six equally distributed peaks on either side of the DC peak in the temporal Fourier domain.

frequency domain is when each chirp interacts with the target, it will pick up time delays depending on the depth of the target. This puts a bound on the “width” of the interference peak, depending on the round trip depth of the target. The peaks need to be spaced enough so that the target information contained in one peak does not spread to the neighboring peak. If one looks at Figure 7.5, it can be seen that the maximum “width” of each peak, including the DC peak, is one thirteenth of the camera’s total framerate.

$$\text{Width}_{\max} \equiv l_{\max} = \frac{39,840}{13} \quad (7.5)$$

I can then write an expression for the maximum target depth, which is measured as the distance between the closest point to the farthest point on the target,

$$R_{\max} = \frac{\frac{l_{\max}}{2}c}{2\gamma n_{\text{air}}}, \quad (7.6)$$

where the two in the denominator indicates that I am using the round trip time delay corresponding to the target’s depth R . This is a lot of constraints on the object depth, and it can be tricky to get all of the parameters to work together. A lot of the planning of this experiment went into this type of thought process, where choosing the fiber delays, chirp rate, target depth, and camera bandwidth were all important and interconnected.

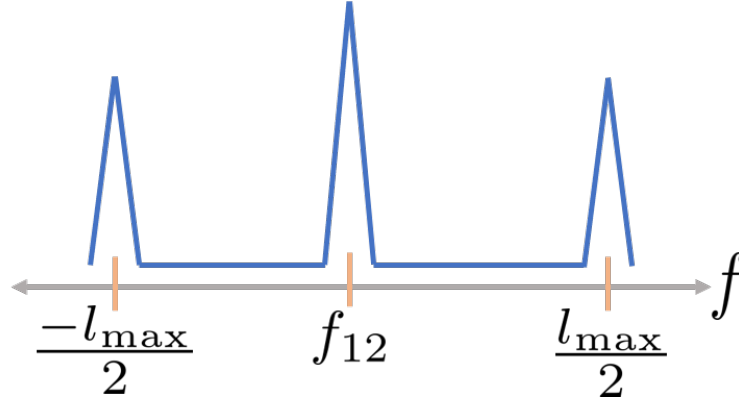


Figure 7.6: This is to help visualize how a target with depth affects the spread of the interference peaks seen in Figure 7.5. If there is a target that has a depth of $R_{\max}$, then there will be peaks that show up symmetrically around the f_{12} peak. These peaks contain target information, so I need them to synthesize an image. If they were to spread any further, they would interact with the target information of the neighboring band, and I would lose the ability to demultiplex the spatial shears.

Calibration of Frequency Peaks

Now that I have defined the constraints on the target depth, and set up the system to keep the spatial shearing information separated in the temporal frequency domain I will now describe how to actually separate and demultiplex the spatial shears. First a calibration dataset is used to know exactly where to pull spatial shears apart from each other in the temporal frequency domain. To do this, I put a single mirror down range, in the center of the target that would reflect all four transmit beams directly back onto the camera sensor. This produced nearly delta function peaks at the center locations of the interference frequencies, see Figure 7.7. I then use a Matlab function *dftregistration*, which finds the locations of these peaks down to several decimal places of pixel bin size [19]. This is called sub-pixel registration. This Matlab function upsamples the data using Fourier transforms, then convolves the data with a sinc function representing the point spread function of the original signal, to find the correlation peak. If sub-pixel registration was not used one would

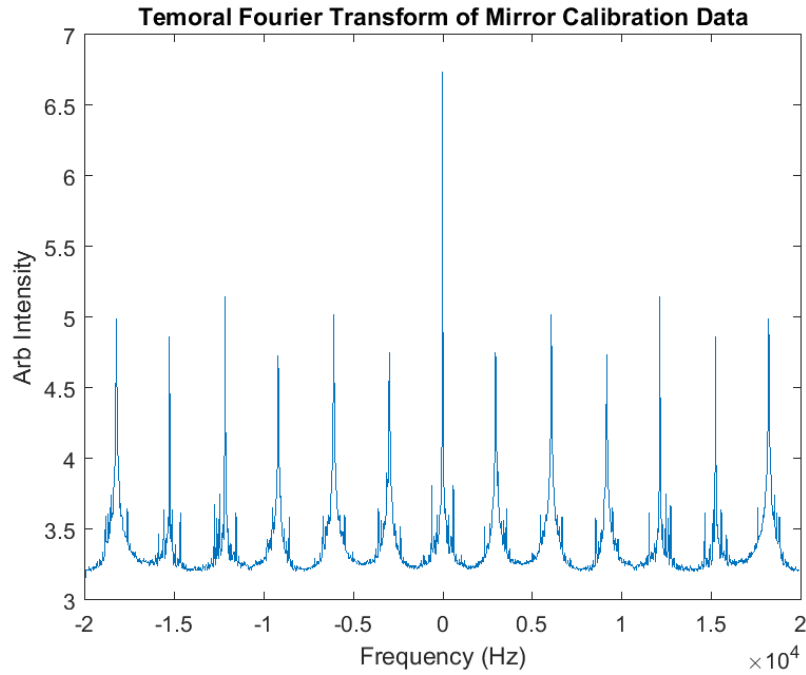


Figure 7.7: This is the Fourier transform along the time dimension of a target comprised of a single mirror. This gives the locations in the temporal frequency domain for where to crop out the different shear measurements.

get an imperfect shifting of the spatial frequencies, as illustrated in Figure 7.8.

With the precise locations of the peaks from the calibration data, I can use those to demultiplex the shears for real imagery. Since I know that the peaks on the left side of the DC peak are just complex conjugates of the peaks on the right, I only care about the locations of the six peaks on the right side of the DC peak. These are held in the “locs” variable in the Matlab code in Appendix C.

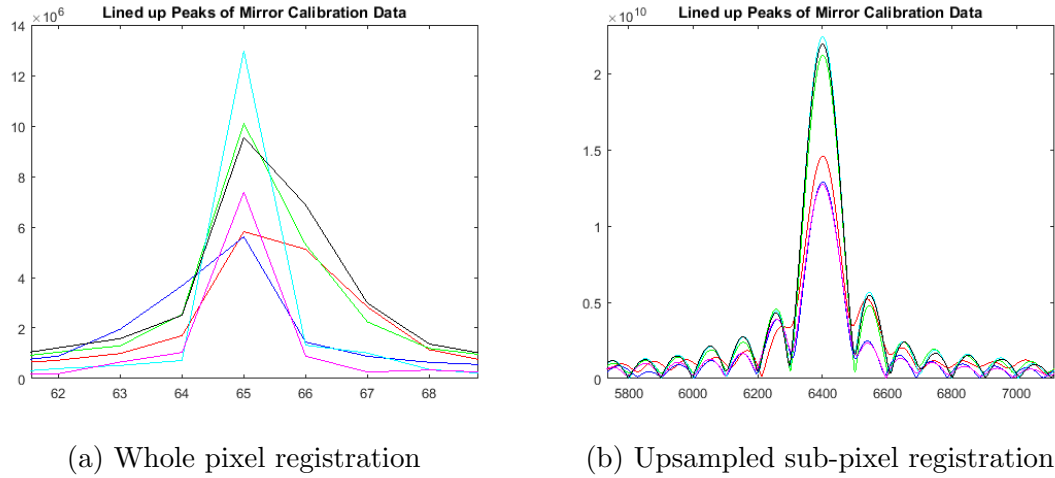


Figure 7.8: The left plot shows what would happen if one attempted to register the peaks as is without upsampling first. One can see that they are not truly aligned. The right plot is after upsampling, where it is clear that all the peaks are well aligned. It is from the right plot that the calibration peak locations are obtained.

Image Processing

Here I show how the experimental data was processed, starting with the raw camera data from a tilted Air Force Bar Target as the object. The raw data structure looks something along the lines of Figure 7.9. The first step in processing this raw data is using the locations of the peaks found in the mirror calibration data to isolate the spatial shears from each other. The next subsection will go over this process.

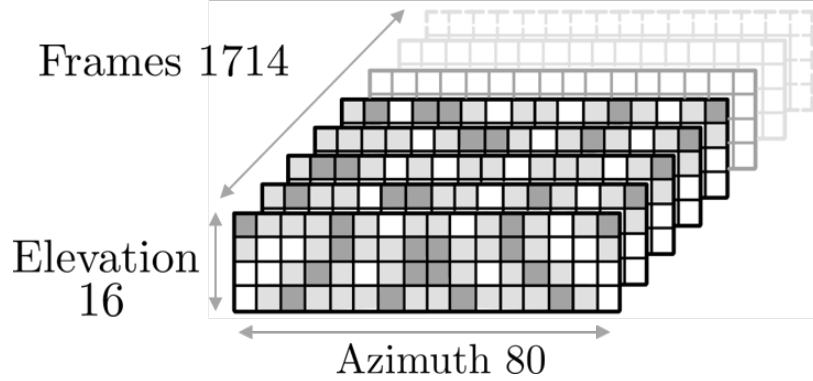


Figure 7.9: This is a visual representation of the raw data structure. 1714 images are taken over the stable duration of the chirp. Each frame is 16 pixels tall by 80 pixels wide. The pixel dimensions are the spatial dimensions, elevation and azimuth. The temporal dimension will be referred to as time or frames.

Using the Calibration Data to Demultiplex the Spatial Shears

The dataset has 1714 frames, or time samples. Taking a Fourier transform along the time dimension for each pixel returns a similar picture to Figure 7.5, except instead of delta functions, the peaks are spread out and noisier looking. This spreading is due to the depth of the target, and Figure 7.10 contains a plot showing what this data looks like. There are 13 “bands”, so the maximum number of samples in the temporal frequency domain that can be cropped down to without overlapping is,

$$\frac{1714}{13} = 131.8. \quad (7.7)$$

With this constraint I chose to crop down to 128 bins, because Fast Fourier Transforms like powers of two. [13] Figure 7.10 shows how to crop down to one band that corresponds to ∇_{12} , then take the inverse Fourier transform to bring it back to the original time domain. This new datacube is the isolated ∇_{12} , which only contains information about the interference between the returned speckle fields S_1 and S_2 . This is repeated for the remaining five shears, to demultiplex the spatial shears using

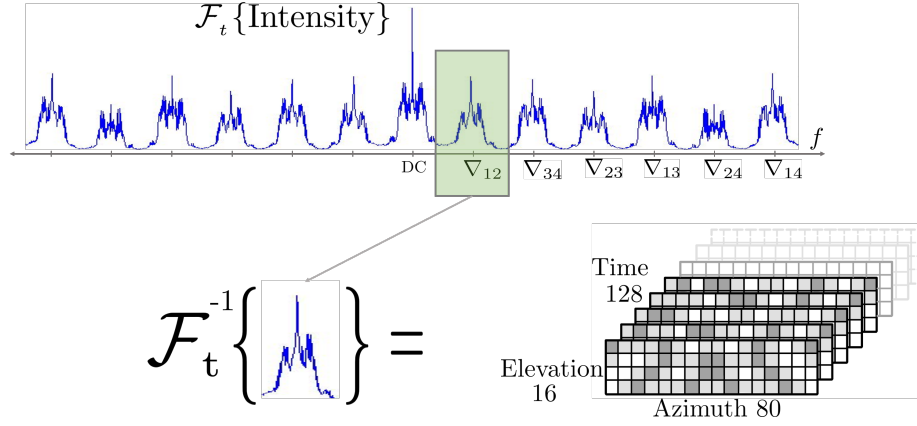


Figure 7.10: Here I show the process of isolating each shear. First Fourier transforming along the frames dimension to get the blue plot for each pixel. Then crop out each “band” using the locations that were determined by the mirror calibration data, where the width of the window is 128 samples wide. Then inverse Fourier transform this cropped band back to the original raw data time domain, but now it only contains information about that particular spatial shear. This process is repeated for the other five shears.

the frequency offsets built into the transmitters using time delays and the properties of chirped optical signals. [16]

Compressing the Dataset from 3D to 2D Shears

This subsection describes how these three dimensional measurements can be compressed into two dimensional, matrix representations, of the same shears, without losing information. The motivation behind doing this is that I already understand the processing algorithm in terms of two dimensional shear measurements from the simulation in the previous chapter. Another reason is that by taking a sparse data-cube and flattening it to a 2D matrix, I effectively increase the signal-to-noise ratio which should help the reconstruction algorithm in synthesizing an image.

In order to combine the elevation and time dimensions I use the fact that because the target is flat and tilted away from the detector there is a one-to-one

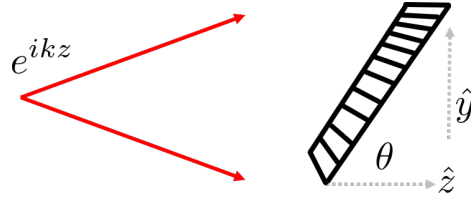


Figure 7.11: Simple geometry that explains how the elevation of the target is related to the range of the target for a flat target. In the far field, the scattered light from this target will be shifted spatially because it picked up a linear phase at the object, and the Fourier Shift Theorem will result in a shift in the far field.

correspondence between elevation and range dimensions. Using this relationship I can combine the many frames into one, higher resolution, higher SNR, image. [20]

Another way to visualize this is that when illuminating a tilted target with a chirped laser, the speckle pattern received at the detector plane shifts vertically as a function of optical frequency because of the equivalence of frequency and distance in the z phase part of the Fraunhofer propagation. Figure 7.11 shows how if one illuminates a target with a plane wave,

$$E = e^{ikz}, \quad (7.8)$$

and can express the z axis of the tilted target as,

$$z = y \tan(\theta), \quad (7.9)$$

which allows one to define the returned optical field as a function of the angle and elevation of the target,

$$E = e^{iky \tan(\theta)}. \quad (7.10)$$

One can now see that if this is Fourier transformed to get the field at the detector plane, the linear phase contribution leads to a spatial shift, thanks to the Fourier Shift Theorem. So as the chirp changes frequency, which changes k , one sees a shift in the speckle pattern at the detector plane, where each frame of the camera is sampling this speckle as it shifts. As the camera is sampling “strips” of the shear of this shifting

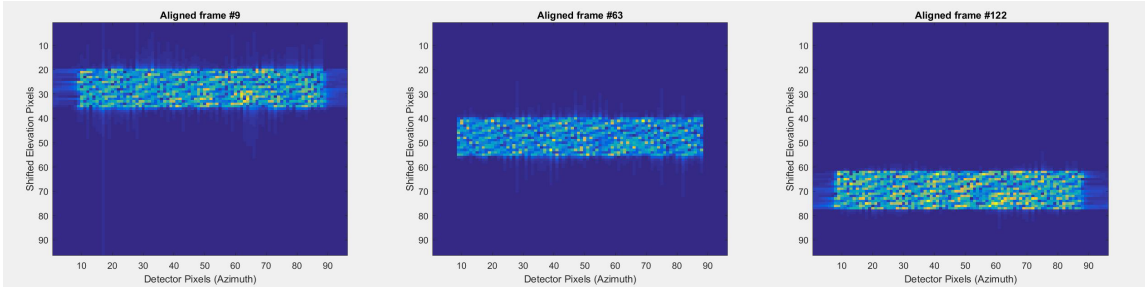


Figure 7.12: Here one can see three of the frames that have been shifted so that when they are summed along that dimension, one would get a two dimensional shear instead of 128 small versions of that same shear. Figure 7.13

speckle pattern, they shift up a little between each frame. So if the detector stays in one location, it samples different portions of the shear in the elevation dimension. Then registering the frames to each other using *dftregistration* [19], one can see that there is indeed a vertical shift of about half of a pixel between each frame. If all of the frames are registered to each other, they spread out in the elevation dimension, but not in the azimuth dimension. If I add all of the frames together with them in this new arrangement, the portions where frames overlap will add to the SNR of the shear, while also increasing the effective resolution in the elevation dimension because it is sampling more of the shear field in that dimension. This process can be seen in Figure 7.12, which shows three frames of the total 128 frames in their final registered locations prior to being summed along the frame dimension. In Figure 7.13, one can see the result of summing the registered frames. It now has more content along the

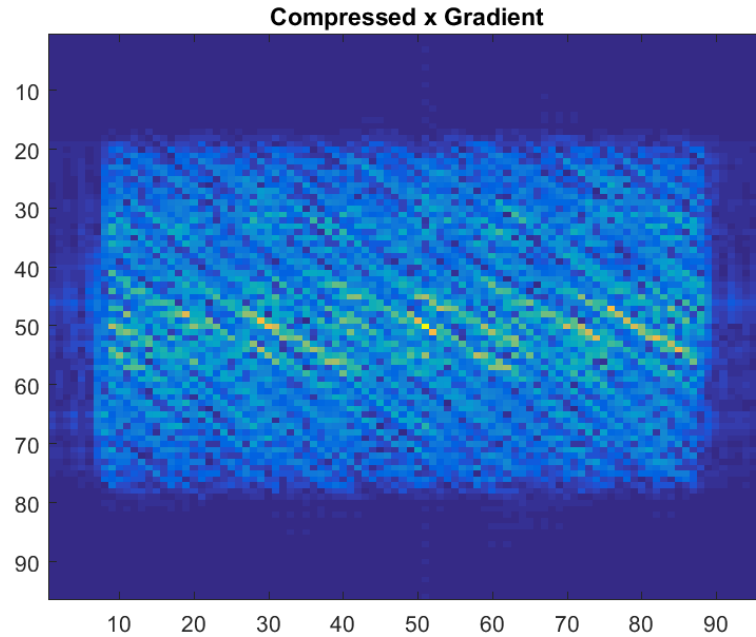


Figure 7.13: This is the result of registering each frame of the isolated ∇_{12} datacube, then summing along the frames dimension.

vertical direction than any individual frame had, which again is due to the fact that the speckle pattern shifts up throughout the duration of the chirp, and each frame is sampling a portion of the speckle as it shifts vertically.

