PHASE GRADIENT AVERAGING FOR HOLOGRAPHIC APERTURE LIDAR IN THE PRESENCE OF TURBULENCE

by

Christopher Ross Blaszczyk

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

July 2017

DEDICATION

I would like to dedicate this to several people, the first being my dear wife Lyndsey Duerden. Without her constant support and encouragement, I would not have finished this thesis. She is my rock and my world. So, thank you love for all your support. Second being my two children, Sterling and Freyja, one being the reason to come home every night while this thesis was being written and the other to get it done before you showed up. You give my life so much meaning and make me want to be a better man, engineer, and father.

Next, I would like to thank my parents Chris and Linda Blaszczyk, whose years of care and concern should never be overlooked. Thank you for always being there when I needed you to encourage and to push me on when I needed it. For seeing worth in me when others would have cast me aside as not worth the trouble. My thanks could never fully be expressed for your willingness to fight for me.

To my in-laws, Sterling and Angie Duerden, for supporting my family during the time that I have been studying here at MSU. Being a source of strength and encouragement that always made my day brighter.

ACKNOWLEDGEMENTS

I would first like to acknowledge the whole team at Spectrum lab MSU who have help train and instruct me in my two years at MSU. I would like to thank first Dr. Zeb Barber for being there to answer all my many questions and assisting in my understanding of the topics I learned here at MSU. Thanks also to Diane Harn for taking time to help fix all the issues that seem to arise with being a student employed by the school.

I would also like to acknowledge Prof. William Randall Babbitt for being on my thesis committee and for his advice and counsel during my time here at MSU. I would also like to acknowledge and thank Prof. Joseph Shaw for being my academic advisor and bringing me into the program.

I would like to acknowledge Blackmore Sensor and Analytics who sub-contracted this work to Spectrum Lab MSU, giving me the opportunity to work on this project. Thanks especially goes out to Stephen Crouch who mentored me over one summer and allowed me to work at Blackmore as an intern. The whole crew at Blackmore was amazing.

TABLE OF CONTENTS

TABLE OF CONTENTS CONTINUED

TABLE OF CONTENTS CONTINUED

LIST OF TABLES

LIST OF FIGURES

Figure Page

LIST OF FIGURES CONTINUED

Figure Page

LIST OF FIGURES CONTINUED

Figure Page

ABSTRACT

The resolution of an image is dictated by the size and quality of the imaging system. The imaging system has a physical limit to the resolution dictated by the diffraction limited resolution. This limit can be improved by making the aperture larger on the imaging system. The increase in the physical aperture size can only be practical to a certain extent. However, to get beyond these physical constraints it is possible to use synthetic aperture methods to allow for the aperture to appear to be increased. Synthetic apertures are created by adding apertures coherently together to create a larger aperture that increases the diffraction limited resolution. To sum the aperture coherently the phase information needs to be available.

One way to have access to the phase information is to capture the image as hologram. These holograms are captured by using a coherent light source with a reference beam to create an interference pattern that contains the phase information of the target. Holographic apertures can be used in a synthetic aperture method called Holographic Aperture Lidar (HAL). A problem that can arise while capturing images is turbulence in the atmosphere. Turbulence is a change in the index of refraction caused by a change in the temperature and pressure of the atmosphere. This causes the phase of the light to distort dynamically as it propagates making HAL imaging difficult. This thesis will cover a method to restore the original phase of the signal that has passed through turbulence so that it can be used in digital holography and HAL. This method uses averaging of the phase gradient to remove the dynamic turbulence and keep the phase information of the static target. The improvements observed in actual experiments were small, but the basics of the method worked, and the reason for only small improvement are discussed.

MOTIVATION

Long Range Imaging with High Resolution

A difficult problem that usually requires a very expensive, large telescope is the imaging of asteroids or meteors that are passing through our solar system. Since these asteroids or meteors are very far away they appear very small when viewed optically. It would be beneficial if these asteroids or meteors could be viewed with smaller telescopes that are easier to manufacture. A higher-resolution image would need be needed to see a meteor break up on entry, for example, by looking at various features of the meteor. A smaller telescope though does not have the resolution to capture the features of the asteroids due to having poor diffraction resolution limits that arises from the small entrance pupil. These are the two problems that are addressed in this thesis. These problems do not only exist for objects in orbit or further away in space, but also terrestrially there is a need for this type of imaging. For example, the same problem arises if we want to capture small details for mapping purposes from a plane. This has been used in agriculture when looking for features from water to weeds [1]. It is the small size of the features to be captured, that are important. Other examples where this can be needed is imaging from air to ground, ground to air, and air to air. Basically, anywhere the details that need to be examined are small and closely grouped together.

Diffraction Limit of Imaging

The ability to tell two points or objects apart optically whether they are stars, or two people walking beside each other, is restricted by diffraction. Diffraction is how light propagates from the forward cone of an illuminated object. This makes diffraction a special case of scattering usually only described at small forward angles. Diffraction limits the resolution at which an object can be imaged.

The diffraction limit of imaging resolution is defined by the wavelength of light being observed, the size of the entrance pupil of the optical system, and the distance to the object (Figure 1).

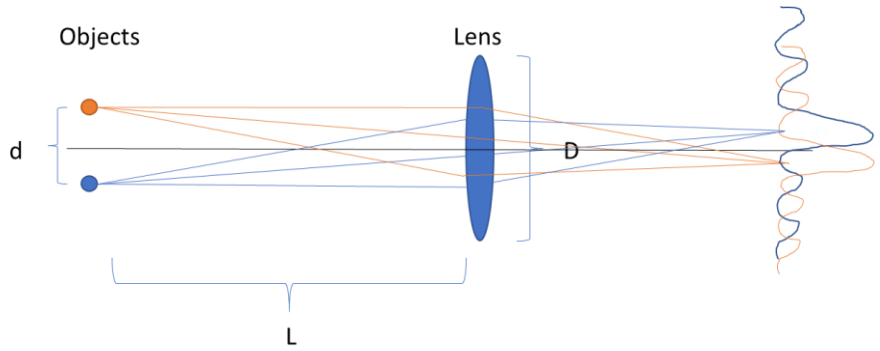

Figure 1 Example of diffraction-limited resolution in practical terms

Since most apertures are circular in shape, we often use the diffraction limit equation that corresponds to that setup, given as

$$d = 1.22\lambda \frac{L}{D}, \tag{1.1}$$

where d is the diffraction- limited spatial resolution, L is the distance to the object, λ is the wavelength of the light source, and D is the entrance pupil diameter. This is the diffraction-limited resolution in terms of the Rayleigh criterion, describes the smallest distance in object space two objects can be from each other and still be resolvable as separate when using a spherical aperture. To illustrate diffraction limited resolution, several examples, at several different scales, follow. All these examples assume visible light at a wavelength of 550 nm.

Magnifying Glass

An everyday magnifying glass, which only has one lens element, can be used to demonstrate the diffraction-limited resolution principle for a small object at a small distance. For example, if the distance is 100 mm to the object, the lens has a diameter of 25 mm, and the illumination light is at 550 nm wavelength, then substituting into Equation (1.1) yields 2.6840×10^{-6} m as the minimum distance that can be differentiated between two points on the object, for a diffraction-limited imaging system (this distance is in the plane transverse to the propagation or viewing axis). Note that this assumes that the magnification is sufficient for the eye or the detector to resolve this level of resolution in the image plane.

This example demonstrates that close distances makes diffraction limited resolution better. However, this example does little for our motivation of wanting to take long range images with a small field of view.

Cell Phone Camera vs 35mm Camera

To show the difference aperture size makes to diffraction limited resolution, this example considers two lens systems that are used for the same purpose. A simple example, with which most people will be familiar, is that of a 35-mm camera lens system and a cell phone camera lens system. Assume the cellphone camera has a 2.5-mm diameter lens. For the 35-mm camera lens, we assume a Canon 80-mm (focal length) lens. EF f/1.8. To calculate the entrance pupil size, we divide the focal length by the f-number. This leads to an entrance pupil size of 44 mm, which is quiet large being almost the size of the physical lens aperture. Further, if we assume the subject is 2 m away from the camera, Equation 1.1 tells us that the 35-mm lens has a diffraction-limited resolution at the object plane of 3.05×10^{-5}m, and the cell phone has resolution of 5.368×10^{-4}m.

This example emphasizes that large lens apertures are better than small ones for diffraction-limited resolution. If we can use a large lens to make our image, it is generally better, though in common photography cases diffraction-limited resolution can be less important because of aberrations, which increase rapidly with the field of view, and because the sensor sometimes has pixels that are larger than the diffraction limit.