First Try: Bar Target Data Processing

Now that I have taken the raw data, demultiplexed and compressed the shear measurements into a form that looks like the simulated shears, I can plug them into the reconstruction algorithm and see if an image comes out. The resulting image of the AFBT is shown in Figure 7.14. This is clearly not a reconstruction of an AFBT. I believe this is due to having too low of a signal to noise ratio (SNR). The next section will go over the experimental link budget using this bar target, and check the predicted SNR in the simulation to see if reconstructing an image is possible.

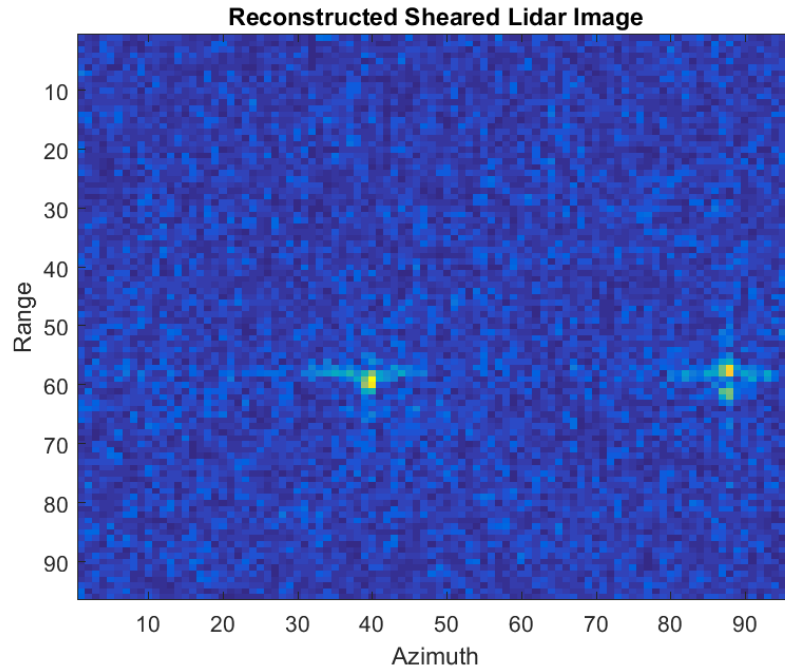


Figure 7.14: This is the result of the reconstruction algorithm if I put in the raw data taken from the experiment that has an AFBT as the object. Clearly this is not anything close to an image of an AFBT. There are a number of reasons that this data failed to reconstruct, but mostly it was due to low SNR.

Link Budget Calculation A link budget, also called a photon budget, is a calculation that seeks to predict how many photons are received by the detector compared to how many it were emitted. The purpose for this calculation is to compare the number of photons each pixel of the detector receives with the noise in the detector. This ratio is called the signal-to-noise ratio (SNR), and can be used to determine whether or not one would expect to be able to reconstruct an image. The measured output power of all four transmitters was 20mW. The signal here is actually a mixing term, and since there are four transmitters, that will lead to sixteen peaks in the Fourier domain, where the DC peak is really four peaks on top of each other, and the other frequency peaks are the individual shear mixing terms. So the optical power

allocated to one mixing term is sixteenth of the total transmitted power. [21, 22]

$$\text{Mixing Term Optical Power} = \frac{20 \text{ mW}}{16} = 1.25 \text{ mW} \quad (7.11)$$

The mode field diameter of the fiber tip transmitters is $10 \mu\text{m}$, which one can think of as a Gaussian beam waist diameter, and the far field illumination angle will be,

$$\theta_{\text{far field}} = \frac{\lambda}{\pi \omega_o}, \quad (7.12)$$

and the illumination beam area will be,

$$A_{\text{illum}} = \pi(R \tan(\theta_{\text{far field}}))^2. \quad (7.13)$$

Where $R = 3$ meters is the distance from the transmitter to the target. Since the target is made of retro reflective tape with a divergence half angle of about 2° , I can express the area of the reflected light in the detector plane as,

$$A_{\text{reflected}} = \pi(R \tan(2^\circ))^2 \quad (7.14)$$

The target is 20 cm square, but only about a quarter of it is retro fabric (the rectangles of the bar target). So the ratio of the area of the reflective part of the target to the illumination area is,

$$\frac{(20 \text{ cm})^2}{4 A_{\text{illum}}}. \quad (7.15)$$

The light is then reflected back towards the transmitters, and is concentrated into a circle defined by $A_{\text{reflected}}$, that includes the detector. The ratio of the area of a single

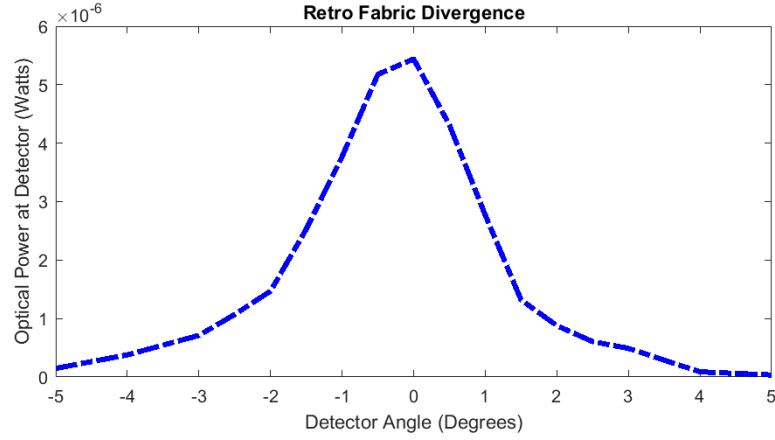


Figure 7.15: This plot shows how the intensity of the reflected light falls off as angle between transmitter and receiver changes. At zero degrees, one sees the greatest intensity. At about two degrees, almost all of the light is gone, which is why 2° is used as the threshold for the link budget calculations.

pixel to the area of that illumination spot is,

$$\frac{(20\mu\text{m})^2}{A_{\text{reflected}}}. \quad (7.16)$$

Most of the light falls within that circle because I tested the divergence of the reflected light from the retro-reflective fabric, and it can be seen in Figure 7.15, that most of the light falls within a 2° angular radius. The energy of a photon at 1550nm is,

$$E_\gamma = \frac{hc}{\lambda} = 1.282 \times 10^{-19}[\text{J}], \quad (7.17)$$

the detector has a quantum efficiency $\eta = 0.8$, and the integration time is $10\mu\text{s}$. Using all of this information one can calculate the number of photons expected to be

detected with a single pixel for one frame.

$$\text{Signal} = \frac{10 \text{ mJ}}{1 \text{ second}} \frac{(20 \text{ cm})^2}{4 A_{\text{illum}}} \frac{(20 \mu\text{m})^2}{A_{\text{reflected}}} \frac{10 \mu\text{s}}{1 \text{ frame}} \frac{1 \text{ photon}}{1.282 \times 10^{-19} [\text{J}]} \eta = 32.6 \left[\frac{\text{photons}}{\text{pixel} \text{ frame}} \right] \quad (7.18)$$

Now with the number of signal photons one can compare this number to the amount of noise electrons in the detector. For this particular camera, the specification sheet indicates that the read noise for the camera is 60 electrons per frame, and the dark current is,

$$\text{Dark Current} = 0.19 \times 10^6 \left[\frac{\text{electrons}}{\frac{\text{second}}{\text{pixel}}} \right] \quad (7.19)$$

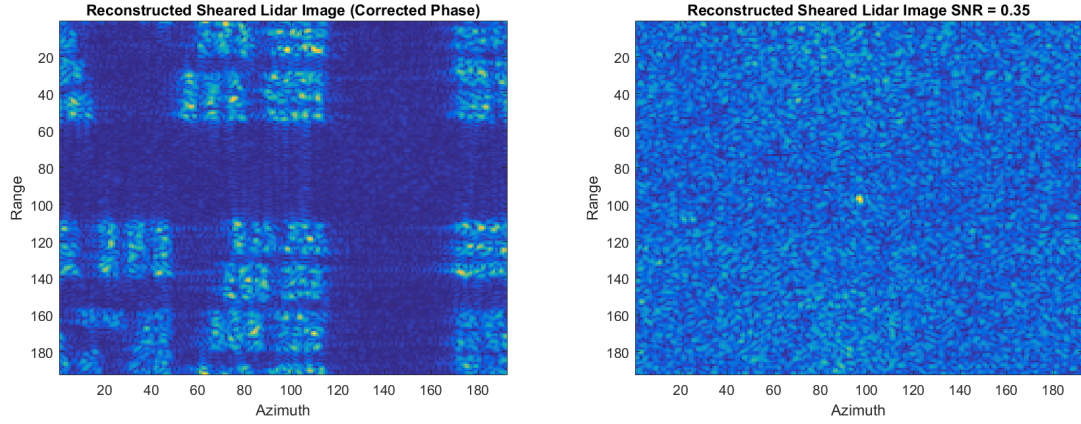
So to calculate the noise electrons per frame, one simply multiplies the dark current by the integration time and add the readout noise.

$$\text{Noise} = \text{Dark Current} \times 10 \mu\text{s} + 60 = 62 \left[\frac{\text{electrons}}{\frac{\text{pixel}}{\text{frame}}} \right] \quad (7.20)$$

With this measure of noise electrons and signal photons I can calculate the expected SNR for this particular experimental setup.

$$\text{SNR} = \frac{\text{Signal}}{\text{Noise}} \approx 0.53 \quad (7.21)$$

So an SNR of .53 is not very good at all. That means that the signal is about half as big as the noise, which means that the signal is indistinguishable from the noise. To see how this might affect the algorithm's ability to reconstruct an image with data that has an SNR of 0.53, I will quickly revisit the spatial shearing lidar simulation



(a) High SNR Simulated SHL Image

(b) SNR = 0.53 Simulated SHL Image

Figure 7.16: The left image is the phase corrected image from the simulation in the previous chapter with a very high SNR. The right image is the same phase corrected image, but reconstructed with the shears having an SNR = 0.53. This was done by adding noise in proportion to the size of the signal, to the signal. It is fairly clear that the reconstruction fails with low SNR, and the image even resembles the output of the real AFBT data from Figure 7.14.

from the previous chapter, but this adding noise so the shears have an SNR of 0.53.

Low SNR Spatial Shearing Reconstruction Simulation Since I was not able to reconstruct an image with the AFBT as the object, I need to justify this result with the simulation. I will use the simulation that I built in the previous chapter to test whether it can reconstruct an image from shears that have a low SNR. Figure 7.16 shows the image reconstruction for two cases. The left image is with a high SNR, and the right has an SNR of 0.53. It is evident that the reconstruction algorithm fails with such a low SNR. One might notice that the right image does look much like the attempted reconstruction with real data in Figure 7.14. The large peak in the center of both the experimental data and the low SNR simulation was a common artifact in failed reconstructions with low SNR.

The simulation has confirmed that the SNR was too low for the reconstruction

algorithm to work properly. One may wonder why I chose to do an experiment that has such a low SNR. When I originally set this experiment up I forgot to think about how the lack of a local oscillator would reduce the amount of effective signal by a very large margin. The “coherent gain” of having a LO makes up for very weak signals allowing for near single photon sensitivity. I made an attempt to increase the SNR of the data, by taking multiple datasets, registering the shears from each dataset, then coherently average over multiple datasets. This relies on registering the amplitude and the phase correctly, and I believe that there was enough variation in the optical chirp between datasets, that the phase registration was not good enough, and since the phase is crucial for reconstructing an image, I was essentially averaging away the signal.

Mirror Target Data Processing

The previous section, argued that the bar target data does not have enough SNR to reconstruct an image. I realized this during my last couple days while working at AFRL, so I quickly switched gears to a different target that would yield a larger SNR. That new target was four mirrors placed downrange in a special configuration, as depicted in Figure 7.17. Each mirror was individually aligned to reflect the transmit beam directly back onto the camera sensor. Better SNR allows for reconstructing from one shot of data, where there is less of a chance of washing out the signal by averaging the shear measurements over multiple different chirps. The reason this target has a much greater SNR than the AFBT, is because it reflects specularly instead of diffusely as with the retro tape. So in a sense I am reflecting four copies of the transmit beam directly back onto the camera sensor. With the AFBT, even though it was retroreflective, a lot of light was lost due to the partially diffuse nature of the target.

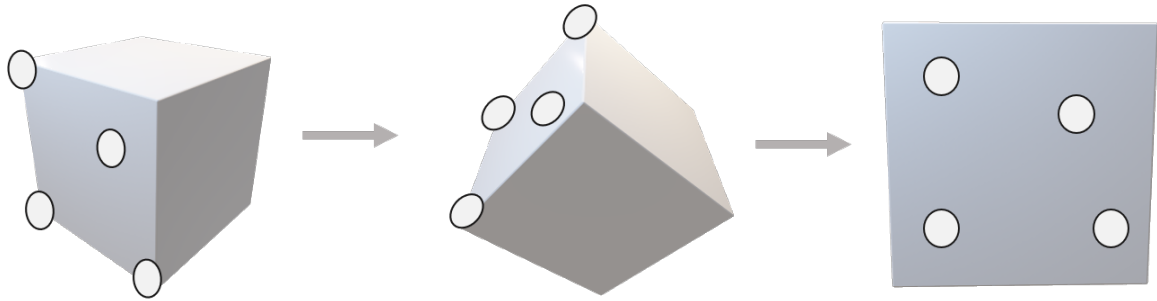


Figure 7.17: This is a visual representation of how the mirrors were arranged in the object plane. Imagine four mirrors all in the plane of one side of a cube. Three are on corners of the cube, and one is shifted into the middle by a little bit. Then this whole arrangement is tilted away from the detector by 45° relative to the y axis, the same angle as the bar target. The rightmost image represents what this would look like as a range-azimuth image. That is the pattern we are looking for in the reconstructed image.

Confidence That the Shears Contain Image Information An easy check to see if the collected data contained the information I expected it to was to look at the Fourier transform of the collected shears. To understand how this works one needs to remember that the shears are defined as the interference between the speckle of the target due to one transmitter, and the speckle of the target due to another transmitter that is shifted relative to the first transmitter. Another way to think about this is the scattered light from the target mixing with the scattered light from a shifted version of the target. This can be seen for shear field ∇_{13} on the leftmost image of Figure 7.18, where the gray dots correspond to the mirrors for the first transmitter, and the blue dots correspond to the same mirrors, but shifted diagonally, representing the target as seen by the second transmitter. Now if one thinks about each mirror as point sources, that become plane waves in the far field (at the detector) then one should see interference between each possible pair of mirrors. To start there will be a large DC peak because of the interferences between the gray and blue dots, because they are so

close to each other it will be a very low spatial frequency interference. Next there will be interferences between each group of mirrors, as shown by the arrows one through six. In the 2D Fourier transform of ∇_{13} , I would expect to see interference peaks at the locations that correspond to the arrows one through six, and this is seen in the middle image of Figure 7.18, where each interference has a conjugate peak denoted with a star. Finally, one can see the experimental result of a Fourier transform of ∇_{13} on the rightmost image of Figure 7.18. Here one can observe that the peaks in the correct locations. There is one exception however, which is the fifth interference term. It shows up in the same quadrant as the sixth interference term, which I believe is because the fifth term's spatial frequency is too high and thus gets aliased around into the wrong quadrants.