To obtain the highest image resolution, the pixels in the sensor should be sized to match that of the diffraction limit, including the magnification. This means that if we want a field of view of 1.58 m ×1.58 m at a 2-m object distance, then for the 35-mm camera we are going to need $1.58\text{m} \div (3.05 \times 10^{-5}\text{m}) = 5.18 \times 10^{4}$ pixels for each side of the sensor and if the camera had a square sensor, then that would be a 2.68

gigapixel sensor. This is an unrealistic sensor size even today. This means that, for common photography, it is often unimportant to try to reach the diffraction limit.

Aberrations are the other reason why the diffraction limit is rarely a limit in common photography problems. Aberrations are caused by the physical nature of the lens having actual thickness with surfaces that have a spherical shape. The ability to make a large, perfect camera lens with no aberration for all wavelengths would be prohibitively expensive for many applications. This leads to the optical waves bending not in perfect spherical waves after leaving the lens, which leads to the image becoming warped in a manner described by the type of aberration. Many of the common aberrations have a dependence on the distance of the ray from the lens center. So, to help reduce aberrations for everyday photography, the aperture stop can be reduced so that the rays are kept to small diameters, or a more complicated lens design can be used that reduces the aberrations. This explains why there are so many lens designs that try to reduce the aberrations for different applications.

Telescope

The most extreme example of long range imaging is that of using a telescope to look at extraterrestrial objects. The large distance to the stars means that measuring diffraction in terms of distances is not helpful. The best way to express resolution in this type of situation is with angular resolution, expressed in terms of the wavelength and the aperture diameter, as follows:

$$\theta = 1.22\frac{\lambda}{D}. \quad (1.2)$$

The difference in aperture diameter becomes very apparent in terms of scale. For example, look at an entry-level Orion telescope vs. the South African Large telescope (SALT) to see how aperture diameter can affect the angular resolution (see Figure 2). The Orion has a 75-mm aperture while SALT has an 11-m aperture. According to Equation 1.2, the Orion has an angular resolution of $8.94\text{x}10^{-6}$ radians, while the SALT has an angular limit of $6.1\text{x}10^{-8}$ radians.

To give a context for these diffraction angles, we can look at what the distance resolution would be at a kilometer. The Orion telescope would have a diffraction-limited resolution of 0.0089 m, while the SALT would have a resolution of 6.1×10^{-5} m. The difference is like being able to barely read the title on a newspaper and being able to easily read the article underneath it.

Figure 2 The Salt [2] and the Orion telescope [3] used to illustrate how the difference in aperture size is the main reason that their resolutions are so different.

Practical Considerations

It appears from these examples and formulas that it would be easy to get high resolution images, but this is not the case in real life. The fact that the lenses are made of glass with dispersion and have spherical shape adds aberrations, which limits the resolution. Mirrors, like those in telescopes, have aberration due to their imperfect shape. Aberrations are just a natural factor when working with real materials and fabrication methods. However, aberrations can be reduced with complex lens design for instance, by adding more lens elements or by using an iris in photographic systems where diffraction-limited resolution may not be a concern. If you want to build a system with a large aperture and you cannot reduce the aperture stop it is sometimes possible to correct aberrations through post-processing methods [4].

Another issue when imaging in the air, particularly over long distance, is turbulence. Turbulence can be easily visualized by thinking of heat coming off a road on a summer day (see Figure 3). The wavy air that you see as the heat dissipates off the road is turbulence. In more scientific terms, turbulence is fluctuations in pressure and temperature of the air, which cause the index of refraction to fluctuate. This leads to fluctuations in the phase information of light that passes through the turbulence and produces similar results as aberrations. This constantly changing phase information can greatly hinder imaging, including aperture formation in synthetic aperture imaging.

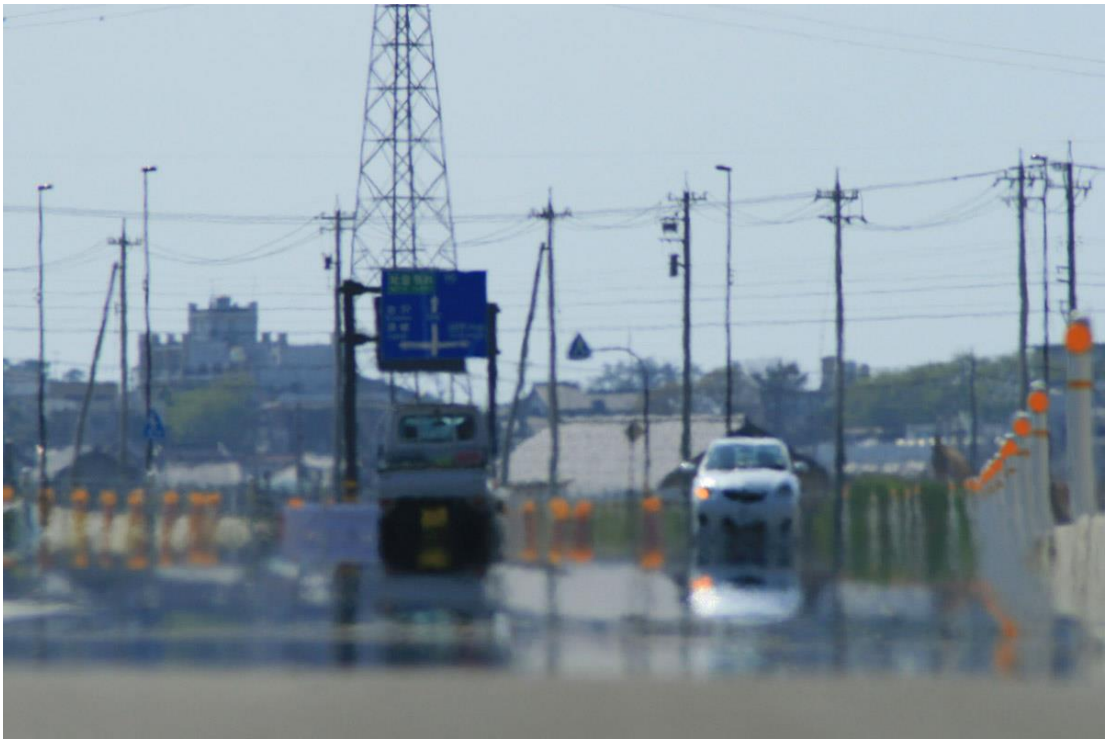

Figure 3 An example of turbulence in real life [5]. This is an image of the heat waves coming off the road during the summer causing the index of refraction in the air to change, which cause the wobbling nature of the image.

Another constraint, which has much less to do with optical quality of an imaging system, is that of cost and weight. The ability to create an optical system with very high-quality optics is expensive. A portion of this expense is the cost of materials, but the main cost is the skill to make these lenses. Also, when custom lenses are made they need to be tested. The weight is a consideration since the glass from a lens system can make the system very heavy, which becomes an issue when the imaging system is going to be used on a vehicle. This is especially true for a system on aircraft or satellites where weight is at a premium either due to size and/or fuel constraints.

Holographic Aperture Lidar

Holographic Aperture Lidar (HAL) is a method of synthetic aperture imaging that allows for greater resolution in holographic images [6, 7]. With this method, relative motion between imaging system and target is used to collect multiple perspectives of an object to increase image resolution. It is a subset of synthetic aperture methods that were first used in the 1960's for radar [8], and has seen continued development as lasers have become easier to make and more common in everyday use. The use of synthetic aperture is a way of dealing with weak signals. Another way to deal with weak signals is to use sparse imaging, in which you capture a target signal with not just one static aperture but many smaller static apertures [9]. The multiple apertures summed together allows for a larger synthetic aperture, which allows for the diffraction-limited resolution to become smaller. Synthetic aperture imaging can be done by rotating the target or moving the imaging system across the target.

A problem that can arise when capturing images at large distance is that the air in the atmosphere is undergoing changes in its index of refraction, which is called turbulence [10]. Turbulence hinders the use of synthetic aperture as the turbulence decorrelates the phase across real apertures. Methods currently used to adjust for turbulence are computationally very heavy [11]. The goal of this thesis that was to investigate a different turbulence compensation methods that could be relatively computationally light. The proposed method takes the average of phase gradients in the apertures to suppress the turbulence, which should be dynamic while the image should not be. We demonstrated a new processing technique for correcting turbulence in

coherent imaging and applied it to real-world turbulent images and synthetic aperture imaging.

Active Illuminations

For some imaging applications, including lidar and digital holography, active illumination is used or required. Active illumination means that one uses a light source as part of the imaging system to illuminate the object, versus using light coming from an outside source like the sun or self-illumination. One advantage of active illumination is being able to image at night, or when illumination is weak. Sometimes the amount of light available from natural illumination is not enough to allow for a high level of detail, particularly at long distances. By using active illumination this is no longer an issue, as we can control the amount of light that is being used to illuminate the object (assuming enough source light can still reach the distant object).

Incoherent Imaging Versus Coherent Imaging

When using active illumination there are two choices: incoherent illumination and coherent illumination. Incoherent illumination sources are forms of light where the phase of the light is random in time. A common example of an incoherent light source would be light bulbs, which are great for illuminating normal photography, when we do not care about the phase information of the captured image. The signal captured in incoherent imaging is intensity only. Coherent light sources are ones where the phase of the light is consistent when emitted. An example of this is a laser.