Being able to see the interference peaks in the shears was a nice sanity check, to

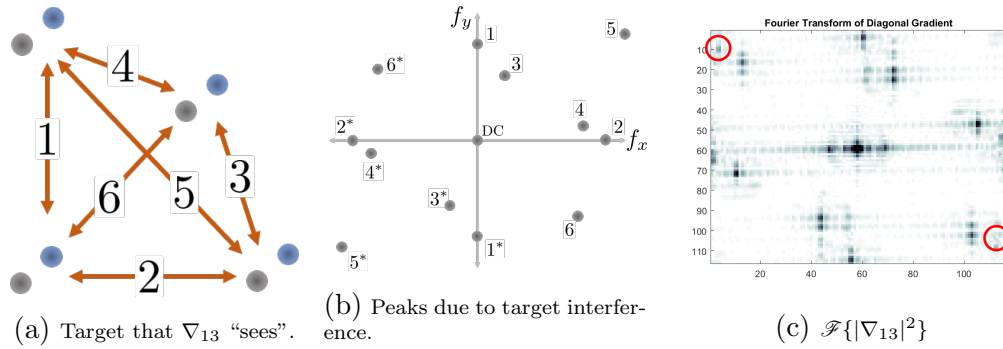
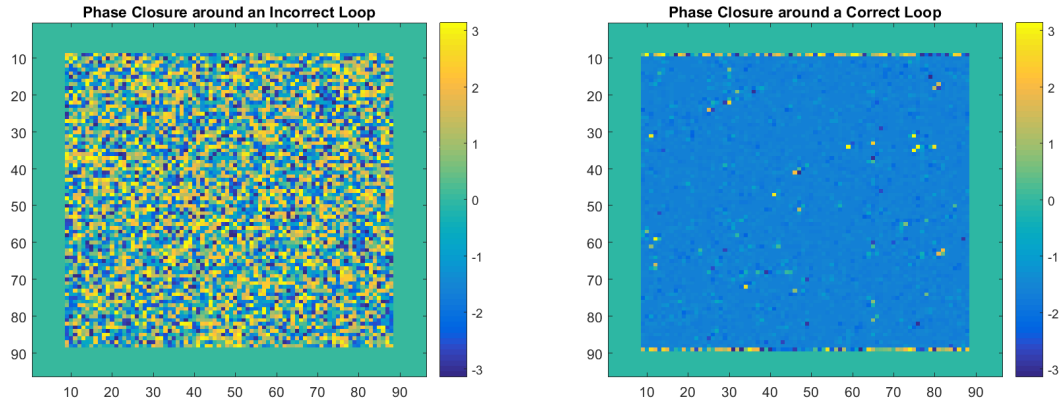


Figure 7.18: ∇_{13} is measuring the spatial frequency content of the interference between the target (four mirrors) and a shifted copy of the target, as illustrated in the leftmost image. This generates six large spatial interferences and four small (near DC) interferences. When I take the Fourier transform of the shear I expect to see peaks at the locations shown in the middle image, and they are quite evident in the real data (right image). There is one exception which is the fifth interference, which I expected to see in the bottom left and top right quadrants, but it showed up next to the sixth interference term in the real data, as indicated by the red circles. I believe that it has slightly too high of a spatial frequency, and thus aliased around to the other quadrants.

make sure that the data actually contained the image information that I expected it to contain. This was a key realization that gave me the motivation to continue pursuing the data processing of this experimental data. So with this in mind I continued to the next step in synthesizing images from this data, which is the phase closure correction algorithm.

Finding the Correct Phase Closure Loops As was mentioned in Chapter 5, the “direction” of the shears that are measured is unknown. So to get the right sign on the phase closure loops, I guess a direction, then look at the leftover phase for all the pixels to see if it was the correct guess. The guess was correct if the leftover phase is nearly the same for all the elements of the shear. This looks like a “flat” result in the leftover phase image. Because this is somewhat difficult to see with experimental data, I will first show what this looks like with incorrect and correct phase closure loops in the simulation data. Figure 7.19 demonstrates how when the incorrect phase closure is chosen the result is noise, but with the correct phase chosen most of the pixels have the same offset phase and look “flat”. [18]

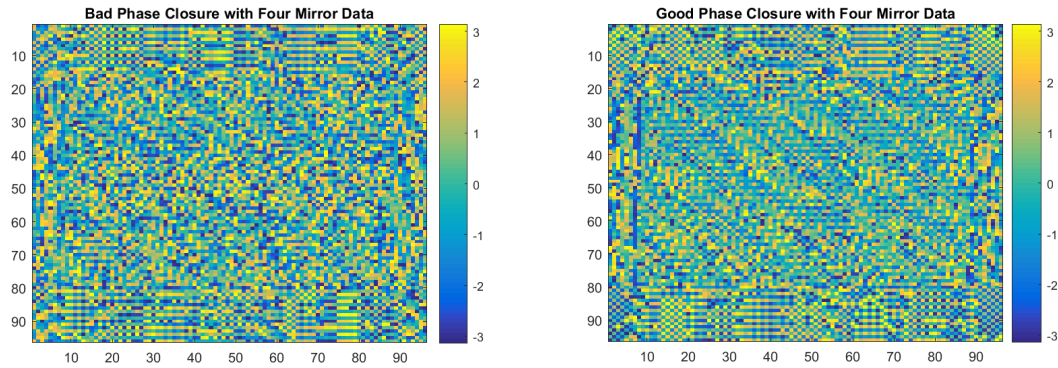
With the experimental data it is a little harder to tell which is “flatter”, but an example of a correct and incorrect phase closure loop are shown in Figure 7.20. On the left is an incorrect phase closure loop that results in what looks like noise. If the phase closure loop is incorrect, it will pick up on the target phase differences instead of the global phase offset that I am seeking to correct. The right image in Figure 7.20 shows the result of a correctly chosen phase loop. It is not perfectly flat, but it is more uniform than the left image. This is the basis for finding the correct sign of the phase closure loop. Essentially, I tried all possible combinations of signs in the phase closure loop equations, plotted the leftover phase for each, and look for ones that have a uniform offset. I then check the consistency of the signs of the shears for each



(a) Simulated incorrect phase closure loop chosen. (b) Simulated correct phase closure loop chosen.

Figure 7.19: This is an example of incorrect and correct phase closure from the simulated data. The left image is the closure phase of each pixel around a phase closure loop, where one of the shears directions was incorrectly chosen. This results in what looks like random noise across the image. The right image is the result of a correctly chosen phase closure loop, where there is an overall shift that means all of the pixels have roughly the same closure phase, which is an indication that the correct closure loop was used.

equation to make sure they make sense, which is to say that if an x shear is positive going right in one equation, it would be negative if it is going left in another phase closure equation. Once I have found the correct signs for the phase closure loops, I used the phase closure equations to calculate the phases that need to be removed from the shears and subtracted to get the phase corrected shears.



(a) Incorrect phase closure loop chosen.

(b) Correct phase closure loop chosen.

Figure 7.20: The left image is the closure phase of each pixel around a phase closure loop, where one of the shears directions was incorrectly chosen. This results in what looks like random noise across the image. The right image is the result of a correctly chosen phase closure loop, where there is an overall shift that means all of the pixels have roughly the same closure phase, which is an indication that the correct closure loop was used.

Zero Out DC Peaks in Shears This section is about removing an artifact from the reconstructed images of the data. I found that during the reconstruction, the algorithm can grab onto features that are much stronger than the actual image content that I want to reconstruct. One of these artifacts that I do not really care about is the DC content of the shears. This content essentially tells me about how I shifted the object (think back to the interference between a gray dot and the nearest blue dot in Figure 7.18). What I really would like to know is the spatial relationships between each mirror, not necessarily how big the mirror is. The reason I need to remove this DC peak is because it is about three orders of magnitude larger than the other interference peaks that I observe in the spatial frequency of the shears, which leads to artifacts in the reconstruction that just resemble a large DC peak.

So to fix this I simply take a spatial Fourier transform of each shear, and zero out the peak in the middle. This can be seen in Figure 7.21. The left image is before I

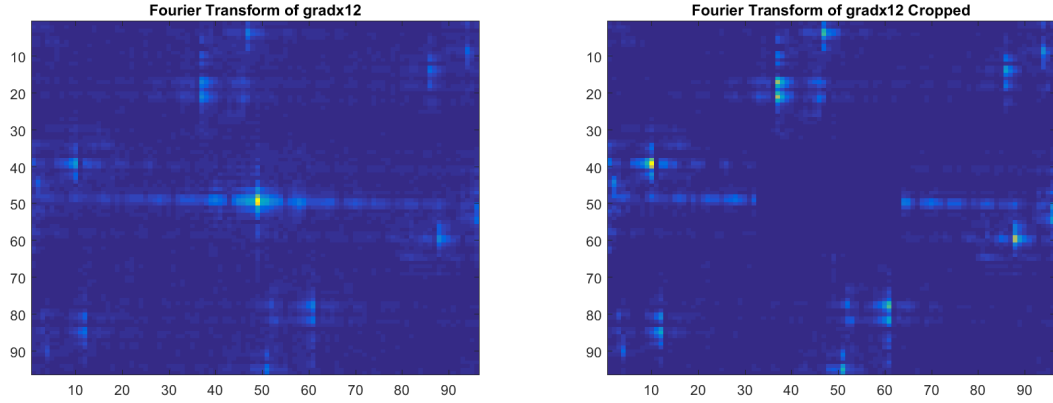
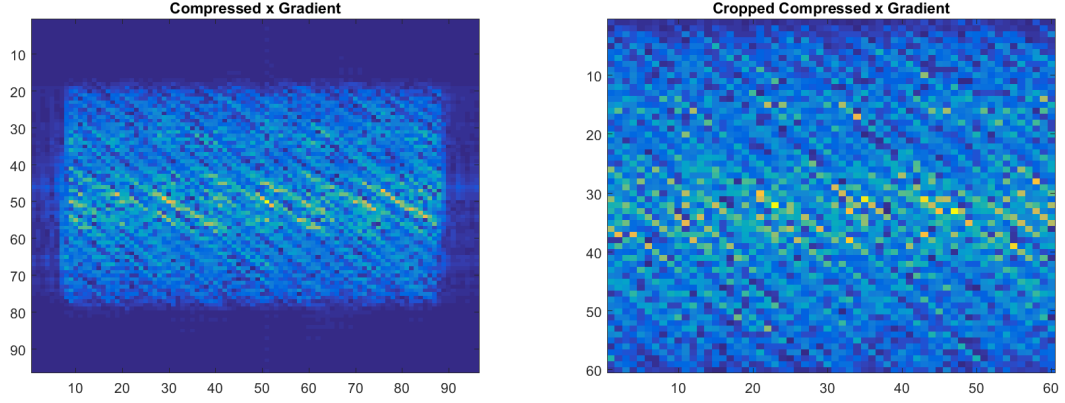
(a) $\mathcal{F}\{\nabla_{12}\}$ (b) $\mathcal{F}\{\nabla_{12}\} - \text{DC Peak}$

Figure 7.21: The left image is the spatial Fourier transform of one of the compressed shear measurements for the mirror target data. There is a DC peak, and multiple smaller peaks that contain the target information. In the right image I have cropped out the DC peak. To get back to the original domain of the shear, one simply takes an inverse Fourier transform.

zero out the peak, and the right is after I zero out the peak. Then to get the corrected shear back one simply takes an inverse Fourier transform.

Removing Padding Around Shears Thinking back to why I am using a covariance matrix to solve for the phase of the speckle of the target, one realizes that having zeros in the shears can create issues. The reason this can cause issues is because a covariance matrix looks at the variance between one element of one shear, and all the other elements of the rest of the shears. So if I have zeros in the shears, I am “telling” the covariance matrix that there is no relationship between one shear and another for some of the points. What I am getting at here is that the padding (zeros) around the shears used for most of the processing steps, see the left side of Figure 7.22, causes problems. So I got rid of this padding. This is easily done by cropping the shear down to just the middle portion where there is plenty of signal, which is shown in the right image of Figure 7.22.



(a) Compressed Shear

(b) Cropped Compressed Shear

Figure 7.22: The covariance matrix compares a point in one shear with every other point in all the other shears. If I leave the zero padding around the compressed shear, I am telling the system that there are a lot of zeros to compare things with, which does not really make sense. So here I take the compressed shear and crop it before I put it into the covariance matrix.

Finding the Correct Conjugations for the Shears Lastly I input these shears into the amplitude and phase solution algorithm. I did this so often that I built a Matlab function to do this, which is in Appendix D called *shear_recon*. This function takes four input shears and outputs the amplitude and phase solution for those shears. Remember that with real data, I do not know the direction of the shears, and while I have a bit more of an idea after the phase closure check I did earlier, I still wanted to cover all the bases and check each possible combination of conjugations of the shears input into the *shear_recon* function. One of the combinations, where say ∇_{12} and ∇_{14} were conjugated, would look like this,

$$[\text{amp}, \text{phase}] = \text{shear_recon}(\text{conj}(\nabla_{12}), \text{conj}(\nabla_{14}), \nabla_{13}, \nabla_{24}). \quad (7.22)$$

Where that is just one example of all sixteen possible combinations of conjugations of the shears.

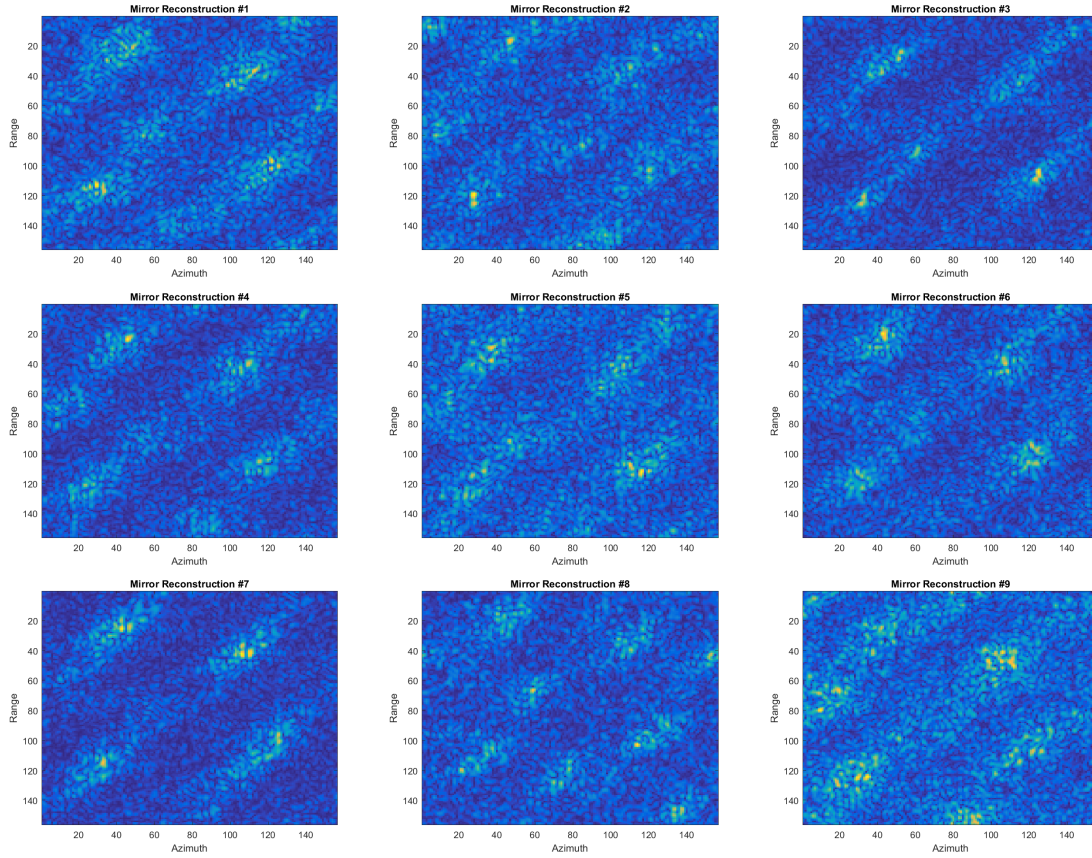


Figure 7.23: Here one can see the same four dot pattern showing up across all nine shots. This is after they have been registered and shifted so that the pattern shows up in the same place in each image. It is reassuring to see the same pattern in multiple data sets, which makes it less likely that it is an artifact.

How I found the current final image was by checking the output images from all sixteen combinations, and looking for something that resembled the target. After I found one that looked like I expected, I tested the same reconstruction settings for eight other datasets I had of the same target, and I saw the same pattern in each of the reconstructions, but shifted randomly relative to each other. These nine output images from nine different datasets can be seen in Figure 7.23. Where they have been registered to each other so that the image is in the same location for each output.