Collimated Laser Illumination

One source of active illumination is a collimated laser. Collimation is when the laser beam size is not expanding or contracting, other than the inevitable diffraction-induced spreading. A collimated laser is close to being a plane wave, which does not diffract quickly, letting one illuminate a small area that is far away. This allows for a lot of light to be concentrated onto a small spot. Using a laser has another nice property of being coherent, which means the frequency and phase of the laser is contained to a relatively small bandwidth.

Speckle

Coherent images have a unique property called speckle. Speckle is the result of random phase interference of the light on the recording medium from different parts of the object. This interaction of phase does not produce the nicest looking images, since speckle creates intensity nulls and high intensity spots. This is through random constructive and destructive interference of many phase terms from many different scattering points off the object.

We do not see this speckle in regular photography, is because light from the sun or a light bulb is not coherent. This means that the phase of the emitted light is random, so the light from one location on the source cannot interfere with light from a different location. This makes speckle a unique property of coherent imaging. What is the advantage of speckle if it does not produce a high-quality image? The use of speckle relies on being able to make use of phase information during processing, and is a main reason to use coherent light.

Since there are no sensors that actually measure the phase of light directly, we need to find a way to extract this phase information. This is done by using the ability of coherent light to interfere with itself. This can be done using a beamsplitter to use some of the light to illuminate the object and the rest to act as a reference to interfere with the light from the object. This reference beam is called a local oscillator (LO). The speckle is only useable for obtaining the phase information if the LO is present and the phase information can be extracted from the image by using image processing, generally using Fourier transforms. The phase coherent speckle contains detailed information about the object, including statistical information [12].

Another useful advantage of coherent active illumination is that, since the LO is not sent to the target and is relatively close to the sensor, it is able to increase the overall intensity of the collected image, which produces an effect called coherent gain due to coherent interference. This has advantages especially in digital holography, where the overall intensity of the return signal from a target at a great distance is very small.

With the phase information in a coherent image, it is also possible to correct for aberrations from the lens system. This can be done by including phase terms to the image by means of digital post processing [12].

Synthetic Aperture Imaging

With the phase information now being available for use in coherent imaging, it is possible to stitch together multiple apertures coherently to create a larger “synthetic” aperture. The apertures can be collected by moving the target, the receiver, the beam, or a combination of these elements. This was first discussed as a method for increasing

resolution by Brown and Porcello [8]. They developed the method to be used with radar, so it was initially called Synthetic Aperture Radar (SAR). This early method made use of an antenna to collect radio signals versus optically collected light.

The production of high quality lasers led to the development of Lidar, light detection and ranging (lidar), which is similar to radar. The same principles that allowed SAR to work were also found to work for lidar [13]. The name given to this optical technique is called Synthetic Aperture Lidar (SAL) for this reason. Using synthetic apertures allows for increased resolution by increasing the effective aperture size. (see Eq. 1.1)

There are several different types of SAL, depending on what part and how the system is being moved. There is the strip map method where the receiver and transmit receiver beam are moving along together across an object, capturing strips that can then be stitched together. Spotlight is where the beam is steered to stay on the target while the receiver moves across the target path. Then, there is inverse spotlight mode, where the beam is steered onto the target and the target moves or is rotated to create a relative motion. For this thesis, I used the inverse spotlight method with rotation for data capture. The receiver was located on a lab bench while the target was located on a rotation stage.

The process of making the holographic aperture lidar (HAL) image is to offset the apertures individually from each other properly and to simply add them together coherently to form a larger-aperture field. This forms what is known as a mosaic of apertures. This mosaic of apertures is now your synthetic aperture, which can be used to create a higher-resolution image. However, the synthetic aperture only increases the

image resolution in the direction(s) that the mosaic is expanded. So, for simple linear motion image resolution is only improved in one direction.

Turbulence

If turbulence is present, it is difficult to satisfy the requirement for the phase to be coherent across the apertures being used in the synthetic aperture. The turbulence adds phase information that is random in time, leading to neighboring apertures having different phase contributions from the turbulence. This prevents the correlation of the apertures to be correctly offset from each other and distorts the object information differently in the different apertures. This prevents synthetic aperture techniques from being effective. This also means that there are limits to the diffraction limited resolution that can be achieved.

HOLOGRAPHY AND FOURIER OPTICS

Holography

A hologram is a phase sensitive recording of the spatial information from an object of interest. A hologram is captured using the coherent light of a laser and interference with an LO being present on the recording medium to serve as a phase reference (see Figure 4). To view holograms requires reconstructing them with optical wavefronts that are like the reference beam. This is because what is captured is the interference pattern of the target and the LO. When a light source containing a similar wavefront that was used for the capture is used as the source for the reconstruction, a virtual image is created. This virtual image seems to the observer to be in the same location as the object was originally (see Figure 4).

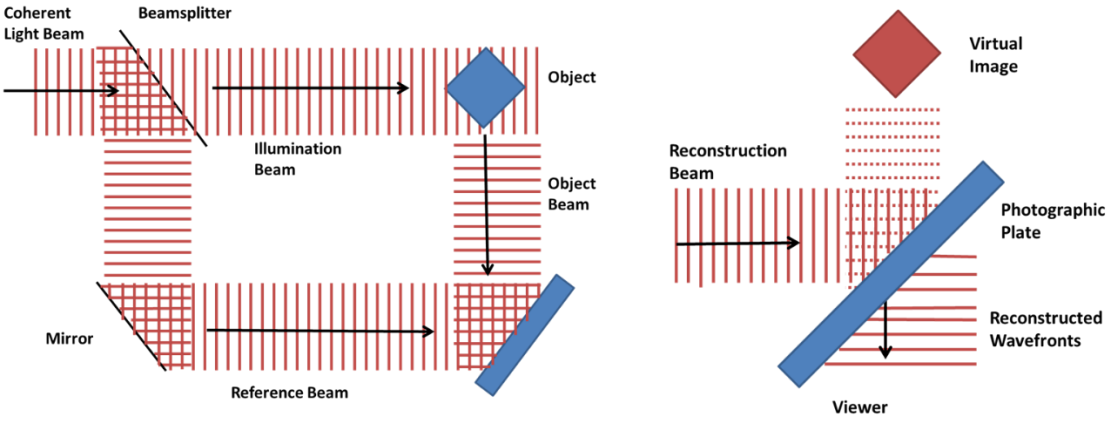

Figure 4 The setup for capture of holograms and how to retrieve a hologram. A digital hologram is one recorded with a digital sensor instead of a photographic plate.

Digital holography and Computer-Generated Hologram

With the introduction of computers and their ability to do numerical computations it is possible for computers to create the interference patterns associated with holograms, if given the parameters about the light source to be used for projection, the target parameters, and at what angle capture occurs. These computer-generated patterns can be printed onto a medium and be projected like any other hologram. Another use of computers is that the holographic interference pattern can be captured onto a digital recording medium then digitally reconstructed to recreate the object. This holographic projection makes use of Fourier transforms to propagate the interference pattern to different planes containing the image field.

Theory of Fourier Optics

Fourier optics is a form of wave optics in which the light is treated as a wave, generally using complex representation, and Fourier transforms are used to represent the propagation and diffraction process. This complex representation allows the light to be described not only in terms of amplitude, but as phase as well. This phase information allows for different types of optical wavefronts to be described. Light as a wave can be described in a number of ways such as plane waves, spherical waves, and Gaussian beams.

Plane Waves

Planes waves are waves that have flat wavefronts that technically extend to infinity in all directions. This means that if one were to represent the wavefronts moving

along the z axis as the wave propagates in space, the wavefronts would look like straight lines. To mathematically represent this, we call the amplitude of the wave A, with the phase information being contained in a complex exponential as

$$Ae^{i\vec{k}\vec{r}}, \tag{2.1}$$

where the $\vec{k}$ is the wave vector, given by 2π over the wavelength λ, multiplied by the unit vector of propagation. The vector $\vec{k}$ represents the propagation direction of the phase. Often this is predominately in one direction, which is defined as the z direction. This leads to an approximate form where $k_z = k$ and the corresponding phase term is pulled out in front of the following expression for the complex field amplitude U (EQ. 2.2), using the definition of k in Eq. 2.3,

$$U(r) = Aexp(ikz)exp(i(k_x x + k_y y)) \tag{2.2}$$

$$k = \frac{2\pi}{\lambda} \tag{2.3}$$

This special form shows the angle based nature of the wave as k_x and k_y represent cosines of the propagation direction away from the z axis.

Spherical Waves

Spherical waves have wavefronts, either emanating out from a central point or heading towards a focal point, that look like an expanding, or contracting circle when viewed as cross-section on the axis of propagation. These types of waves are found from

point sources or approximated by light diverging after passing through a pinhole. Also, when scattering occurs off an object the emitted wave can be treated as a set of point sources that release spherical waves. Spherical waves are described mathematically in spherical coordinates with r being the radial distance from the center of the coordinate frame used in analysis, according to

$$U(r) = \frac{A_0}{r}\exp(-ikr). \tag{2.4}$$

Spherical waves can be approximated along the z-axis as a paraxial spherical wave, which in Cartesian coordinates is

$$U(r) = \frac{A_0}{z}\exp(\mathrm{ikz})\exp(-ik/2z(x^2 + y^2). \tag{2.5}$$

Another difference from plane waves is that the spherical wave amplitude is also dependent on how far from the scattering center you are looking at the wavefront.

Gaussian Beam

Modeling how light propagates from a laser is most often done using a Gaussian beam approximation. This treats the light beam as having a Gaussian intensity distribution along the wavefront. A Gaussian beam has a predictable shape based on the size and location of the beam's waist. The beam waist, W_o, is the radius at the point at which the beam has the smallest radius along its travel path. Once this point is defined ($z = 0$) and the Rayleigh range (z_0) and the radius of the beam waist are known

it is possible to describe a Gaussian wave anywhere along its beam path (see Eq. 2.6-2.9). The wavefront radius of the Gaussian beam, $R(z)$, is dependent on where the beam is measured from when compared to the Rayleigh range (Eq. 2.8). The Gouy phase $\xi(z)$ describes the phase velocity of light near the focus [14]. The complex wave amplitude for a Gaussian beam and these various related parameters can be written as

$$U(r) = A_0 \frac{W_o}{W(z)} \exp\left[-\frac{\rho^2}{W^2(z)}\right] exp\left[-ikz - ik\frac{\rho^2}{2R(z)} + i\xi(z)\right], \tag{2.6}$$

$$W(z) = W_o\sqrt{1 + \left(\frac{z}{z_o}\right)^2}, \tag{2.7}$$

$$R(z) = z\left[1 + \left(\frac{z}{z_o}\right)^2\right], \tag{2.8}$$

$$\xi(z) = \tan^{-1}\frac{z}{z_o}, \text{ and} \tag{2.9}$$

$$W_o = \sqrt{\frac{\lambda z_o}{\pi}}, \tag{2.8}$$

Fourier Transforms

A good understanding of Fourier transforms is necessary to use Fourier optics and to make use of holograms. The Fourier transform relates the spatial domain to the frequency domain. There are several functions and their transform that are most used in Fourier optics, including the calculation of light diffracted from apertures. These transforms are listed below for reference (see Table 1).

Table 1 List of functions and their Fourier transforms [15]. These functions are common examples from Fourier optics.

Function	Transform
$\exp[-\pi(a^2x^2+b^2y^2)]$	$\frac{1}{\lvert ab\rvert}\exp\left[-\pi\left(\frac{{f_x}^2}{a^2}+\frac{{f_y}^2}{b^2}\right)\right]$
$\mathrm{rect}(ax)\mathrm{rect}(by)$	$\frac{1}{\lvert ab\rvert}\mathrm{sinc}\left(\frac{f_x}{a}\right)\mathrm{sinc}\left(\frac{f_y}{b}\right)$
$\Lambda(ax)\Lambda(by)$ Triangle Function	$\frac{1}{ab}\mathrm{sinc}^2\left(\frac{\mathrm{f_x}}{\mathrm{a}}\right)\mathrm{sinc}^2\left(\frac{f_y}{b}\right)$
$\delta(ax,by)$	$\frac{1}{\lvert ab\rvert}$
$\exp[i\pi(ax+by)]$	$\delta\left(f_x-\frac{a}{2},f_y-\frac{b}{2}\right)$
$\mathrm{sgn}(ax)\mathrm{sgn}(by)$	$\frac{ab}{\lvert ab\rvert}\frac{1}{i\pi f_x}\frac{1}{i\pi f_y}$
$\mathrm{comb}(ax)\mathrm{comb}(by)$	$\frac{1}{\lvert ab\rvert}\mathrm{comb}\left(\frac{f_x}{a}\right)\mathrm{comb}\left(\frac{f_y}{b}\right)$
$\exp[i\pi(a^2x^2+b^2y^2)]$	$\frac{i}{\lvert ab\rvert}\exp\left[-i\pi\left(\frac{{f_x}^2}{a^2}+\frac{{f_y}^2}{b^2}\right)\right]$
$\exp[-(a\lvert x\rvert+b\lvert y\rvert)]$	$\frac{1}{ab}\frac{2}{1+(2\pi f_x/a)^2}\frac{2}{1+\left(2\pi f_y/b\right)^2}$

In addition to specific Fourier transform pairs, there are theorems relating Fourier transforms. A theorem that is used often in Fourier Optics is the Fourier shift theorem. This theorem says that a shift in the space domain introduces a linear phase shift in the

frequency domain and vice-versa. This is often used to realign images after transformations. For a spatial function g with Fourier transform G (Eq. 2.9), the Fourier transform of a shifted version of g can be expressed according to Eq. 2.10

$$F\{g(x,y)\} = G(f_x, f_y) \tag{2.9}$$

$$F\{g(x-a, y-b)\} = G(f_x, f_y)\exp\left[-i2\pi(f_x a + f_y b)\right] \tag{2.10}$$

Continuous versus Discrete Fourier Transforms

A continuous Fourier transform goes across its domain with infinite resolution and can therefore go to infinity in both the space and frequency domains. Conversely, a discrete Fourier transform is one where the function or object is sampled in discrete steps in units over which the transform is performed to convert the discrete function into the frequency domain. This is in contrast to continuous Fourier transforms in which the function or object is never sampled and is analytically solved. Since I made use of a computer to do the Fourier transforms for the use of data analysis, I am using discrete Fourier transforms. I am also using discrete Fourier transforms because I used a digital camera that has pixels, which act as discrete steps in the image plane. This means that the images I capture had a maximum frequency that can be represented in each direction. I show below how to obtain different units of scale in the Fourier domain for the x axis (the same process can be used in the y axis).

To find the maximum frequency that can be represented, one takes the inverse of the size of the discrete unit (i.e. pixel size) used to record the image (Eq. 2.11). For the work in this thesis, this was the size of my camera pixels. The size of the discrete units in

the Fourier domain is the inverse of the total length (or width) of the image (Eq. 2.12), in my case the length (or width) of the sensor.

$$f_x = \frac{1}{dx} \tag{2.11}$$

$$df_x = \frac{1}{L} \tag{2.12}$$

If one wants the scale to be in angles measured in radians, so one can look at the angles and relate them to a real-world experiment, then the maximum angle is the maximum frequency times the wavelength of coherent light that is used to illuminate the scene

$$\theta_x = f_x \lambda. \tag{2.13}$$

The final scaling that is useful to show here is to put these quantities in terms of distance from the central axis, Δ_x, at a specific propagation distance z. This makes use of the angle that was previously derived and the distance to the object. If the angle is small, it can be simply multiplied with the distance to the object as

$$\Delta_x(z) = \theta_x z. \tag{2.14}$$

Nyquist Sampling Criterion

A question that arises when using discrete Fourier transforms is, what sampling criteria needs to be met to model the signal well. This is called the Nyquist sampling criterion. It states that the sampling frequency (the maximum frequency span represented

in the Fourier domain) needs to be two times the maximum frequency that you want to observe (see Eq. 2.15).

$$f_c = 2f_{max} \tag{2.15}$$

If this criterion is not met, then certain distortions can occur, including aliasing, rippling, and smearing in the Fourier plane, and virtual periodic replication in the image plane [16]. Aliasing is when the signal goes back around to the other side of the image and continues around the axis. This is obviously not desirable as it adds unwanted signal to the data or image being processed. The rippling and smearing in the Fourier plane acts like the aberration in the image plane. The virtual replication can appear as a non-real object that would interferes with the data collected.

Fraunhofer Propagation

As part of the study of propagation, the setup of the problem involves two planes with one having axes labeled x_1 and y_1 from which the signal originates, and a second plane separated from first by the distance of propagation z with axes labeled x_2 and y_2 (Figure 5).