With nine images of the target I can sum them point-by-point to get an

“incoherently averaged” output image. This final output image is seen in Figure 7.24. This clearly has the correct shape of the target, as seen in Figure 7.17, and does not have any significant artifacts. So at face value, I have succeeded in reconstructing an image from the spatial shearing setup! This shows that spatial shearing lidar is a feasible imaging method. The next section will go into some detail about how I can reassure myself that this is indeed a reconstructed image, and not just some lucky artifacts that look like the expected image.

One thing that I did not mention earlier is that I have not attempted to use quadratic focusing or PGA to make the image sharper. As for the quadratic focusing, the reason this was not needed was that the light hitting the detector was nearly a

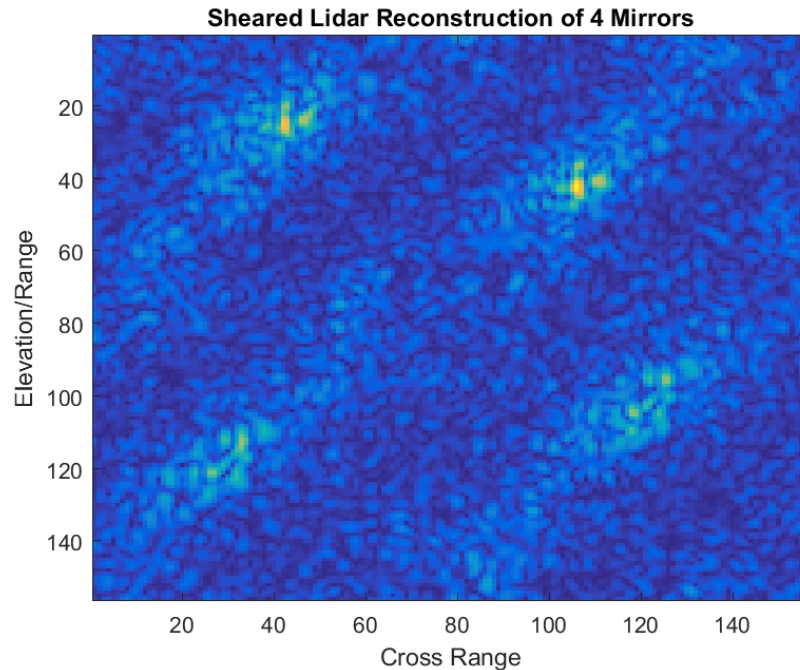


Figure 7.24: This is the final output image after incoherently averaging the reconstructed image from nine different shots. One can clearly see the shape of the mirror target, where three of the points are on the corners of a square, with one pulled in slightly.

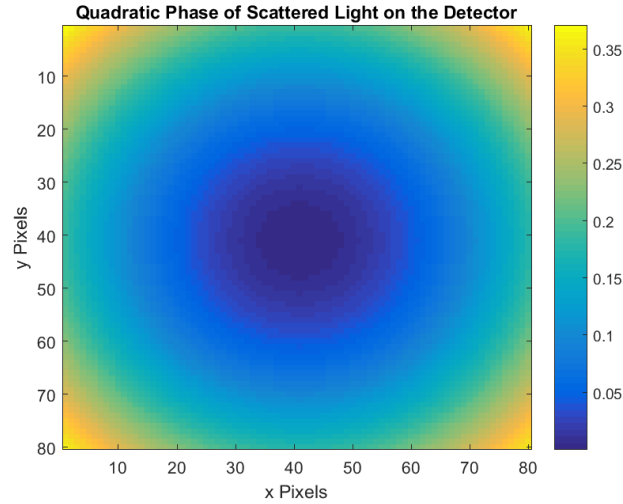


Figure 7.25: This is the quadratic phase of the light that scattered off of the target. I have not tried to fix this in the processing because the target is far enough away that the light is pretty close to a plane wave, so I do not have to worry about it.

plane wave. Figure 7.25 shows a simulation of the quadratic phase of the scattered light from the object assuming the scatterer is a point source, that falls onto an 80 by 80 pixel square of the detector.

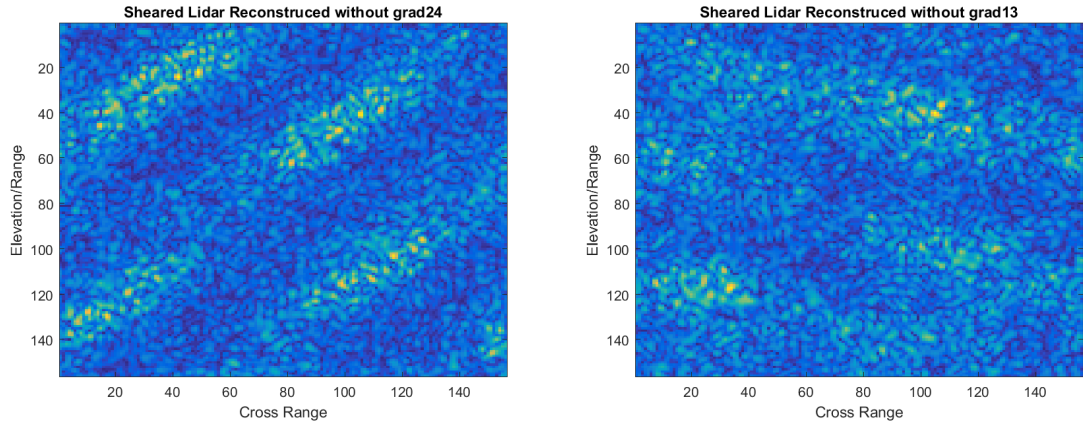
Validations of the Mirror Image

Test 1: Same Pattern over Multiple Datasets Figure 7.23 shows all nine reconstructions of the image, each from a different dataset. The fact that the same pattern shows up in each dataset means that it is unlikely to be just a random artifact from one dataset that happened to look like the pattern of the mirrors.

Test 2: Geometry of the Target, and Spatial Interference Peaks In the spatial Fourier transform of the shears of the mirror data, I saw twelve peaks, each corresponding to a real interference that one would expect to see based off of the geometry of the target. This was shown in Figure 7.18, and is good evidence that the data contained information about the target from the beginning. This is an important

check because it lets me know that the information is at least in the shears, and that all I had to do was correctly interpret that information in the reconstruction algorithm.

Test 3: Removing Shears from the Covariance Matrix One more check to see if this image is correct is by removing some constraints on the reconstruction. I can do this by leaving one of the shear measurements out of the covariance matrix, during the reconstruction. If this is done for a diagonal shear, say ∇_{24} , I would expect to see the reconstruction become worse, but only in the dimension that was defined by that shear, which was a diagonal from bottom right to top left. Figure



(a) Removed ∇_{24} from covariance matrix (b) Removed ∇_{13} from covariance matrix

Figure 7.26: Removing one of the diagonal shears from the covariance matrix removes the constraint along that dimension from the reconstruction. If ∇_{24} is removed the left image is smeared out in the direction that is constrained by that shear. It looks backwards because of the way Matlab indexes things. This is also seen in the right hand reconstruction where ∇_{13} was left out of the covariance matrix.

7.26 demonstrates this for situations where I have removed a diagonal from the reconstruction in each case. The reason it appears to be smearing out is because I have removed the information about that dimension. This is a good check, because if it did not behave like this, I would be inclined to say that the spots are artifacts, that are not related to the image content. This is similar to a situation where one

might have higher resolution in one dimension than the other, where it appears to be smeared out in the low resolution dimension. [21]

In the end I have reconstructed an image of four mirrors, and validated that the image is what I think it is. I have demonstrated the utility of the phase closure algorithm in removing the random piston phases that are on each shear. In all I have shown that it is possible to reconstruct an image of an object with multiple simultaneous shear measurements of its spatial frequency content using the covariance matrix approach.

CONCLUSION

Is Spatial Shearing Lidar Practical?

This section will discuss different situations or scenarios that this imaging technique could be applied to, and how the experimental parameters will need to be changed for each situation.

Retroreflective AFBT at 3 meters

To start I will readdress the initial experiment described in the previous chapter, where the target is a tilted retroreflective AFBT located about 3 meters from the sensor. The simulation of spatial shearing lidar demonstrated that an SNR of about 2 was the minimum required to reconstruct an image. That will be the baseline for the link budget. Using the model in Chapter 7, I can evaluate what changes need to be made in order to successfully reconstruct an image of the retroreflective AFBT.

$$\text{SNR} = \frac{\left(\text{optical power}\right)\left(\frac{\text{target area}}{4 A_{\text{illum}}}\right)\left(\frac{\text{pixel area}}{A_{\text{reflected}}}\right)\left(\frac{\text{integration time}}{1 \text{ frame}}\right)\left(\frac{1 \text{ photon}}{1.282 \times 10^{-19} [\text{J}]}\right)(\eta)}{\text{Noise}} \geq 2 \quad (8.1)$$

One can see that there are many different parameters that can be tweaked to achieve an SNR greater than 2. In Chapter 7, the link budget predicted an SNR of 0.5 for the experimental parameters. The first, and easiest way to increase this SNR to 2 would be to increase the optical power. In the actual experiment the total transmit power was 20 mW, so to bring the SNR up, one need only increase the optical power by a factor of four. 80 mW is not an unreasonable amount of power, and so this would be one straightforward way that I could have changed the experimental setup to allow for imaging the retroreflective AFBT at a distance of 3 meters.

Diffuse AFBT at 3 meters

Now I will discuss the exact same experimental setup as used in the previous case, but with a diffusely scattering AFBT instead of a retroreflective one. This should change the SNR equation slightly, where one needs to account for light scattering in all directions instead of most of it coming straight back to the camera sensor.

$$\text{SNR} = \frac{\left(\text{optical power}\right)\left(\frac{\text{target area}}{4 A_{\text{illum}}}\right)\left(\frac{\text{pixel area}}{\pi R^2}\right)\left(\frac{\text{integration time}}{1 \text{ frame}}\right)\left(\frac{1 \text{ photon}}{1.282 \times 10^{-19} [\text{J}]}\right)(\eta)}{\text{Noise}} \geq 2 \quad (8.2)$$

The biggest change is that $\frac{1}{A_{\text{reflected}}}$ is replaced with a $\frac{1}{\pi R^2}$ term, which treats the target as a Lambertian reflector. This change leads to a required optical power at the transmitter of about 60 Watts. This is a lot of power for such a short range application. In order to bring the transmitter power down to something more reasonable, like 1 Watt, a couple experimental parameters need to be modified.

The first change could be to make the target area and illumination beam area both smaller by a factor of four. This will lead to a more concentrated illumination beam, and lead to each point in the target reflecting more light.

The second change will be to increase the effective pixel area on the detector by a factor of ten, i.e. $20 \mu\text{m}$ pixels become $63 \mu\text{m}$ pixels. Changing the pixel size decreases the spatial frequency bandwidth that can be detected, but since the target size was decreased, the new pixel size is still well within the required Nyquist sampling frequency. A way this could be accomplished is by using a set of lenses with a larger overall aperture, where the scattered light from the target will be demagnified onto the camera sensor.

The last change would be to use a camera with lower noise. The camera used in this experiment had about 60 electrons of noise. If that could be decreased to about 20 electrons of noise per frame, which is possible for some state-of-the-art InGaAs

cameras, then a factor of three in power could be saved.

With all these changes, the factor of four in target area, ten in pixel area, and three in camera noise, the transmitter power would need to be about a Watt of power. Which is a more reasonable amount for a short range diffuse target.

Diffuse AFBT at 1 km

A last scenario would be long range imaging at say 1 km. Like the previous target, the most notable change that can be done to make this imaging system work at ranges up to 1 km would be to narrow the field of view, by increasing the aperture on the receiving optics. Assuming a Gaussian beam propagation of the transmit beams, where the waist is at the transmitter, one would be able to image an object that is the same size as the previous diffuse target if the transmitting beam waist diameter was $1/3$ cm. The other necessary modification would be to increase the pixel area by a factor of 10,000, which would lead to pixels that are 2 mm, and a total receiving aperture size of 16 cm, assuming that the array is 80 by 80 pixels. These changes along with the assumption that the noise level is still at 20 electrons per frame lead to an optical power requirement of about 233 Watts. That is still a lot of power, but some long range imaging systems have been in that range before.

So while these calculations show that it might be possible, how useful would it be? To start, the object being imaged is very small, about 3 cm on a side. Also, the whole point of self-referencing lidar is to attempt to remove atmospheric turbulence from the scattered light coming back from the object. The self-referencing can only remove the effects of turbulence that has length scales much larger than the shear size. Pixels that are 2 mm, are approaching the inner length scales of atmospheric turbulent cells. Since the techniques presented in this thesis rely on the shears being the size of a pixel, that means that as the size of the pixel gets larger so does the

length scale of the turbulence at which this technique is not beneficial.

The integration time of the camera cannot be made longer, because the frame rate of the camera must be high enough to capture the temporal mixing frequencies of the different shears. This limits the overall “frame rate” of spatial shearing lidar in this mode to about 20 Hz, which is fairly slow as far as imaging systems go.

One last thing that could help increase the field of view or decrease the power requirements would be to use a single photon counting camera. The noise would fall to about 1 electron per frame, which is really the last hope for making this a practical long range imaging system. So overall, while I do believe it is possible to reconstruct images of a diffuse target at short range, this method may not be ready for long range applications which would require very expensive cameras, large receiving optics, and large optical powers. In the future work section I discuss an idea for increasing the SNR by bringing back the coherent gain of a local oscillator, which is another way to effectively decrease the noise of the sensor. This I believe this could be the best solution to using this imaging technique at long range.

Successful Experiments

Temporal Heterodyne Holography

In Chapter 3, I explored how to take the ideas of spatial offset digital holography, and transfer them to a temporal heterodyne method that utilizes frequency shifts instead of spatial shifts to create temporal intensity modulations instead of spatial intensity modulations. These can be interpreted in nearly the same way that the spatial case can, but has the added advantage of retaining the full spatial resolution of the camera. This is a type of holography, that has only recently become possible with the advent of photonic mixing devices that detect the phase of amplitude modulated light. Perhaps as this technology develops, they may find a way to detect multiple

frequencies in each pixel, allowing for multiple interference frequencies that could then be used to get a coarse range dimension in addition to the spatial dimensions.

Temporal Frequency Demultiplexing of Spatial Shears

At the beginning of the Chapter 7 I demonstrated a new demultiplexing method that relies on the temporal frequency offsets generated by interfering transmitter beams with varying time delayed chirped signals. This created non-redundant interference frequencies that can be separated from each other in the temporal frequency domain. This is a relatively new method because detector arrays with enough temporal bandwidth in the near IR are only recently becoming available.

This demultiplexing method is unique because all of the demultiplexing happens in the post processing, which keeps the experiment much simpler to setup, and also allows for simultaneous measurements of multiple experimental parameters that may not be consistent if taken sequentially.

Spatial Searing Lidar Verification

In the second half of Chapter 7, I showed that spatial shearing lidar is a viable coherent imaging technique. I have yet to test its abilities to remove atmospheric phase aberrations, but for these first steps it is important to see if it was even possible to create images with this technique.

Some of the drawbacks to spatial shearing lidar are the extremely sensitive processing steps required to synthesize and image, and the SNR limitations. I explored the sensitivity of the processing in Chapter 6 where I simulated the experiment, and I would not have been able to reconstruct an image in the Chapter 7 without the knowledge I gained from the simulation. The other main problem is the lack of coherent gain on the detector. A common way to amplify the signal in a coherent imaging system is to use a powerful local oscillator, as the interference term between

the local oscillator and the scattered light is proportional to the product of their amplitudes. In the experiment, I did not have an LO, and so the signal was very weak. This is clearly a bad sign if this system is to be used in long range applications, if it has trouble with an object only 3 meters away. In the next section, I discuss a new experiment that might be able to circumvent this limitation.

Future Work

Temporal Shearing Self-Referencing Lidar

In Chapter 5, I introduced self-referencing lidar in terms of spatial shearing methods. Originally I was going to do an experiment that utilized a single pixel detector, and time delays to do “frequency shearing” which would measure the phase derivatives of the range spatial frequencies, and reconstruct a range profile from that. [6] The reason I went with spatial shearing was because of experimental bandwidth constraints that were not fully thought through at the time, namely the size of the shears and the size of the target are inversely related. So I switched to spatial shearing where the bandwidth is not tied to the size of the target. The next useful experiment would be to validate frequency shearing lidar as a method for measuring the range profile of a target. This would require much larger targets, faster chirp rates, and longer fiber delays (or frequency shifts), and so might be better suited for an experiment that is all in fiber optics. This is being worked on by myself and an undergraduate student at MSU Spectrum Lab.