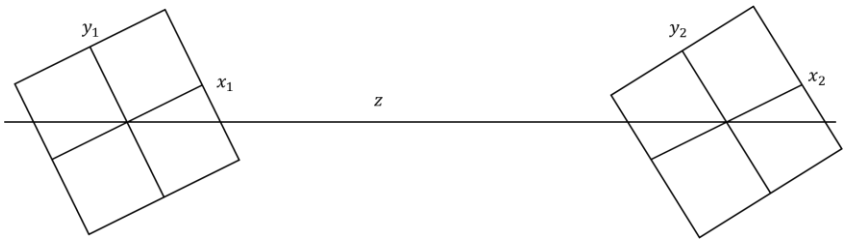

Figure 5 Axis of the planes of propagation

To capture diffraction and wave propagation in the far-field the Fraunhofer approximation is used. This approximation requires a large distance from diffracting object to the observation plane as

$$z \gg \frac{k(x^2 + y^2)_{max}}{2}. \tag{2.16}$$

If this criterion is met, then the Fraunhofer approximation can be used. This approximation describes the complex wave amplitude of a diffracted wave as

$$U(f_x, f_y) = \frac{e^{ikz} e^{i\frac{k}{2z}(x_2{}^2 + y_2{}^2)}}{i\lambda z} \iint_{-\infty}^{\infty} U(x_1, y_1) exp\left[-i\frac{2\pi}{\lambda z}(x_2 x_1 + y_2 y_1)\right] dx_1 dy_1 \tag{2.17}$$

The term in front of the integral contains factors independent of x_1 and y_1, which are multiplied to the field at all points. These factors are often dropped when speaking qualitatively about propagation, but are used in calculations. This occurs because when one views images in intensity, these terms go to one. The $U(x, y)$ is our initial optical field before the propagation. This includes any aperture that it is passing through. The exponential in the integral is essentially a Fourier transform kernel, which means Fraunhofer propagation is essentially a Fourier transform.

To examine these effects, I model the diffraction from a square aperture of 0.05 cm by 0.05 cm that has been illuminated by a monochromatic plane wave. The distance to the observation plane is 10 m, which meets the requirements of the Fraunhofer

diffraction approximation. One notices in Figure 6 the sinc side lobes, which is the Fourier transform of a square aperture.

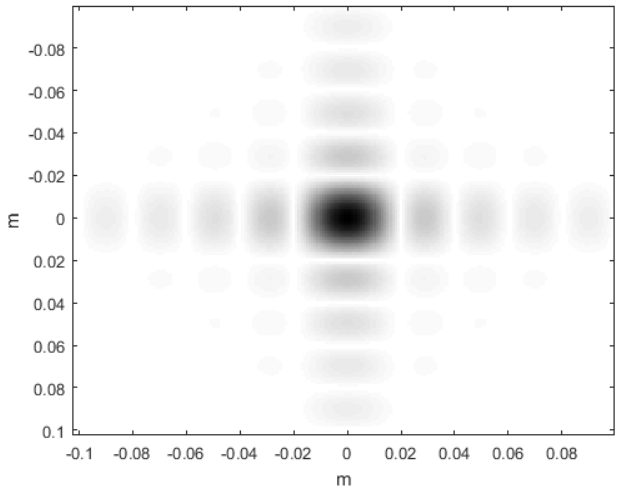

Figure 6 Fraunhoufer diffraction from square aperture at 10 m observation.

Fresnel Diffraction

If one is within the distance that does not allow the use of Fraunhofer diffraction, one can use the Fresnel diffraction approximation.

$$U(x_2, y_2) = \frac{e^{ikz}}{i\lambda z} e^{i\frac{k}{2z}({x_2}^2+{y_2}^2)} \iint_{-\infty}^{\infty} \left\{ U(x_1, y_1) e^{i\frac{k}{2z}({x_1}^2+{y_1}^2)} \right\} e^{-i\frac{2\pi}{\lambda z}(x_2 x_1 + y_2 y_1)} dx_1 dy_1 \quad (2.18)$$

The Fresnel approximation is more accurate than Fraunhofer, and treats the waves to be paraxial spherical waves (i.e., parabolic waves) at the observation plane instead of plane waves. Like Fraunhofer diffraction Fresnel propagation contains phase terms in front of the integral, based on the observation location and the distance propagated. The departure

from Fraunhofer is that the initial wave is multiplied by a phase term based on the location of the wave in the source plane before propagation occurs. The other difference is in form of the Fourier transform kernel

To demonstrate the use of Fresnel diffraction, I modeled the same square aperture from the previous example, except with an observation distance of 0.025 m. The diffraction pattern in this Fresnel diffraction case is interesting because the lobes have been modified by the additional phase terms and are more complicated (Figure 7).

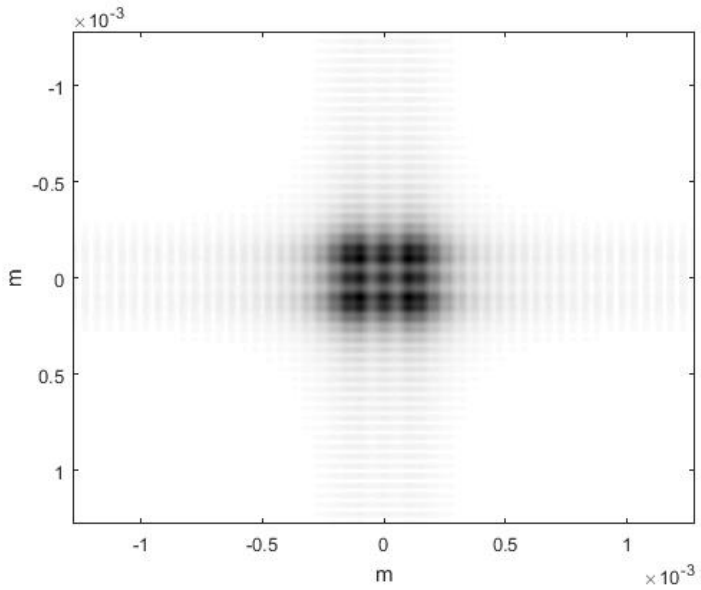

Figure 7 Fresnel Diffraction pattern of square aperture at a 0.025 m distance from the aperture. We see the Fresnel pattern repeats on the axis.

Turbulence

In Fourier Optics theory, turbulence is treated as a phase screen through which the wave passes. This means that the turbulence has no amplitude component, just phase components that are characterized as random patterns in the x and y directions. Real

turbulence evolves in time, which means that as time passes the phase at a given point will change. The phase screen for a given time can be numerically modeled by using the Kolmogorov model. The model is built on an inner scale l_o and outer scale L_o that describes the minimum and the maximum size of the turbulence eddies. The inner and outer scale act as the boundary conditions to the model. Kolmogorov developed this model by looking at the unit analysis to relate changes in the velocity and index of refraction [17].

The turbulence phase screen model is based on the structure function of refractive index,

$$D_n(r) = \begin{cases} {C_n}^2 {l_o}^{-4/3}, & 0 \leq r \ll l_o \\ {C_n}^2 r^{2/3}, & l_o \ll r \ll L_o \end{cases}, \tag{2.19}$$

where the scale of the turbulence eddies is given by the variable r. The ${C_n}^2$ is the refractive index structure parameter and is a function of P, the pressure in millibars, T, the temperature of the air in Kelvins, and ${C_t}^2$, the temperature variance. Using these parameters, the refractive index structure parameter can be written as

$${C_n}^2 = \left[77.6 \times 10^{-6}(1 + 7.52 \times 10^{-3}\lambda^{-2})\frac{P}{T^2}\right]^2 {C_T}^2. \tag{2.20}$$

To create a turbulent phase screen, the refractive index structure function is transformed to a power spectral density given as

$${\Phi_n}^K(\kappa) = 0.033{C_n}^2\kappa^{-11/3} \tag{2.21}$$

$$for \quad \frac{1}{L_o} \ll \kappa \ll \frac{1}{l_o},$$

$$\kappa = 2\pi\left(f_x\hat{\imath} + f_y\hat{\jmath}\right)$$

A phase screen is then created by applying a random phase to the spectral density at every point of f_x and f_y, then applying an inverse Fourier transform to recover the space domain. This produces a real and imaginary part, both of which are valid phase screens. I used the real part in my calculations

Theory of Holography

With an understanding of Fourier transforms and diffraction theory, it is possible to now discuss holograms and how to create them in terms of waves and optical fields.

LO & Target Beam Interactions

Referring back to Figure 4, we see that the light incident on the recording medium consists of two fields. The field from the object is called E_{signal}, and the field from the reference is called E_{LO}, where ideally the LO is a plane wave. E_{signal} can have several different forms in digital holography. The setup in Figure 4 shows the signal field from the object incident on the recording medium with no imaging lenses. If the distance between the object and the medium is far enough to meet the criteria for Fraunhofer propagation, the signal captured in this manner will be the Fraunhofer diffraction pattern. Alternatively, if the light field from the object is shorter, the Fresnel approximation can be used to simulate the signal. Where one decides to capture the wave determines what the signal will look like. If I capture in the image plane it looks like the target, if I capture the signal in the Fourier plane I see the speckle pattern (see Figure 8).