Combining Shearing Methods

After frequency shearing lidar has been validated, the next logical step would be to attempt to combine the two shearing methods. This would allow for a fully three dimensional measurement of the target, that can mitigate atmospheric turbulence

effects. One trick that might be difficult to overcome would be the processing burden. A three dimensional measurement would be similar to the spatial shearing processing, but each range bin would require the same amount of processing as a single image from a spatial shearing lidar setup.

Fresnel Turbulence Simulation

One other current area of research is to create a more accurate Fresnel-based simulation of spatial shearing lidar. This would allow for more flexibility, while giving more realistic simulation results. [13] We could also place turbulent phase screens into the propagation of the light to see how it effects the reconstruction of the image, and compare that to conventional lidar systems in the same turbulence.

Computing Digital Shears for Self-Referencing Lidar

One possible solution to the SNR problem I faced with self-referencing lidar, is as follows. What I would do is have the same experimental setup as for spatial shearing lidar, but instead of having all four transmitters illuminating the target, I would take one of them and illuminate the sensor with it, which will act as the local oscillator. What happens then is that in the temporal frequency domain of the raw data I would have two different types of coherent measurements. Three of the peaks would be the interference between the LO and the scattered light from the target. The other three peaks would be interferences between the speckle patterns returning from the target. Essentially it would be three standard lidar images, and three speckle shear measurements.

The next step would be to synthesize the shears digitally in the post processing using the three standard lidar measurements. We could then compare them to the actual shear measurements to estimate the turbulence effects. Then I would use these digitally generated speckle shears as the inputs to the processing algorithm outlined

in the simulation chapter to get the speckle of the object in amplitude and phase, but without the atmospheric turbulence. This would bring back the coherent gain, while retaining the atmospheric turbulence rejection effects of self-referencing lidar.

Future work on self-referencing lidar will continue both at the Air Force Research Laboratory Laser Radar Branch, as well as at Montana State University Spectrum Lab.

REFERENCES CITED

- [1] J. W. Goodman, *Introduction to Fourier Optics*. Roberts and Company Publishers, 2005, google-Books-ID: ow5xs_Rtt9AC.
- [2] A. E. Tippie, *Aberration correction in digital holography*. University of Rochester, 2012.
- [3] E. Hecht, *Optics*, 5th ed. Boston: Pearson Education, Inc, 2017.
- [4] R. A. Hutchin, “Sheared coherent interferometric photography: a technique for lensless imaging,” in *SPIE’s 1993 International Symposium on Optics, Imaging, and Instrumentation*. International Society for Optics and Photonics, 1993, pp. 161–168. [Online]. Available: <http://proceedings.spiedigitallibrary.org/proceeding.aspx?articleid=937625>
- [5] D. G. Voelz, J. D. Gonglewski, and P. S. Idell, “SCIP computer simulation and laboratory verification,” in *SPIE’s 1993 International Symposium on Optics, Imaging, and Instrumentation*. International Society for Optics and Photonics, 1993, pp. 169–176. [Online]. Available: <http://proceedings.spiedigitallibrary.org/proceeding.aspx?articleid=937627>
- [6] D. J. Rabb and J. W. Stafford, “Spectral Shearing Ladar Application #15782327;,” Patent, aFRL RYMM.
- [7] N. Pavillon, C. S. Seelamantula, J. Khn, M. Unser, and C. Depeursinge, “Suppression of the zero-order term in off-axis digital holography through nonlinear filtering,” *Applied optics*, vol. 48, no. 34, pp. H186–H195, 2009. [Online]. Available: <https://www.osapublishing.org/abstract.cfm?uri=ao-48-34-H186>
- [8] M. Liebling, T. Blu, and M. Unser, “Complex-wave retrieval from a single off-axis hologram,” *JOSA A*, vol. 21, no. 3, pp. 367–377, 2004. [Online]. Available: <https://www.osapublishing.org/abstract.cfm?uri=josaa-21-3-367>
- [9] H. Kraft, J. Frey, T. Moeller, M. Albrecht, M. Grothof, B. Schink, H. Hess, and B. Buxbaum, “3d-camera of high 3d-frame rate, depth-resolution and background light elimination based on improved PMD (photonic mixer device)-technologies,” *OPTO, Nuernberg, May*, 2004. [Online]. Available: http://www2.uni-siegen.de/~reg-st2/3D-View/doc/Opto04_4.2.pdf
- [10] Z. W. Barber and S. Crouch, “Table-top explorations of synthetic aperture ladar imaging,” in *Lasers and Electro-Optics (CLEO), 2013 Conference on*. IEEE, 2013, pp. 1–2.
- [11] A. D. Payne, A. P. Jongenelen, A. A. Dorrington, M. J. Cree, and D. A. Carnegie, “Multiple frequency range imaging to remove measurement ambiguity,” in *Optical 3-D Measurement Techniques*, 2009. [Online]. Available: <https://researchcommons.waikato.ac.nz/handle/10289/4032>

- [12] P. A. Roos, R. R. Reibel, T. Berg, B. Kaylor, Z. W. Barber, and W. R. Babbitt, "Ultrabroadband optical chirp linearization for precision metrology applications," *Optics Letters*, vol. 34, no. 23, pp. 3692–3694, Dec. 2009. [Online]. Available: <https://www.osapublishing.org/abstract.cfm?uri=ol-34-23-3692>
- [13] J. D. Schmidt, *Numerical simulation of optical wave propagation with examples in MATLAB*. Bellingham, Wash: SPIE, 2010, oCLC: ocn606405811.
- [14] J. W. Stafford, B. D. Duncan, and D. J. Rabb, "Phase gradient algorithm method for three-dimensional holographic ladar imaging," *Applied Optics*, vol. 55, no. 17, p. 4611, Jun. 2016. [Online]. Available: <https://www.osapublishing.org/abstract.cfm?URI=ao-55-17-4611>
- [15] B. Krause, *Motion compensated multi-wavelength digital holography*. Google Patents, Feb. 2017. [Online]. Available: <https://www.google.com/patents/US9581967>
- [16] B. Spivey, D. Sandler, P. A. Johnson, P. Fairchild, and L. Cuellar, "High Resolution 3-D Holographic Camera," Patent.
- [17] C. V. Jakowatz and D. E. Wahl, "Eigenvector method for maximum-likelihood estimation of phase errors in synthetic-aperture-radar imagery," *JOSA A*, vol. 10, no. 12, pp. 2539–2546, 1993. [Online]. Available: <https://www.osapublishing.org/abstract.cfm?uri=josaa-10-12-2539>
- [18] T. J. Cornwell, "The applications of closure phase to astronomical imaging," *Science*, vol. 245, no. 4915, pp. 263–269, 1989.
- [19] M. Guizar-Sicairos, S. T. Thurman, and J. R. Fienup, "Efficient subpixel image registration algorithms," *Optics letters*, vol. 33, no. 2, pp. 156–158, 2008.
- [20] Z. W. Barber, J. R. Dahl, A. B. Mateo, S. Crouch, and R. Reibel, "High resolution FMCW ladar for imaging and metrology," in *Applications of Lasers for Sensing and Free Space Communications*. Optical Society of America, 2015, pp. LM4F–2.
- [21] A. V. Jelalian, *Laser radar systems*. Boston: Artech House, 1992.
- [22] S. Crouch, "Synthetic Aperture Ladar Techniques," Ph.D. dissertation, Montana State University, Jul. 2012.

APPENDICES

APPENDIX A

LFM LIDAR PROCESSING SCRIPT

```

1 % AFRL Experiment 1 data processing
2 % single temporal shear, with LO bouncing off of mirror in scene.
3 % Ryan Galloway
4
5 % 7.18/17
6
7 clear all
8 close all
9
10 %loop to import and process the data:
11 for k = 1:20
12
13     %% Import Saved Data from Xenics camera using my modified function:
14
15     % Define the path:
16     filename = ['E:\Drive\School\AFRL\AFRL summer\Research Stuff\data\7.25.2017\SAL2-20-800-3-',
17               num2str(k), '.bin'];
18
19     % Read .bin file, retrieving the data as a row x column x frames data cube
20     % and the metadata as a struct file
21     [timeData, metadata1] = readXenicsData(filename);
22
23     %% Perform LO/Background Subtraction:
24     % Import Dark snap
25
26     % Define the path:
27     filename = ['E:\Drive\School\AFRL\AFRL summer\Research Stuff\data\7.25.2017\dark2-20-800-3-',
28               num2str(k), '.bin'];
29
30     % Read .bin file, retrieving the data as a row x column x frames data cube
31     % and the metadata as a struct file
32     [darkData, metadata2] = readXenicsData(filename);
33
34     % Subtract each corresponding dark frame from the corresponding raw data
35     timeData = timeData - darkData;
36
37     %% Check the data:
38
39     % Define the number of frames based off of the metadata
40     frames = metadata1.frames;
41     rows = metadata1.rows;
42     cols = metadata1.cols;
43
44     %% Processing
45
46     % Cropping out the first 30 frames (laser not stablized until 950us after trigger)
47     timeData = timeData(:, :, 32:1280);
48
49     % Padding 2nd and 3rd dimensions:
50     timeData = padarray(timeData, [0, 40, 626]);
51
52     % Fourier Transform along the third dimension for each pixel (time domain
53     % to range domain)
54     rangeData = fftshift(fft(fftshift(timeData, 3), [], 3), 3);
55
56     % Crop the range data down to the spectrum of interest
57     rangeData_crop = rangeData(:, :, 1575:1734);
58
59     % Cross range compression: (Fourier transform along the 80 pixel dimension)
60     cross_and_range_Data = fftshift(fft(fftshift(rangeData_crop, 2), [], 2), 2);
61
62     %% Record the Data from each loop to be processed outside the loop
63
64     % Capture every subframe from every chirp
65     hypercube(:, :, :, k) = cross_and_range_Data;
66
67     % Capture only the summed subframes (one image per chirp)
68     cube(:, :, k) = rot90(squeeze(sum(abs(cross_and_range_Data).^1.2, 1)), -1);
69
70 end
71
72 %% Register each Chirp with respect to the first chirp (This is Necessary)
73
74 % Take the first image (sum of all subframes from the first chirp) as a
75 % reference:
76 ref = cube(:, :, 1);
77
78 % dftregistration takes 3 inputs, the first is what you reference the
79 % second with, and the third entry is how many sub pixels, so it will
80 % register shifts in 10ths of a pixel
81 for kk = 2:k
82     [reg, Greg] = dftregistration(fft2(ref), fft2(cube(:, :, kk)), 10);
83     ref = ref + abs(ifft2(Greg));
84     % figure(11)
85     % imagesc(ref)
86     pause(.3)
87     %reg

```

```

86 end
87
88 SAL_Output = ref;
89 %% Output final images:
90
91 % Power to raise the images to
92 x = .7;
93
94 figure(12)
95 imagesc(double(((squeeze(abs(SAL_Output).^x))))
96 xlabel('Cross Range')
97 ylabel('Range')
98 title('Registered Output SAL Image across all chirps')
99 caxis([1.6e4,1.4e5])
100
101 % upsampled output image
102 figure(13)
103 imagesc(double(((squeeze(abs(ft2(padarray(ift2(ref),[80,80]))).^x))))
104 xlabel('Cross Range')
105 ylabel('Range')
106 title('Upsampled Registered Output SAL Image across all chirps')
107 caxis([1.6e4,1.4e5])
108
109 % Cropped to horizontal bars on bottom right
110 figure(14)
111 imagesc(double(((squeeze(abs(ft2(padarray(ift2(SAL_Output(80:110,120:145)),[300,300]))).^x))))
112 xlabel('Cross Range')
113 ylabel('Range')
114 title('Cropped Upsampled Horizontal Bars')
115 caxis([1.8e4,1.4e5])
116
117 % Cropped to horizontal bars on bottom right
118 figure(15)
119 plot(sum(((squeeze(abs(ft2(padarray(ift2(SAL_Output(80:110,120:145)),[300,300]))).^x))),2))
120 xlabel('Range')
121 ylabel('Intensity')
122 title('Cropped Side Summed Horizontal Bars')
123 caxis([1.8e4,1.4e5])

```

APPENDIX B

SPATIAL SHEARING LIDAR SIMULATION

```

1  %% This Code is a simulation of the measurement of six spatial frequency gradients of a three
2    dimensional target.
3  % There are three main peices:
4  % 1) Reconstructing an image of the target using gradients that do not contain
5    % a random phase offset.
6  % 2) Reconstruct an image with gradients that do contain a phase offset ,
7    % but the image quality is very poor.
8  % 3) Use phase closure to remove the random phase offset , then reconstruct
9    % the image.
10 % Authors: Ryan Galloway, Zeb Barber, Dave Rabb, Jason Stafford
11
12 %% Preliminaries:
13 clear all; close all; clc;
14
15 % Choose a seed for the random numbers so the script used the same set of
16 % random numbers each time it is ran, unless the seed is changed:
17
18 rng(8); % Really destructive random phases
19 % rng(600); % what i started with for reference
20 % rng(6); % not too bad random phases
21
22 %% Create Target
23 % CAD file of the target that I am using in the lab, cropped then flipped
24 load('E:\Drive\School\AFRL\AFRL summer\Research Stuff\Code\Matlab\testbedTarget')
25 barTarget=flipud(barTarget(1:1928/2,1:1928/2));
26
27 % Create Meshgrid for x and y distances (on the target)
28 % The target is about 20cm wide, so dividing that distance up by the number
29 % of pixels:
30 [x, y] = meshgrid(-.1:.2/1928*2:-.2/1928*2);
31
32 % Define the range, as the negative of the y meshgrid, + some random
33 % roughness (to make it a diffuse target)
34 range=-y+1e-5*randn(size(y));
35
36 % This simulation is for a single wavelength using elevation and azimuth,
37 % because that is what we end up wiht in the 4 shear case after collapsing
38 % the frames into one 2d pupil plane measurement
39
40 n=96; % set the size of the object (in samples)
41 lambda=1.55e-6;
42 k = 2*pi/lambda; %Wavenumber
43 padmult=lambda*3.5/2e-5/(8*.0254);
44 padsize = round((1928/2)*padmult/2)-1928/4;
45 mid = 1928/4+padsize;
46
47 %% Here I will attempt to change the speckle size to be approximately what was seen in the lab.
48 % The smallest speckle size was 27um, and the pixels were 20um, so this
49 % could potentially cause issues with the reconstruction algorithm, and
50 % will be tested with code added to this cell
51
52 dsf = .5;
53 barTo = padarray(imresize(barTarget, dsf), [ceil((size(barTarget,1)*(1-dsf))/2), ceil((size(barTarget
54 ,1)*(1-dsf))/2)]);
55 barTo= barTo(1:964,1:964).*exp(1j*k*2*range);
56
57 barT=ft2(padarray(barTo,[padsize padsize])); % Fourier transform (with necessary shifts) so that
58 % it can be down sampled in the fourier domain (crop out high frequencies)
59 data=barT(mid-n/2:mid+n/2-1,mid-n/2:mid+n/2-1); % Down sampling the data array
60
61 %% Show what the output image would be for a standard lidar setup
62 figure;
63 imagesc(abs(ft2(padarray(data.*padarray(ones(80),[8 8]),[48 48]))))
64 title('Output Image expected from Lidar System with LO')
65 xlabel('Azimuth')
66 ylabel('Range')
67
68 %% This first part is simulating the experiment without random phases on the gradients
69
70 %% Zeb's modification; (to have non integer pixel spatial shears)
71 shearPhase = @(x_s,y_s) exp(1i*pi*(x/.2*x_s+y/.2*y_s));
72
73 % Adjust these to test different shear sizes (why does 1.5 work best?)
74 x_sh = 1.5;
75 y_sh = 1.5;
76
77 shear00=ft2(padarray(barTo.*shearPhase(0,0),[padsize padsize]));
78 data00=shear00(mid-n/2:mid+n/2-1,mid-n/2:mid+n/2-1);
79 shear01=ft2(padarray(barTo.*shearPhase(x_sh,0),[padsize padsize]));
80 data01=shear01(mid-n/2:mid+n/2-1,mid-n/2:mid+n/2-1);
81 shear10=ft2(padarray(barTo.*shearPhase(0,y_sh),[padsize padsize]));
82 data10=shear10(mid-n/2:mid+n/2-1,mid-n/2:mid+n/2-1);
83 shear11=ft2(padarray(barTo.*shearPhase(x_sh,y_sh),[padsize padsize]));
84 data11=shear11(mid-n/2:mid+n/2-1,mid-n/2:mid+n/2-1);
85 shear1m1=ft2(padarray(barTo.*shearPhase(x_sh,-y_sh),[padsize padsize]));
86 data1m1=shear1m1(mid-n/2:mid+n/2-1,mid-n/2:mid+n/2-1);

```



```

85
86 %% Create synthesized versions of the gradients:
87 % Naming Convention: transmitter locations 1,2,3,4 correspond to
88 % data00,data01,data11, and data10 respectively:
89 %
90 % 4 , data10          3, data11
91 %   o-----o
92 %   |       |
93 %   |       |
94 %   |       |
95 %   |       |
96 %   |       |
97 %   o-----o
98 %   1, data00      2, data01
99 %
100
101 % Defining the gradients , without random piston phase
102 gradx12=data00.*conj(data01);% gradx12 is from 1 to 2, bottom left to bottom right
103 gradx43=data10.*conj(data11);% grad x from 4 to 3, top left to top right
104 grady14=data00.*conj(data10);% grad y from 1 to 4, bottom left to top left
105 grady23=data01.*conj(data11);% grad y from 2 to 3, bottom right to top right
106 gradxy13=data00.*conj(data11);% grad xy from 1 to 4, bottom left to top right
107 gradxmy24=circshift(data10.*conj(data01),[-1 0]); % grad xmy from 2 to 4, bottom right to top left
    but circ shifted down by 1
108
109 %% Add some noise to the gradients:
110 % Set the simulation SNR
111 snr=10000;
112
113 % Add noise to match the SNR set above to all gradients
114 gradx12 = noise(gradx12,snr);
115 gradx43 = noise(gradx43,snr);
116 grady14 = noise(grady14,snr);
117 grady23 = noise(grady23,snr);
118 gradxy13 = noise(gradxy13,snr);
119 gradxmy24 = noise(gradxmy24,snr);
120
121 % pad the gradient fields so that they match the experimental data
122 gradx12=gradx12.*padarray(ones(80),[8 8]);
123 gradx43=gradx43.*padarray(ones(80),[8 8]);
124 grady14=grady14.*padarray(ones(80),[8 8]);
125 grady23=grady23.*padarray(ones(80),[8 8]);
126 gradxy13=gradxy13.*padarray(ones(80),[8 8]);
127 gradxmy24=gradxmy24.*padarray(ones(80),[8 8]);
128
129 % define sparse matrix and diagonals that will fill it
130 N = size(gradx12,1);
131
132 % Amp and phase are the amp and phase of the ft2 of the object
133 [amp, phase, v] = shear_recon(gradx12,grady14,gradxy13,gradxmy24);
134
135 % Construct the whole solution (combine amplitude and phase solutions)
136 solution = amp.*phase;
137
138 % Comparing the amplitude of the reconstruction to the amplitude of the
139 % standard lidar approach
140 figure;
141 imagesc(abs(abs(data)-amp)./mean(abs(data(:)))));
142 title('Actual Targ Amp.vs Reconst. Amp (No Rand Phase)')
143 caxis([0 1])
144
145 % Comparing the phase of the eigenvector solution to the phase of the
146 % standard lidar approach (right?)
147 figure;
148 imagesc(angle(reshape(v,N,N).*((data))))
149 title('Reconst. Phase vs actual targ. phase (No Rand Phase)')
150
151 % Plot the reconstructed image (ft2 first)
152 figure;
153 imagesc(abs(ift2(padarray(solution,[48 48])).^1)
154 title('Reconstructed Sheared Lidar Image (No Rand Phase)')
155 xlabel('Azimuth')
156 ylabel('Range') % or should this be elevation and range?
157 % Plot the log amplitude of the eigenvector (it is a measure of how good
158 % the reconstruction is)
159
160 figure;
161 imagesc(log(abs(reshape(v,N,N))))
162 title('Quality of Reconstruction (No Rand Phase)')
163
164 %% Re Build all of this and base it off of gradients that have a random phase on them
165 % Here we put a specific random phase offset on each gradient.
166 rand_array = 2*pi*rand(1,6);
167
168 % Defining the gradients , with random piston phase
169 gradx12=data00.*conj(data01).*exp(-1i*rand_array(1));% gradx12 is from 1 to 2, bottom left to
    bottom right
170 gradx43=data10.*conj(data11).*exp(-1i*rand_array(2));% grad x from 4 to 3, top left to top right

```

```

171 grady14=data00.*conj(data10).*exp(-1i*rand_array(3));% grad y from 1 to 4, bottom left to top left
172 grady23=data01.*conj(data11).*exp(-1i*rand_array(4));% grad y from 2 to 3, bottom right to top
    right
173 gradxy13=data00.*conj(data11).*exp(-1i*rand_array(5));% grad xy from 1 to 4, bottom left to top
    right
174 gradxmy24=circshift(data10.*conj(data01),[-1 0]).*exp(-1i*rand_array(6)); % grad xmy from 2 to 4,
    bottom left to top right but circ shifted down by 1
175
176 % Add noise to each gradient such that it has an SNR as defined earlier
177 gradx12 = noise(gradx12,snr);
178 gradx43 = noise(gradx43,snr);
179 grady14 = noise(grady14,snr);
180 grady23 = noise(grady23,snr);
181 gradxy13 = noise(gradxy13,snr);
182 gradxmy24 = noise(gradxmy24,snr);
183
184 % pad the gradient fields so that they match the experimental data
185 gradx12=gradx12.*padarray(ones(80),[8 8]);
186 gradx43=gradx43.*padarray(ones(80),[8 8]);
187 grady14=grady14.*padarray(ones(80),[8 8]);
188 grady23=grady23.*padarray(ones(80),[8 8]);
189 gradxy13=gradxy13.*padarray(ones(80),[8 8]);
190 gradxmy24=gradxmy24.*padarray(ones(80),[8 8]);
191
192 % define sparse matrix and diagonals that will fill it
193 N = size(gradx12,1);
194
195 % Amp and phase are the amp and phase of the ft2 of the object
196 [amp, phase, v] = shear_recon(gradx12,grady14,gradxy13,gradxmy24);
197
198 % Construct the whole solution (combine amplitude and phase solutions)
199 solution = amp.*phase;
200
201 % Comparing the amplitude of the reconstruction to the amplitude of the
202 % standard lidar approach
203 figure;
204 imagesc(abs(abs(data)-amp)./mean(abs(data(:)))));
205 title('Actual Targ Amp.vs Reconst. Amp (With Rand Phase)')
206 caxis([0 1])
207
208 % Comparing the phase of the eigenvector solution to the phase of the
209 % standard lidar approach (right?)
210 figure;
211 imagesc(angle(reshape(v,N,N).*((data)))));
212 title('Reconst. Phase vs actual targ. phase (With Rand Phase)')
213
214 % Plot the reconstructed image (ft2 first)
215 figure;
216 imagesc(abs(ift2(padarray(solution,[48 48]))).^1)
217 title('Reconstructed Sheared Lidar Image (With Rand Phase)')
218 xlabel('Azimuth')
219 ylabel('Range') % or should this be elevation and range?
220 % Plot the log amplitude of the eigenvector (it is a measure of how good
221 % the reconstruction is)
222 figure;
223 imagesc(log(abs(reshape(v,N,N)))));
224 title('Quality of Reconstruction (With Rand Phase)')
225
226
227 %% Now we would like to attempt to remove the random phase seen in the previous section
228 %% Solve for phase offsets using phase closure:
229
230 % Use my fancy new function to solve for the closure phase
231 [p, p_checks] = phase_closure(gradx12,grady14,gradxy13,gradxmy24,gradx43,grady23);
232
233 %check to make sure that there is a global phase offset for each closure
234 figure; imagesc(angle(exp(1i*p_checks(:,:1))))); colorbar; title('p12341 (Phase Closure around the
    square)')
235 figure; imagesc(angle(exp(1i*p_checks(:,:2))))); colorbar; title('p1231 (Phase Closure around
    Triangle #1)')
236 figure; imagesc(angle(exp(1i*p_checks(:,:3))))); colorbar; title('p1341 (Phase Closure around
    Triangle #2)')
237 figure; imagesc(angle(exp(1i*p_checks(:,:4))))); colorbar; title('p2342 (Phase Closure around
    Triangle #3)')
238 figure; imagesc(angle(exp(1i*p_checks(:,:5))))); colorbar; title('p2142 (Phase Closure around
    Triangle #4)')
239
240 % Now use the computed phases to correct the original gradients:
241 gradx12=gradx12.*exp(-1i*p(1)); % grad x from 1 to 2, bottom left to bottom right
242 grady14=grady14.*exp(-1i*p(2)); % grad y from 1 to 4, bottom left to top left
243 gradxy13=gradxy13.*exp(-1i*p(3)); % grad xy from 1 to 4, bottom left to top right
244 gradxmy24=gradxmy24.*exp(-1i*p(4)); % grad xmy from 2 to 4, bottom left to top right
245 gradx43=gradx43.*exp(-1i*p(5)); % grad x from 4 to 3, top left to top right (Not used for
    reconstruction)
246 grady23=grady23.*exp(-1i*p(6)); % grad y from 2 to 3, bottom right to top right (Not used for
    reconstruction)
247
248

```

```

249 %% Reconstruct the phase and amplitude of the fourier transform of the object
250 N = size(gradx12,1);
251
252 % Amp and phase are the amp and phase of the ft2 of the object
253 [amp, phase, v] = shear_recon(gradx12,gradyl4,gradxy13,gradxmy24);
254
255 % Construct the whole solution (combine amplitude and phase solutions)
256 solution = amp.*phase;
257
258 % Comparing the amplitude of the reconstruction to the amplitude of the
259 % standard lidar approach
260 figure;
261 imagesc(abs(abs(data)-amp)./mean(abs(data(:)))));
262 title('Actual Targ Amp.vs Reconst. Amp (Fixed Phase)')
263 caxis([0 1])
264
265 % Comparing the phase of the eigenvector solution to the phase of the
266 % standard lidar approach (right?)
267 figure;
268 imagesc(angle(reshape(v,N,N).*((data)))));
269 title('Reconst. Phase vs actual targ. phase (Fixed Phase)')
270
271 % Plot the reconstructed image (ft2 first) oh and shift it so it is roughly
272 % centered
273 figure;
274 imagesc(circshift(abs(ift2(padarray(solution,[48 48]))).^1,[-50,-20]))
275 title('Reconstructed Sheared Lidar Image (Fixed Phase) (Circ Shifted)')
276 xlabel('Azimuth')
277 ylabel('Range') % or should this be elevation and range?
278
279 % Plot the reconstructed image (ft2 first)
280 figure;
281 imagesc(abs(ift2(padarray(solution,[48 48]))).^1)
282 title('Reconstructed Sheared Lidar Image (Fixed Phase) (Not Circ Shifted)')
283 xlabel('Azimuth')
284 ylabel('Range') % or should this be elevation and range?
285
286
287 % Plot the log amplitude of the eigenvector (it is a measure of how good
288 % the reconstruction is)
289 figure;
290 imagesc(log(abs(reshape(v,N,N)))));
291 title('Quality of Reconstruction (Fixed Phase)')

```

APPENDIX C

SPATIAL SHEARING LIDAR EXPERIMENTAL DATA PROCESSING

```

1 % This script does all of the processing necessary to synthesize images
2 % from a spatial spectral shearing Lidar system.
3
4 % Written by Ryan Galloway, Zeb Barber, Dave Rabb, and Jason Stafford
5 clear all
6 close all
7
8 %% Define Constants, and Parameters of the experimental data.
9 stable_length = 43e-3; % Stable time duration of the laser chirp
10 chirp_BW = 50e9; % Total bandwidth of optical chirp
11 frames = 1792; % Total number of frames taken from camera
12 t0 = 950e-6; % Start time for stable chirp region
13 fr = 39840; % Frame rate of high speed camera
14 dt = 1/fr; % Time steps
15 bad_frames = ceil(t0/dt); % Number of bad frames to be removed
16 frame_stop = ceil(stable_length*fr); % Where to cut off the number of frames
17 t = (1:(frame_stop))*dt-dt; % Time space
18 f = linspace(-1/(2*dt), 1/(2*dt), frame_stop); % Frequency Space
19 c = 2.998e8; % Speed of light
20 d = 20e-6; % Pixel Size
21 z = 3.5; % Distance from transmitter to target
22 lambda = 1550e-9; % Center Wavelength of the laser
23 rows = 16; % Number of rows in a frame from the camera
24 columns = 80; % Number of columns in a frame from the camera
25
26 % Calculate the effective bandwidth and the corresponding range resolution.
27 actual_BW = (((frames-bad_frames)*dt)/stable_length)*chirp_BW;
28 % Calculate the range resolution Zres = C/2B
29 res_range = c/(2*actual_BW);
30 % azimuth aka cross range, (pixel size*number of pixels)= aperture size
31 res_azimuth = (lambda*z)/(d*columns);
32
33 %% This for loop will create the gradients to be used in the reconstruction
34 % for all 9 chirps worth of data.
35
36 for k = 1:9
37
38     %% Import data from Xenics camera:
39
40     % Define the path:
41     filename = ['E:\Drive\School\AFRL\AFRL summer\Research Stuff\data\8.7.2017\4-mirrors-16-475-2-
42         ', num2str(k), '.bin'];
43
44     % Read .bin file, retrieving the data as a row x column x frames data cube and the metadata
45     % as a struct file
46     [timeData, metadata] = readXenicsData(filename);
47
48     % Crop out the first set of bad frames, and stop before the end of the chirp
49     timeData = timeData(:, :, bad_frames:frame_stop+bad_frames-1);
50
51     %% Load mirror calibration data
52     % This calibration data tells us how to position the experimental data
53     % for demultiplexing into the different spatial shears.
54
55     % Loads the variables gregs, locs, and spacing from the mirror calibration test.
56     load('E:\Drive\School\AFRL\AFRL summer\Research Stuff\data\8.1.2017\caldata.mat')
57
58     %% Demultiplexing the spatial shears using the temporal shears
59
60     % This is the fourier transform of the time data along the 3rd dimension
61     spectrum = double((((fftshift(fft(fftshift(timeData, 3), [], 3), 3)))));
62
63     % crop each frequency peak out based off of peak locations from mirror
64     % calibration data
65     f12 = spectrum(:, :, locs(1)-floor(spacing/2):locs(1)+floor(spacing/2)-1);
66     f34 = spectrum(:, :, locs(2)-floor(spacing/2):locs(2)+floor(spacing/2)-1);
67     f23 = spectrum(:, :, locs(3)-floor(spacing/2):locs(3)+floor(spacing/2)-1);
68     f13 = spectrum(:, :, locs(4)-floor(spacing/2):locs(4)+floor(spacing/2)-1);
69     f24 = spectrum(:, :, locs(5)-floor(spacing/2):locs(5)+floor(spacing/2)-1);
70     f14 = spectrum(:, :, locs(6)-floor(spacing/2):locs(6)+floor(spacing/2)-1);
71
72     % Center each autocorrelation term
73     f12 = ifft(bsxfun(@times, fft(f12, [], 3), permute(conj(gregs(:, 1)), [3, 2, 1])), [], 3);
74     f34 = ifft(bsxfun(@times, fft(f34, [], 3), permute(conj(gregs(:, 2)), [3, 2, 1])), [], 3);
75     f23 = ifft(bsxfun(@times, fft(f23, [], 3), permute(conj(gregs(:, 3)), [3, 2, 1])), [], 3);
76     f13 = ifft(bsxfun(@times, fft(f13, [], 3), permute(conj(gregs(:, 4)), [3, 2, 1])), [], 3);
77     f24 = ifft(bsxfun(@times, fft(f24, [], 3), permute(conj(gregs(:, 5)), [3, 2, 1])), [], 3);
78     f14 = ifft(bsxfun(@times, fft(f14, [], 3), permute(conj(gregs(:, 6)), [3, 2, 1])), [], 3);
79
80     % fourier transform because we want to register each frequency step not
81     % time step (from frame to frame) and pad it so we can sum them later.
82     f12 = padarray(fftshift(fft(fftshift(f12, 3), [], 3), 3), [40 8 0]);
83     f34 = padarray(fftshift(fft(fftshift(f34, 3), [], 3), 3), [40 8 0]);
84     f23 = padarray(fftshift(fft(fftshift(f23, 3), [], 3), 3), [40 8 0]);
85     f13 = padarray(fftshift(fft(fftshift(f13, 3), [], 3), 3), [40 8 0]);
86     f24 = padarray(fftshift(fft(fftshift(f24, 3), [], 3), 3), [40 8 0]);
87     f14 = padarray(fftshift(fft(fftshift(f14, 3), [], 3), 3), [40 8 0]);