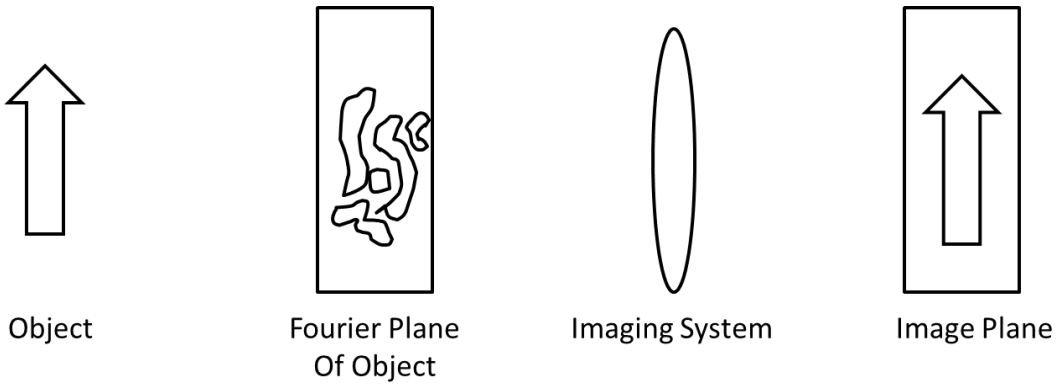

Figure 8 What you expect to see in the different imaging planes. The image plane should contain an in-focus image of the object, while the Fourier plane should contain an in-focus speckle pattern of the object.

The signal intensity in the Fourier plane is speckled due to the target not being a simple square or circle. This is due to the coherent light hitting the target which is represented by an amplitude mask and a phase mask that simulates the diffuse scattering nature of most objects. This is done by including a random phase for each point as

$$E_{signal} = U(x,y) \times A(x,y)e^{(i\varphi_{random(x,y)})}. \quad (2.22)$$

The LO is also collected onto the sensor with E_{signal}. The combination of the E_{LO} and E_{signal} is a simple addition. The LO, which is ideally a plane wave as shown in Eq. 2.20, is described by an angle θ_o between the LO propagation direction and the central ray of E_{signal} in real space. This angle, included in Eq. 2.21 is very important in digital holography, as it shifts the interference terms away from DC. The angle is the location of the pupil in the Fourier plane (discussed further below).

$$E_{LO} = ae^{-i\overrightarrow{k_0}\vec{r}} \tag{2.23}$$

$$\overrightarrow{k_0} = \frac{\theta_0}{\lambda} \tag{2.24}$$

The intensity of these signals is the absolute value squared of these combined examples, as shown in Eq. 2.22. The intensity image will produce the interference pattern that is seen in the cross-term of Eq. 2.23 and 2.24 while $\left| E_{signal} \right|^2$ and $|E_{LO}|^2$ are just intensity values and are unused in most digital holography applications.

$$I = \left| E_{signal} + E_{LO} \right|^2 \tag{2.25}$$

$$I = \left| E_{signal} \right|^2 + |E_{LO}|^2 + E_{signal}{}^*|a|e^{-i\overrightarrow{k_0}\vec{r}} + E_{signal}|a|e^{i\overrightarrow{k_0}\vec{r}} \tag{2.26}$$

$$I = \left| E_{signal} \right|^2 + |E_{LO}|^2 + 2|A||a|\cos\left[2\pi\phi_{object} + k_o\right] \tag{2.27}$$

Fourier Plane Holography

In Fourier plane holography, the recording medium (e.g. camera sensor) is placed in (or approximately in) a Fourier plane. The intensity image collected by a camera is the interference of the speckle field from the signal and the reference. To reconstruct the hologram and to be able to manipulate the phase, it is necessary to perform a two-dimensional Fourier transform on the intensity in the Fourier plane as

$$\begin{aligned}\mathcal{F}(I) = &\int \left| E_{signal} \right|^2 e^{-i\vec{k}\vec{r}} dr + \int |E_{LO}|^2 e^{-i\vec{k}\vec{r}} dr \\ &+ \int E_{signal} e^{i\overrightarrow{k_0}\vec{r}} e^{-i\vec{k}\vec{r}} dr + \int E^*{}_{signal} e^{-i\overrightarrow{k_0}\vec{r}} e^{-i\vec{k}\vec{r}} dr\end{aligned} \tag{2.28}$$

.

A feature of Fourier transforms comes into play here, as the individual intensity E_{LO} and E_{signal} values remain at DC (zero spatial frequency), while the cross-terms are shifted away from DC by means of the Fourier shift theorem.

$$G = \mathcal{F}(I) = DC + E_{signal}(k - k_0) + E^*{}_{signal}(k + k_0) \tag{2.29}$$

With the transform now completed, it is easy to see how depending on the angle of the LO field the hologram will appear offset from the DC term (see Figure 9).

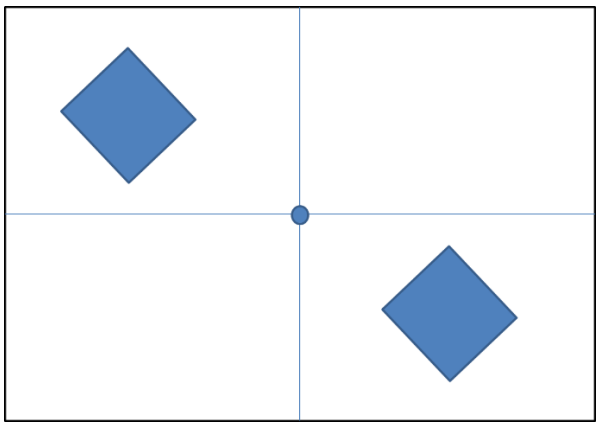

Figure 9 Example of Fourier Plane holography showing how the image of the object shows up in complementary quadrants of the Fourier plane. The circle in the center of the image is a representation of the DC terms that are present in the real image

LO Placement

The LO placement is critical to prevent aliasing of the holograms which would distort their phase information. By scaling the Fourier plane in degrees, it is possible to calculate in real space the proper location of the LO in relation to the sensor and optical

axis. This can become an issue if an optical system is in the way of the LO or the LO optics are in the way of the signal light. These adjustments are enabled by the LO usually being positioned on an optical mount and a mirror mount stage.

Correction with Parabolic

Once a complex hologram has been created it is possible to correct the image by use of phase correction methods. The simplest and most common correction is a quadratic correction in the form

$$exp\left[i\frac{\pi}{\lambda z_{offset}}[(x-x_o)^2+(y-y_o)^2]\right],$$

where $x_o$and y_o indicate the location of the center of the optical axis and z_{offset} is the distance to correct for the focus. Higher-order corrections also can be used to optimize the hologram image, such as a quartic (4^{th}-order) correction, which can correct for spherical aberration. The image correction can be done by visual inspection, as the images goes into and out of focus as you adjust these values. The visual inspection can be further improved through an automated check using image metrics. The z_{offset} is the distance from the center of a spherical wave that has the same wavefront.

Sharpness Metric

One automated check is called the sharpness metric, and has been well explained by Tippie and Fienup [18]. The sharpness metric has been a common way of approaching automated correction when looking at coherent images. The sharpness metric assigns a value based on the contrast in the image. This is done by taking the sum of the rows and

columns of the intensity image to some power of the hologram or an image one wants to correct as:

$$S = \sum_{i=1}^{m} \sum_{j=1}^{n} I(i,j)^{\eta} . \tag{2.30}$$

The most common predetermined power, η, is usually two.

The sharpness is measured after making a correction to see if it has improved the sharpness or not. If it has been improved, the change is kept and incremented again if it is worse, the correction is dropped and the change is incremented differently. This approach has the disadvantage common to optimization problems, where one is never completely sure if a global maximum versus a local maximum is reached. To make sure that a global maximum is reached requires looking at all possible corrections, which would take a lot of time and computational power.

This approach has been used by Liebling [19] to see if his holographic reconstruction method was putting the image into focus and showcases that one of the main uses for the sharpness metric is checking focus. We also employed the sharpness metric for our system corrections when working with actual lens systems.

HOLOGRAPHY SIMULATION

Now that I have established the theory behind holography, it is possible to create a computer simulation of forming and capturing a hologram. This is the subject of this chapter, including a simulation of what happens if turbulence is included. The simulation was written in MATLAB, a choice having more to do with my comfort level than any special advantages that MATLAB renders to the application. The simulation was based on examples from the book by Jason D. Schmidt [20], titled *Numerical simulation of optical wave propagation*. We made use of several of his functions to complete this simulation and they are listed with references in Appendix A. The setup emulated by the simulation is shown in Figure 10

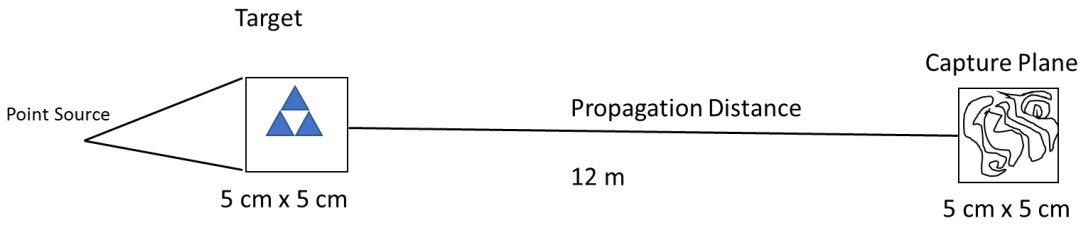

Figure 10 The setup for the simulation

Basic Properties

To start the program, I needed to setup some basic properties, including the distance from the target to the recording medium and size of these planes. The object plane was 5 cm by 5 cm and the size of the recording medium was also 5 cm by 5 cm. The distance that separate these two planes was 12 m, and the laser wavelength was 793 nm (this choice of wavelength is to match the experiment described in Chapter 5). The

method of numerical wave propagation that was used dictates the number of pixels chosen for each grid. I used an angular-spectrum propagation method as discussed on pages 95-102 in Schmidt's book. For this method, the pixel size was $4.4\ x\ 10^{-5}$ m.