```

```

86
87 % concatenate all of the frequencies together and use that as an input to
88 % dftreg_datacube and put in a large upsample factor to solve for the
89 % subpixel shift that should be used on the frequencies
90 f_cat_1 = cat(3,f12(:,:2:end),f34(:,:2:end),f23(:,:2:end),f13(:,:2:end),f24(:,:2:end),f14
    (:,:2:end));
91 f_cat_2 = cat(3,f12(:,:1:end-1),f34(:,:1:end-1),f23(:,:1:end-1),f13(:,:1:end-1),f24(:,:1:
    end-1),f14(:,:1:end-1));
92 [output_cat, Greg] = dftreg_datacube(fft2(f_cat_1),fft2(f_cat_2),1000);
93
94 % create a 3d version of greg, where the difference between each layer is greg
95 greg_3d = zeros(size(f12));
96 for jj = -64:63
97     greg_3d(:,:,jj+65) = -jj*angle(Greg);
98 end
99
100 % Implement the shifting as a fourier shift
101 greg_3d = exp(1i.*greg_3d);
102
103 % Shift and sum the 128 frames that belong with each gradient to go
104 % from a 3d dataset for each gradient to a 2d dataset, where the
105 % relationship between elevation and range has been utilized.
106 f12_shots(:,:,k) = sum(ifft2(fft2(f12).*greg_3d),3);
107 f34_shots(:,:,k) = sum(ifft2(fft2(f34).*greg_3d),3);
108 f23_shots(:,:,k) = sum(ifft2(fft2(f23).*greg_3d),3);
109 f13_shots(:,:,k) = sum(ifft2(fft2(f13).*greg_3d),3);
110 f24_shots(:,:,k) = sum(ifft2(fft2(f24).*greg_3d),3);
111 f14_shots(:,:,k) = sum(ifft2(fft2(f14).*greg_3d),3);
112
113 %% Upsample in the fourier domain
114 % By registering the spatial shear size between the redundant x
115 % gradients and redundant y gradients in the bar target data, we
116 % concluded that our spatial shears were not actually a single pixel,
117 % but 1.2 pixels vertically and 1.5 pixels horizontally. This required
118 % us to resample our gradients so that we end up with a single pixel
119 % shift in each direction. This is necessary to reconstruct using the
120 % covariance solution algorithm.
121
122 % Real shears:
123 actual_x_shear = 1.5;
124 actual_y_shear = 1.2;
125
126 % Our array is 96 by 96 and we want to pad it in the fourier domain by
127 % different amounts in each dimension so we get spatial shears of size 2
128 padx = (((2/actual_x_shear)*96)-96)/2;
129 pady = ceil((((2/actual_y_shear)*96)-96)/2);
130
131 % Perform the resampling:
132 f12up = ift2(padarray(fft2(f12_shots(:,:,k)),[pady,padx]));
133 f34up = ift2(padarray(fft2(f34_shots(:,:,k)),[pady,padx]));
134 f14up = ift2(padarray(fft2(f14_shots(:,:,k)),[pady,padx]));
135 f23up = ift2(padarray(fft2(f23_shots(:,:,k)),[pady,padx]));
136 f13up = ift2(padarray(fft2(f13_shots(:,:,k)),[pady,padx]));
137 f24up = ift2(padarray(fft2(f24_shots(:,:,k)),[pady,padx]));
138
139 % Pad it back to an array that is 192 by 192, then use imfilter to
140 % downsample each group of four pixels to one pixel value.
141 f12up = imfilter(padarray(f12up,[16,32]),ones(2,2)/4);
142 f34up = imfilter(padarray(f34up,[16,32]),ones(2,2)/4);
143 f14up = imfilter(padarray(f14up,[16,32]),ones(2,2)/4);
144 f23up = imfilter(padarray(f23up,[16,32]),ones(2,2)/4);
145 f13up = imfilter(padarray(f13up,[16,32]),ones(2,2)/4);
146 f24up = imfilter(padarray(f24up,[16,32]),ones(2,2)/4);
147
148 % Resample to get back to a 96 by 96 gradient, that now has a shear of
149 % size 1 in both dimensions.
150 f12up = f12up(1:2:end,1:2:end);
151 f34up = f34up(1:2:end,1:2:end);
152 f14up = f14up(1:2:end,1:2:end);
153 f23up = f23up(1:2:end,1:2:end);
154 f13up = f13up(1:2:end,1:2:end);
155 f24up = f24up(1:2:end,1:2:end);
156
157 end
158
159 %% This for loop synthesizes an image from each set of gradients from all 9 chirps worth of data.
160 % Starts off by removing the DC peak of each gradient. Then uses a phase
161 % closure algorithm to correct the unknown phase on each gradient. Then it
162 % will synthesize an image using the shear reconstruction algorithm.
163
164 for f = 1:9
165     % Pick out the gradients from one shot at a time
166     gradx12 = f12_shots(:,:,f);
167     gradx43 = f34_shots(:,:,f);
168     grady14 = f14_shots(:,:,f);
169     grady23 = f23_shots(:,:,f);
170     gradxy13 = f13_shots(:,:,f);
171     gradxmy24 = f24_shots(:,:,f);

```

```

172
173 %% Crop out the DC peak in the fourier transform of the gradients:
174 % We noticed that if you leave the DC content in the gradients, it
175 % leads to a lot of artifacts in the reconstructed images that were
176 % clearly not image content.
177
178 % fourier transform
179 fg12 = ft2(gradx12);
180 fg43 = ft2(gradx43);
181 fg14 = ft2(grady14);
182 fg23 = ft2(grady23);
183 fg13 = ft2(gradxy13);
184 fg24 = ft2(gradxmy24);
185
186 % Zero out the DC peak
187 fg12(37:60,30:70) = 0;
188 fg43(37:60,30:70) = 0;
189 fg14(37:60,30:70) = 0;
190 fg23(37:60,30:70) = 0;
191 fg13(37:60,30:70) = 0;
192 fg24(37:60,30:70) = 0;
193
194 % Fourier transform back
195 gradx12 = ift2(fg12);
196 gradx43 = ift2(fg43);
197 grady14 = ift2(fg14);
198 grady23 = ift2(fg23);
199 gradxy13 = ift2(fg13);
200 gradxmy24 = ift2(fg24);
201
202 %% Solve for phase offsets using phase closure (Zeb and Ryans Idea)
203 % Define the phase around a square and 4 triangles
204 % Notation: the phases at each location of the transmitting square pattern
205 % are denoted as 1(bottom left) 2(bottom right) 3(top right) 4(top left),
206 % so a variable named p12341 is the phase from 1 to 2 , 2 to 3, 3 to 4 , and
207 % back from 4 to 1 in a square
208
209 % Diagram of the gradients as a reference
210 %
211 %      4      gradx43      3
212 %      o----->o
213 %      |         |
214 % grady14 |         | grady23
215 %      |         |
216 %      o----->o
217 %      1      gradx12      2
218 %
219 %
220 % also gradxy13 from bottom left to top right
221 % and gradxmy24 from bottom right to top left
222
223 % Calculate the closure phases for 5 different loops
224 % The signs on these were based off of many iterations guess and check,
225 % where we were using the figures below to evaluate if there was a
226 % global phase offset that needed to be removed (it was also based off
227 % of simulations where we removed a random phase from simulated data)
228 p12341 = angle(gradx12)+angle(grady23)+angle(gradx43)-angle(grady14); %square
229 p1231 = angle(gradx12)+angle(grady23)-angle(gradxy13); %triangle #1
230 p1341 = angle(gradxy13)+angle(gradx43)-angle(grady14); %triangle #2
231 p2342 = angle(grady23)+angle(gradx43)-angle(gradxmy24); %triangle #3
232 p2142 = -angle(gradx12)-angle(grady14)+angle(gradxmy24); %triangle #4
233
234 % Check to make sure that there is a global phase offset for each closure
235 % figure; imagesc(angle(exp(1i*p12341))); colorbar; title('p12341 (Phase Closure around the
square)')
236 % figure; imagesc(angle(exp(1i*p1231))); colorbar; title('p1231 (Phase Closure around Triangle
#1)')
237 % figure; imagesc(angle(exp(1i*p1341))); colorbar; title('p1341 (Phase Closure around Triangle
#2)')
238 % figure; imagesc(angle(exp(1i*p2342))); colorbar; title('p2342 (Phase Closure around Triangle
#3)')
239 % figure; imagesc(angle(exp(1i*p2142))); colorbar; title('p2142 (Phase Closure around Triangle
#4)')
240
241 % Calculate the mean for each phase closure (after wrapping the phase back to between -pi and
pi)
242 m12341 = mean2(angle(exp(1i*p12341(15:84,10:89))));
243 m1231 = mean2(angle(exp(1i*p1231(15:84,10:89))));
244 m1341 = mean2(angle(exp(1i*p1341(15:84,10:89))));
245 m2342 = mean2(angle(exp(1i*p2342(15:84,10:89))));
246 m2142 = mean2(angle(exp(1i*p2142(15:84,10:89))));
247
248 % This is the matrix for the system of equations which describe the 4
249 % different phase closure checks above
250 c = [1,0,-1,0,0,1;0,-1,1,0,-1,0;0,0,0,1,-1,1;-1,1,0,1,0,0];
251
252 % Construct the closure phase values matrix
253 p-close = [m1231;m1341;m2342;m2142];

```

```

254 % Solve for each random phase on the gradients:
255 % Note: This will give a rand deficient warning, that is because there
256 % are two rows of the c matrix that are not linearly independent. We
257 % cannot decouple them, so we just left it. It still works, but gives a
258 % warning each time it is used.
259 p = c\p_close;
260
261 % Now use the computed phases to correct the original gradients:
262 gradx12=gradx12.*exp(-1i*p(1));
263 grady14=grady14.*exp(-1i*p(2));
264 gradxy13=gradxy13.*exp(-1i*p(3));
265 gradxmy24=gradxmy24.*exp(-1i*p(4));
266 gradx43=gradx43.*exp(-1i*p(5));
267 grady23=grady23.*exp(-1i*p(6));
268
269 %% Crop down the gradients so there is no padding around the gradients
270 % We thought that the padding might have been negatively affecting the
271 % quality of the reconstructions, because we were telling it that some
272 % of the gradients were zero, which wouldn't make sense.
273
274 % Also these have to be square!
275 gradx12 = gradx12(18:77,18:77);
276 grady14 = grady14(18:77,18:77);
277 gradxy13 = gradxy13(18:77,18:77);
278 gradxmy24 = gradxmy24(18:77,18:77);
279
280 %% Reconstruct the phase and amplitude of the fourier transform of the object
281
282 % The first and last inputs to this function are conjugated. This was
283 % also determined via guess and check. We looked at all the possible
284 % combinations of conjugations of the gradients (since they are unknown
285 % in an experiment) and looked for one that gave us an image.
286 [amp, phase, v] = shear_recon.mod2(conj(gradx12),(grady14),(gradxy13),conj(gradxmy24));
287
288 % Construct the whole solution (combine amplitude and phase solutions)
289 solution = amp.*phase;
290
291 % figure;
292 % imagesc(abs(ift2(padarray(solution,[48 48]))).^2)
293 % title(strcat('Reconstructed Sheared Lidar 4 Mirror Image. conj. config. #',num2str(i),' ',
294 % chirp # ',num2str(f),' shear size = 1pix'))
295 % xlabel('Azimuth')
296 % ylabel('Range')
297
298 % Collect each solution as it is generated
299 solution_stack(:, :, f) = solution;
300
301 end
302
303 %% Register the output image from each chirp, then incoherently sum them
304
305 % Registering to the brightest solution from chirp #7:
306 for kk = 1:9
307     [out, ggreg] = dftregistration(solution_stack(:, :, 7), solution_stack(:, :, kk), 100);
308     soln(:, :, kk) = ggreg;
309 end
310
311 % Pad the solution, fourier transform in 2 dimensions, sum them along the
312 % 3rd dimension, and circshift it to the center.
313 sum_soln = circshift(sum(ft2(padarray(soln,[48,48])),3)/9,[-22,50]);
314
315 % Plot the final image
316 figure; imagesc(abs(sum_soln));
317 title('Sheared Lidar Reconstructed, Incoherent Summed Image of 4 Mirrors from 9 Chirps')
318 xlabel('Cross Range')
319 ylabel('Elevation/Range')

```


APPENDIX D

MATLAB FUNCTIONS

```

1 function g = ft2(g)
2 % G is the sampled Fourier Spectrum
3
4 g = ifftshift(fftshift(g,1),2);
5 g = fft2(g);
6 g = fftshift(fftshift(g,1),2);
7
8 end

```

```

1 function g = ift2(g)
2
3 g = ifftshift(fftshift(g,1),2);
4 g = ifft2(g);
5 g = fftshift(fftshift(g,1),2);
6
7 end

```

```

1 function [ amp,phase,v ] = shear_recon_mod2( gradx12,grady14,gradxy13,gradxmy24)
2 %Shear Recon is a function that inputs four gradients, and four phase closure values and
3   reconstructs an
4   %output field.
5
6 % The idea of this mod was for it to take non square gradients, but it does
7 % not work, so only input square arrays!
8
9 %% define sparse matrix and diagonals that will fill it
10 Nx = size(gradx12,2);
11 Ny = size(gradx12,1);
12 covMat2=spalloc(Nx*Ny,Nx*Ny,4*Nx*Ny);
13
14 %% Placing diagonals into covariance matrix
15
16 % Define xplus and yplus as empty covariance matrix
17 xplus = covMat2;
18 yplus = covMat2;
19 % Lay their gradients into the diagonals of the matrix
20 xplus = spdiags(gradx12(:),Nx,xplus); % sparse matrix with one diagonal at +96
21 yplus = spdiags(grady14(:),1,yplus); % sparse matrix with one diagonal at +1
22 xminus = xplus'; %conjugates of the above diagonals
23 yminus = yplus';
24 xyplus = covMat2;
25 xyplus = spdiags(gradxy13(:),Nx+2,xyplus);
26 xyminus = xyplus';
27 xplusyminus = covMat2;
28 xplusyminus = spdiags(gradxmy24(:),Nx-2,xplusyminus);
29 xminusyplus = xplusyminus';
30
31 % Add all of the sparse matrix diagonals together, and scale it
32 covMat2=(xplus+yplus+xminus+yminus+xyplus+xyminus+xplusyminus+xminusyplus)/1e5;
33
34 % Scale the magnitude of the covariance matrix so it is only used for
35 % finding the phase solution
36 mag=sum(abs(covMat2),1);
37 magcorr=spdiags(1./(mag'),0,Nx*Ny,Nx*Ny);
38 covMat2=covMat2*magcorr;
39 mag=sum(abs(covMat2),1);
40
41 % Solve for the eigenvector (phase solution)
42 [v,d]=eigs(covMat2,1,'LR');
43
44 % These are the amplitude relationships between gradients
45 g1=abs(grady14);
46 g2=abs(gradxy13);
47 g3=abs(gradx12);
48 g4=abs(gradxmy24);
49 g5=abs(circshift(grady14,[-1,0]));
50 g6=abs(circshift(gradxy13,[-1,-1]));
51 g7=abs(circshift(gradx12,[0,-1]));
52 g8=abs(circshift(gradxmy24,[1,-1]));
53 g9=abs(circshift(gradxmy24,[1,0]));
54 g10=abs(circshift(gradxy13,[-1,0]));
55 g11=abs(circshift(gradxmy24,[0,-1]));
56 g12=abs(circshift(gradxy13,[0,-1]));
57 g13=abs(circshift(gradx12,[1,0]));
58 g14=abs(circshift(grady14,[0,1]));
59 g15=abs(circshift(grady14,[-1,1]));
60 g16=abs(circshift(gradx12,[-1,0]));
61 g17=abs(circshift(gradx12,[-1,-1]));
62 g18=abs(circshift(grady14,[-1,-1]));
63 g19=abs(circshift(grady14,[0,-1]));
64 g20=abs(circshift(gradx12,[1,-1]));
65
66 % Solve for the amplitude of the output image:
67 amp = sqrt((g1.*(g8+g2+g3)+g3.*(g2+g4+g5)+g5.*(g4+g6+g7)+g7.*(g6+g8+g1))./( ...
   (g9+g10+g11+g12+g13+g14+g15+g16+g17+g18+g19+g20+eps)));

```

```

68
69 % Create the phase solution:
70 phase = reshape(exp(1j.*(angle(v))),Ny,Nx);
71
72
73 end

```

```

1 function [ p, p-checks ] = phase_closure( gradx12,grady14,gradxy13,gradxmy24,gradx43,grady23 )
2 % Define the phase around a square and 4 triangles
3 % Notation: the phases at each location of the transmitting square pattern
4 % are denoted as 1(bottom left) 2(bottom right) 3(top right) 4(top left),
5 % so a variable named p12341 is the phase from 1 to 2 , 2 to 3, 3 to 4 , and
6 % back from 4 to 1 in a square
7
8 % Subtract off the large phase offsets from the x and y gradients that will
9 % be used in the reconstruction:
10 gradx12=gradx12.*exp(-1i*mean2(angle(gradx12))); % grad x from 1 to 2, bottom left to bottom right
11 grady14=grady14.*exp(-1i*mean2(angle(grady14))); % grad y from 1 to 4, bottom left to top left
12 gradxy13=gradxy13.*exp(-1i*mean2(angle(gradxy13))); % grad xy from 1 to 4, bottom left to top
13 % right
14 gradxmy24=gradxmy24.*exp(-1i*mean2(angle(gradxmy24))); % grad xmy from 2 to 4, bottom left to top
15 % right
16 gradx43=gradx43.*exp(-1i*mean2(angle(gradx43))); % grad x from 4 to 3, top left to top right (Not
17 % used for reconstruction)
18 grady23=grady23.*exp(-1i*mean2(angle(grady23))); % grad y from 2 to 3, bottom right to top right (
19 % Not used for reconstruction)
20
21 % Calculate the closure phases for 5 different loops
22 p12341 = angle(gradx12)+angle(grady23)-angle(gradx43)-angle(grady14); %square
23 p1231 = angle(gradx12)+angle(grady23)-angle(gradxy13); %triangle #1
24 p1341 = angle(gradxy13)-angle(gradx43)-angle(grady14); %triangle #2
25 p2342 = angle(grady23)-angle(gradx43)+angle(circshift(gradxmy24,[1,0])); %triangle #3 circshift
26 % needed to undo the shift that was put on it when gradxmy24 was defined
27 p2142 = -angle(gradx12)+angle(grady14)+angle(circshift(gradxmy24,[1,0])); %triangle #4 circshift
28 % needed to undo the shift that was put on it when gradxmy24 was defined
29
30 p-checks = cat(3,p12341,p1231,p1341,p2342,p2142);
31
32 % Calculate the mean (using logical indexing to exlude the zeros in the padding) for each
33 % phase closure (after wrapping the phase back to between -pi and pi)
34 m12341 = angle(mean2(exp(1i*p12341(gradx12 ~= 0))));
35 m1231 = angle(mean2(exp(1i*p1231(gradx12 ~= 0))));
36 m1341 = angle(mean2(exp(1i*p1341(gradx12 ~= 0))));
37 m2342 = angle(mean2(exp(1i*p2342(gradx12 ~= 0))));
38 m2142 = angle(mean2(exp(1i*p2142(gradx12 ~= 0))));
39
40 % Construct closure matrix c
41 % This is the matrix for the system of equations which describe the 4
42 % different phase closure checks above
43 c = [1,0,-1,0,0,1;0,-1,1,0,-1,0;0,0,0,1,-1,1;-1,1,0,1,0,0];
44
45 % Construct the closure phase values matrix
46 p-close = [m1231;m1341;m2342;m2142];
47
48 % Solve for each random phase that was added to the gradients earlier:
49 p = c\p_close; % This method works the best
50 % p = pinv(c)*p_close;
51
52 end

```

```

1 function [output, Greg] = dftregistration(buf1ft,buf2ft,usfac)
2 % function [output Greg] = dftregistration(buf1ft,buf2ft,usfac);
3 % Efficient subpixel image registration by crosscorrelation. This code
4 % gives the same precision as the FFT upsampled cross correlation in a
5 % small fraction of the computation time and with reduced memory
6 % requirements. It obtains an initial estimate of the crosscorrelation peak
7 % by an FFT and then refines the shift estimation by upsampling the DFT
8 % only in a small neighborhood of that estimate by means of a
9 % matrix-multiply DFT. With this procedure all the image points are used to
10 % compute the upsampled crosscorrelation.
11 % Manuel Guizar - Dec 13, 2007
12 %
13 % Rewrote all code not authored by either Manuel Guizar or Jim Fienup
14 % Manuel Guizar - May 13, 2016
15 %
16 % Citation for this algorithm:
17 % Manuel Guizar-Sicairos, Samuel T. Thurman, and James R. Fienup,
18 % Efficient subpixel image registration algorithms, Opt. Lett. 33,
19 % 156-158 (2008).
20 %
21 % Inputs
22 % buf1ft      Fourier transform of reference image,
23 %             DC in (1,1) [DO NOT FFTSHIFT]
24 % buf2ft      Fourier transform of image to register,

```

```

25 %          DC in (1,1) [DO NOT FFTSHIFT]
26 % usfac    Upsampling factor (integer). Images will be registered to
27 %          within 1/usfac of a pixel. For example usfac = 20 means the
28 %          images will be registered within 1/20 of a pixel. (default = 1)
29 %
30 % Outputs
31 % output = [error,diffphase,net_row_shift,net_col_shift]
32 % error    Translation invariant normalized RMS error between f and g
33 % diffphase Global phase difference between the two images (should be
34 %          zero if images are non-negative).
35 % net_row_shift net_col_shift Pixel shifts between images
36 % Greg      (Optional) Fourier transform of registered version of buf2ft,
37 %          the global phase difference is compensated for.
38 %
39 %
40 % Copyright (c) 2016, Manuel Guizar Sicaireos, James R. Fienup, University of Rochester
41 % All rights reserved.
42 %
43 % Redistribution and use in source and binary forms, with or without
44 % modification, are permitted provided that the following conditions are
45 % met:
46 %
47 % * Redistributions of source code must retain the above copyright
48 % notice, this list of conditions and the following disclaimer.
49 % * Redistributions in binary form must reproduce the above copyright
50 % notice, this list of conditions and the following disclaimer in
51 % the documentation and/or other materials provided with the distribution
52 %
53 % * Neither the name of the University of Rochester nor the names
54 % of its contributors may be used to endorse or promote products derived
55 % from this software without specific prior written permission.
56 %
57 % THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS AS IS
58 % AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
59 % IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
60 % ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE
61 % LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
62 % CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
63 % SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
64 % INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
65 % CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
66 % ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
67 % POSSIBILITY OF SUCH DAMAGE.
68
69 if ~exist('usfac','var')
70     usfac = 1;
71 end
72
73 [nr,nc]=size(buf2ft);
74 Nr = ifftshift(-fix(nr/2):ceil(nr/2)-1);
75 Nc = ifftshift(-fix(nc/2):ceil(nc/2)-1);
76
77 if usfac == 0
78     % Simple computation of error and phase difference without registration
79     CCmax = sum(buf1ft(:).*conj(buf2ft(:)));
80     row_shift = 0;
81     col_shift = 0;
82 elseif usfac == 1
83     % Single pixel registration
84     CC = ifft2(buf1ft.*conj(buf2ft));
85     CCabs = abs(CC);
86     [row_shift, col_shift] = find(CCabs == max(CCabs(:)));
87     CCmax = CC(row_shift,col_shift)*nr*nc;
88     % Now change shifts so that they represent relative shifts and not indices
89     row_shift = Nr(row_shift);
90     col_shift = Nc(col_shift);
91 elseif usfac > 1
92     % Start with usfac == 2
93     CC = ifft2(FTpad(buf1ft.*conj(buf2ft),[2*nr,2*nc]));
94     CCabs = abs(CC);
95     [row_shift, col_shift] = find(CCabs == max(CCabs(:)),1,'first');
96     CCmax = CC(row_shift,col_shift)*nr*nc;
97     % Now change shifts so that they represent relative shifts and not indices
98     Nr2 = ifftshift(-fix(nr):ceil(nr)-1);
99     Nc2 = ifftshift(-fix(nc):ceil(nc)-1);
100    row_shift = Nr2(row_shift)/2;
101    col_shift = Nc2(col_shift)/2;
102    % If upsampling > 2, then refine estimate with matrix multiply DFT
103    if usfac > 2,
104        %%% DFT computation %%%
105        % Initial shift estimate in upsampled grid
106        row_shift = round(row_shift*usfac)/usfac;
107        col_shift = round(col_shift*usfac)/usfac;
108        dftshift = fix(ceil(usfac*1.5)/2); %%% Center of output array at dftshift+1
109        % Matrix multiply DFT around the current shift estimate
110        CC = conj(dftups(buf2ft.*conj(buf1ft),ceil(usfac*1.5),ceil(usfac*1.5),usfac,...
111            dftshift-row_shift*usfac,dftshift-col_shift*usfac));
112        % Locate maximum and map back to original pixel grid
113        CCabs = abs(CC);

```

```

113     [rloc, cloc] = find(CCabs == max(CCabs(:)), 1, 'first');
114     CCmax = CC(rloc, cloc);
115     rloc = rloc - dftshift - 1;
116     cloc = cloc - dftshift - 1;
117     row_shift = row_shift + rloc/usfac;
118     col_shift = col_shift + cloc/usfac;
119 end
120
121 % If its only one row or column the shift along that dimension has no
122 % effect. Set to zero.
123 if nr == 1,
124     row_shift = 0;
125 end
126 if nc == 1,
127     col_shift = 0;
128 end
129
130 end
131
132 rg00 = sum(abs(buf1ft(:)).^2);
133 rf00 = sum(abs(buf2ft(:)).^2);
134 error = 1.0 - abs(CCmax).^2/(rg00*rf00);
135 error = sqrt(abs(error));
136 diffphase = angle(CCmax);
137
138 output=[error, diffphase, row_shift, col_shift];
139
140 % Compute registered version of buf2ft
141 if (nargout > 1)&&(usfac > 0),
142     [Nc, Nr] = meshgrid(Nc, Nr);
143     Greg = buf2ft.*exp(1i*2*pi*(-row_shift*Nr/nr-col_shift*Nc/nc));
144     Greg = Greg*exp(1i*diffphase);
145 elseif (nargout > 1)&&(usfac == 0)
146     Greg = buf2ft*exp(1i*diffphase);
147 end
148 return
149
150 function out=dftups(in, nor, noc, usfac, roff, coff)
151 % function out=dftups(in, nor, noc, usfac, roff, coff);
152 % Upsampled DFT by matrix multiplies, can compute an upsampled DFT in just
153 % a small region.
154 % usfac      Upsampling factor (default usfac = 1)
155 % [nor, noc] Number of pixels in the output upsampled DFT, in
156 %            units of upsampled pixels (default = size(in))
157 % roff, coff Row and column offsets, allow to shift the output array to
158 %            a region of interest on the DFT (default = 0)
159 % Recieves DC in upper left corner, image center must be in (1,1)
160 % Manuel Guizar - Dec 13, 2007
161 % Modified from dftus, by J.R. Fienup 7/31/06
162
163 % This code is intended to provide the same result as if the following
164 % operations were performed
165 % - Embed the array in in an array that is usfac times larger in each
166 %   dimension. ifftshift to bring the center of the image to (1,1).
167 % - Take the FFT of the larger array
168 % - Extract an [nor, noc] region of the result. Starting with the
169 %   [roff+1 coff+1] element.
170
171 % It achieves this result by computing the DFT in the output array without
172 % the need to zeropad. Much faster and memory efficient than the
173 % zero-padded FFT approach if [nor noc] are much smaller than [nr*usfac nc*usfac]
174
175 [nr, nc]=size(in);
176 % Set defaults
177 if exist('roff', 'var')~=1, roff=0; end
178 if exist('coff', 'var')~=1, coff=0; end
179 if exist('usfac', 'var')~=1, usfac=1; end
180 if exist('noc', 'var')~=1, noc=nc; end
181 if exist('nor', 'var')~=1, nor=nr; end
182 % Compute kernels and obtain DFT by matrix products
183 kernc=exp((-1i*2*pi/(nc*usfac))*( ifftshift(0:nc-1).' - floor(nc/2) )*( (0:noc-1) - coff ));
184 kernr=exp((-1i*2*pi/(nr*usfac))*( (0:nor-1).' - roff )*( ifftshift([0:nr-1]) - floor(nr/2) ));
185 out=kernr*in*kernc;
186 return
187
188
189 function [ imFTout ] = FTpad(imFT, outsize)
190 % imFTout = FTpad(imFT, outsize)
191 % Pads or crops the Fourier transform to the desired ouput size. Taking
192 % care that the zero frequency is put in the correct place for the output
193 % for subsequent FT or IFT. Can be used for Fourier transform based
194 % interpolation, i.e. dirichlet kernel interpolation.
195 %
196 % Inputs
197 % imFT      - Input complex array with DC in [1,1]
198 % outsize   - Output size of array [ny nx]
199 %
200 % Outputs

```

```

201 % imout - Output complex image with DC in [1,1]
202 % Manuel Guizar - 2014.06.02
203
204 if ~ismatrix(imFT)
205     error('Maximum number of array dimensions is 2')
206 end
207 Nout = outsize;
208 Nin = size(imFT);
209 imFT = fftshift(imFT);
210 center = floor(size(imFT)/2)+1;
211
212 imFTout = zeros(outsize);
213 centerout = floor(size(imFTout)/2)+1;
214
215 % imout(centerout(1)+[1:Nin(1)]-center(1),centerout(2)+[1:Nin(2)]-center(2)) ...
216 % = imFT;
217 cenout_cen = centerout - center;
218 imFTout(max(cenout_cen(1)+1,1):min(cenout_cen(1)+Nin(1),Nout(1)),max(cenout_cen(2)+1,1):min(
219     cenout_cen(2)+Nin(2),Nout(2))) ...
220     = imFT(max(-cenout_cen(1)+1,1):min(-cenout_cen(1)+Nout(1),Nin(1)),max(-cenout_cen(2)+1,1):min(
221         (-cenout_cen(2)+Nout(2),Nin(2))));
222
223 imFTout = ifftshift(imFTout)*Nout(1)*Nout(2)/(Nin(1)*Nin(2));
224 return

```

```

1 function [data1, metadata] = readXenicsData(filename)
2
3 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
4 % readTrexXenicsData
5 % Read binary pixel data saved by the TrexXenicsGui program
6 %
7 %Input:
8 % filename to the data as a string.
9 %
10 %Outputs:
11 % data: Returned as a data cube of, [M,N,Frames]
12 %
13 % metadata: This is a struct detailing additoinal information about the
14 % collect and will contain the following:
15 %
16 % version: Data version as saved by TrexXenicsGui
17 % x: X location of the upper left corner with regard
18 % to the physical sensor. if full frame: 0
19 % y: Y location of the upper left corner with regard
20 % to the physical sensor. if full frame: 0
21 % rows: The number of rows per image frame
22 % cols: The number of columns per image frame
23 % frames: The number of Frames
24 % bigframecount: Number of 640x512 frames to store a full
25 % dataset. Used for internal GUI purposes.
26 % integrationTime: The integration time in microseconds
27 % bufferMode: 0 = fullframe
28 % 2 = Region of Interest Mode
29 % frameSize: Rows * Cols
30 %
31 % [data metadata] = readCameraData(filename)
32 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
33
34 % constants
35 nHeader = 18;
36 dataFormat = 'uint8';
37
38 % open the file
39 fid = fopen(filename, 'r', 'b');
40 if fid <= 0
41     error('file open error');
42 end
43
44 try
45     %% Read the data
46     data = fread(fid, inf, [dataFormat, '=>single']);
47
48     metadata.version = uint32(typecast(uint8(data(1:2)), 'uint16'));
49     metadata.x = uint32(typecast(uint8(data(3:4)), 'uint16'));
50     metadata.y = uint32(typecast(uint8(data(5:6)), 'uint16'));
51     metadata.rows = uint32(typecast(uint8(data(7:8)), 'uint16'));
52     metadata.cols = uint32(typecast(uint8(data(9:10)), 'uint16'));
53     metadata.frames = uint32(typecast(uint8(data(11:12)), 'uint16'));
54     metadata.bigframecount = uint32(typecast(uint8(data(13:14)), 'uint16'));
55     metadata.integrationTime = uint32(typecast(uint8(data(15:16)), 'uint16'));
56     metadata.bufferMode = uint32(typecast(uint8(data(17:18)), 'uint16'));
57
58     metadata.frameSize = metadata.rows * metadata.cols;
59     %% Format
60     %Pull necessary data and format into frame arrays
61     data = reshape(data((1+nHeader):(prod(metadata.frameSize)*metadata.frames+nHeader)), [fliplr(
        metadata.frameSize), metadata.frames]);

```

```

62     %Reshape into a data cube
63     data = reshape(data,metadata.cols,metadata.rows,[]);
64
65     %% START MODIFICATION:I had to have it rotate one frame at a time (rot90 can only rotate a
        matrix not a data cube), while still performing the flip
66     for i = 1:metadata.frames
67         data1(:, :, i) = fliplr(rot90(data(:, :, i), -1));
68     end
69     % END MODIFICATION
70
71 catch ME
72     try
73         fclose(fid);
74     catch ME2
75         getReport(ME2)
76     end
77     rethrow(ME)
78 end
79
80 fclose(fid);

```