Target

The target model started with a bit map image that acted as an amplitude mask on the transmit beam. To model the target as diffuse, it was given a pixel by pixel random phase. This resulted in extremely high frequencies that could lead to aliasing. Therefore, to prevent this, I convolved the target signal and a windowed point source, as seen at the sensor plane. This filtered the higher frequencies from the object. To perform the convolution, I Fourier transformed the point source and target, multiplied the two transforms together, and finally inverse Fourier transformed back to the image plane (see Figure 11).

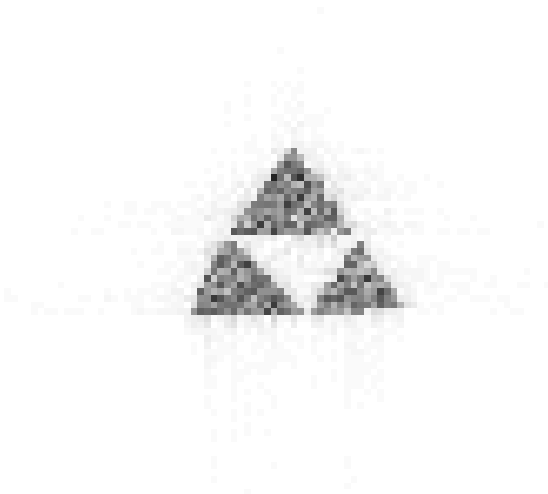

Figure 11 Image of target with the interaction with the laser.

Angular Spectrum Propagation

For wave optics propagation, I used angular spectrum propagation, which allowed control of magnification. The angular spectrum propagation function I used was created by Schmidt and explained in his book [21]. This function has the following variables for input that are required to complete the computation. It required the field to be propagated, the pixel size for the object space, and the pixel size of the pupil planes. The final parameter was the distance between the two planes. This function was used to propagate

the target signal from the object plane to the Fourier plane (see

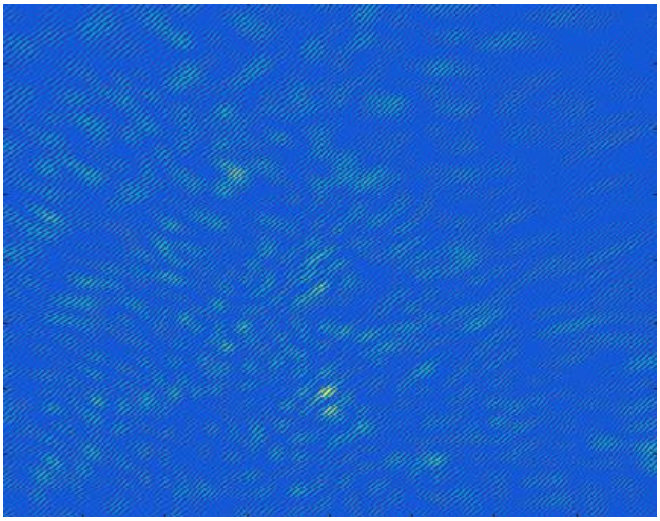

Figure 12). The function takes the inputs and creates the following three quadratic functions:

$$Q_1 = e^{\left(i\frac{k1-m}{2\ D_z}\left(x_1{}^2+y_1{}^2\right)\right)} \tag{3.1}$$

$$Q_2 = e^{\left(-i\pi^2 2\frac{D_z}{m/k}\left(f_x{}^2+f_y{}^2\right)\right)} \tag{3.2}$$

$$Q_3 = e^{\left(i\frac{k\ 1-m}{2(mD_z)}\left(x_2{}^2+y_2{}^2\right)\right)} \tag{3.3}$$

The quadratic functions combine with Fourier transforms denoted by $F\{\ \}$ in Eq 3.4.

$$U_2 = Q_3\, F^{-1}\left\{Q_2\, F\left\{Q_1 \frac{U_1}{m}\right\}\right\} \tag{3.4}$$

The three quadratic functions are used to simulate the propagation. These quadratic functions are made up of several variables, the distance traveled being D_z in

meters, and the magnification represented by m. The plane from which the light was being propagated was designated by a subscript 1 and the plane after propagation was denoted by subscript 2.

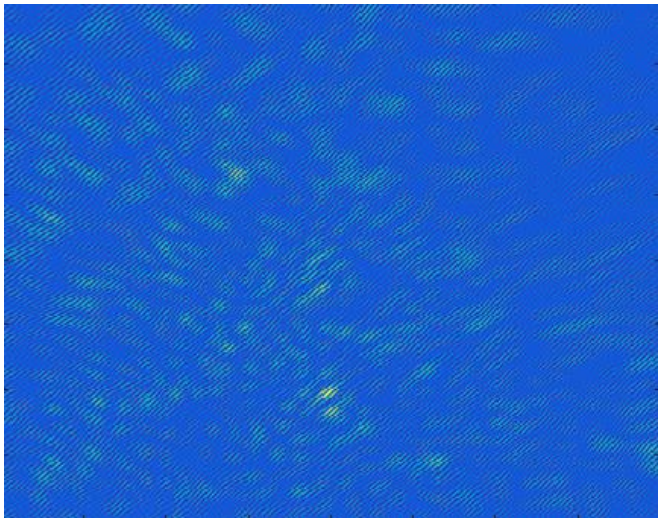

Figure 12 Propagation to the pupil plane of the target beam combined with the LO. This is the hologram of the diffraction pattern from the object interfering with the LO generating the fine bright and dark fringes.

LO

Once I had propagated the target signal to the Fourier plane, it became necessary to add the LO, which is a plane wave, to create the hologram. Since I knew the frequency of the image space, it was possible to set the LO angle so the image ended up in the middle of the quadrant. This was done by simply setting the angle of the LO to half the maximum frequency in the observation plane.

Image Plane

After adding the LO to the target beam, I could take the Fourier transform of the hologram of the scene and see the holographic image. It is important to note that, as the theory dictated, there were two images present in the quadrants (see

Figure 13 and Figure 14). The signal that is being captured is not perfectly in the far field so a parabolic phase correction needs to be applied. After applying the correction to the signal and looking at the holographic images, one is in focus while the other is highly defocused. This is due to the images in the opposite quadrants being complex conjugates of each other. The box in the center is the LO beam being defocused by the parabolic correction.

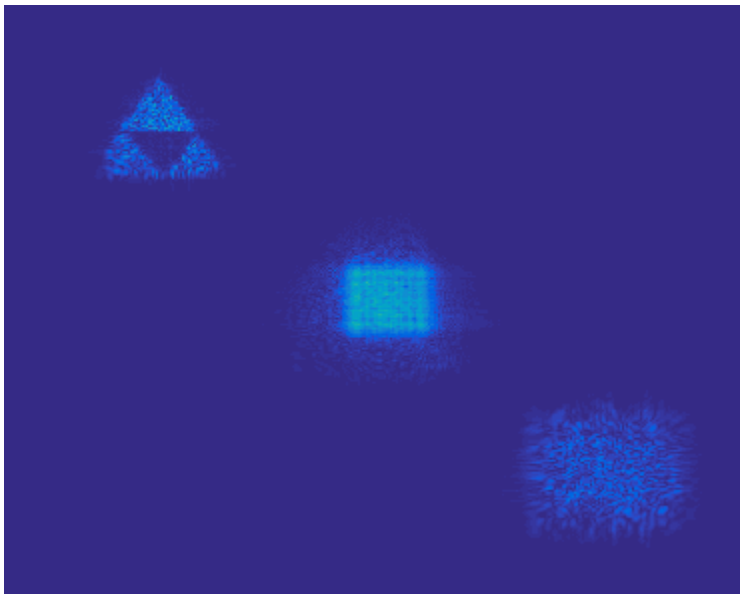

Figure 13 Image plane showing that our mathematical models were correct with the image being in both quadrants.

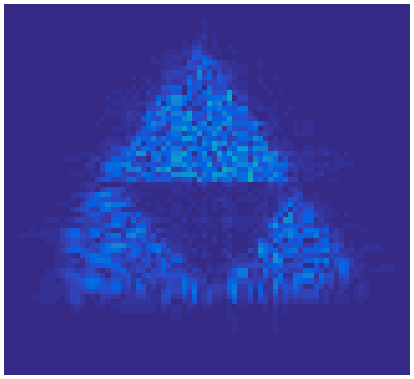

Figure 14 close-up view of the holographic image

Simulating Turbulence

To simulate turbulence in this program, I simply put a phase screen at the distance where we wanted to simulate turbulence. This was done by propagating to the phase screen location and multiplying the beam by the phase screen transmittance function. After applying the phase screen, the beam was propagated until it reached either the next turbulent phase screen or the observation plane. The creation of the turbulence phase screen was covered in great detail by Schmidt in chapter 9 [22]. Using a MATLAB function created by Schmidt and included in the Appendix, this function modeled the turbulence using the von Kármán spectral model. The von Kármán atmospheric phase was described using parameters r_o for the coherence diameter, l_o for the inner scale, and L_o for the outer scale, all in meters. The von Kármán parameters used to make the phase screen shown in Figure 15 are $r_o = 0.001$, $l_o = 0.001$, $L_o = 10$.

When this phase screen was applied to the target field after propagating to the observation plane, we saw that the holograms became unfocused (see Figure 16 and Figure 17). This is because the signal now had inconsistent phase elements caused by the cloud-like nature of the phase screen. This was applied right before the observation plane in the simulation, as this was similar to what was done in the experiment.

Figure 15 The phase screen acting as a model for turbulence. The cloud-like nature of turbulence is clearly visible here.

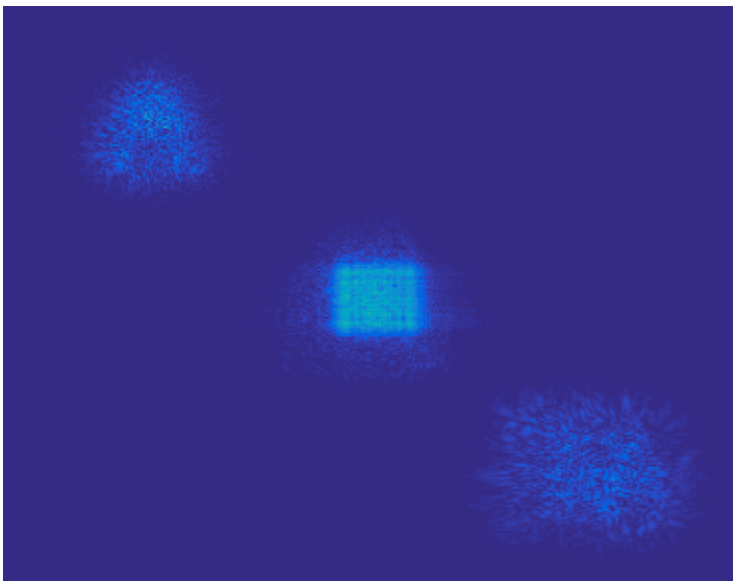

Figure 16 The image plane with turbulence added has lower resolution than before. Spreading out of the speckle is caused by refraction when light passed through by the turbulent phase screen.

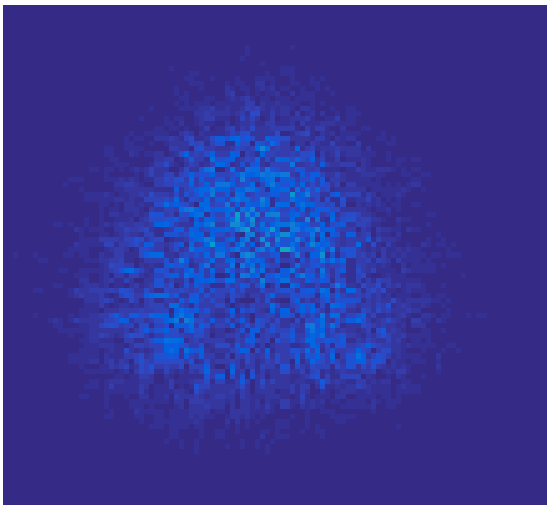

Figure 17 A close-up of the holographic image after the turbulent phase screen shows how the image has become less resolved.

HOLOGRAPHIC APERTURE LIDAR

How to Construct HAL Images

The ability to record phase information provided by digital holography makes it possible to do synthetic aperture imaging, called holographic aperture lidar (HAL). Making use of the phase information allows for several individual apertures to be coherently summed to each other. This was discussed first by Duncan and Dierking [23] as a method to use coherently illuminated active sensing to create higher-resolution images.

The increase in resolution is easy to see from Eq. 1.3 defining the diffraction limited resolution. Increasing the size of the aperture reduces the resolution distance (i.e. improves the resolution). As discussed in Chapter 1, often there are physical limits to how large you can make the aperture in optical systems. HAL gets around this issue by using smaller, well-behaved optical systems, then taking multiple apertures from different perspective and adding them in software to create a mosaic, which is the coherent sum of the shifted apertures, that forms an image from an effectively larger aperture with appropriately improved resolution in the image plane.

The concept of what is trying to be accomplished here is almost easier to understand visually than in terms of mathematics. The HAL image is made by placing the optical fields from the apertures taken from different locations properly in a larger field such that they combine coherently. The aperture can be collected by using relative motion, multiple apertures, or a combination of the two methods.

When the target is rotated to implement HAL, as done in my experiments, the rotation translates to a shift equal to two times the rotation in the aperture plane. One can see this by treating the object as a mirror, which when rotated causes the reflection angle to change at twice the angle of the incident light. This geometry can be seen in Figure 18. If we use a coordinate system like this, one can see that Eq. 4.1 and Eq. 4.2 are equivalent expressions.

$$G = F(E_{object}(x, y, z, \varphi = \delta\varphi)) \tag{4.1}$$

$$G = F\left(E_{object}\left(x - \frac{2\delta\varphi}{z}, y, z, \varphi = 0\right)\right) \tag{4.2}$$

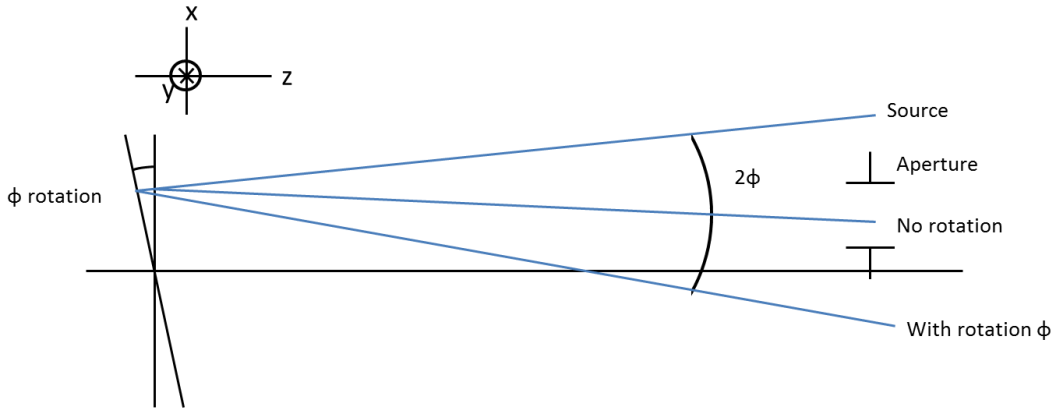

Figure 18 The geometry behind how inverse spotlight method works. By rotating the object, the speckle pattern actually moves allowing one to create larger synthetic apertures.

With these multiple digital holograms, it was possible to add them together if the difference in position of the two apertures is known. The relative position of the holograms could be determined by correlation. Correlation was done by taking the

inverse Fourier transform of the two apertures and, multiplying one transform by the complex conjugate of the other,

$$F^{-1}\{G_1\}F^{-1}\{G_2\}^*, \tag{4.3}$$

and Fourier transforming back to the aperture plane. Then, the maximum of this correlation was found and used to offset the second aperture from the other as

$$G_{corr} = F\{F^{-1}\{G_1\} \times F^{-1}\{G_2\}^*\} \tag{4.4}$$

$$offset(x, y) = Max\{\, G_{corr}\} \tag{4.5}$$

$$[x_{off}, y_{off}] = offset(x - \frac{x_{size\ of\ G}}{2}, y - \frac{y_{size\ of\ G}}{2}) \tag{4.6}$$

The offsets were used in the creation of a mosaic that added the Fourier plane of the image together in a larger aperture. This made it possible to get a smaller diffraction limit by increasing the effective aperture size.

The mosaic looks like Figure , shown with the pupils overlapping. These mosaics needed to have their amplitudes normalized due to some regions having more overlap in areas of the apertures where addition occurred. To retrieve the image the mosaic was inverse Fourier transformed to recover a higher-resolution image than from an individual rotation position.

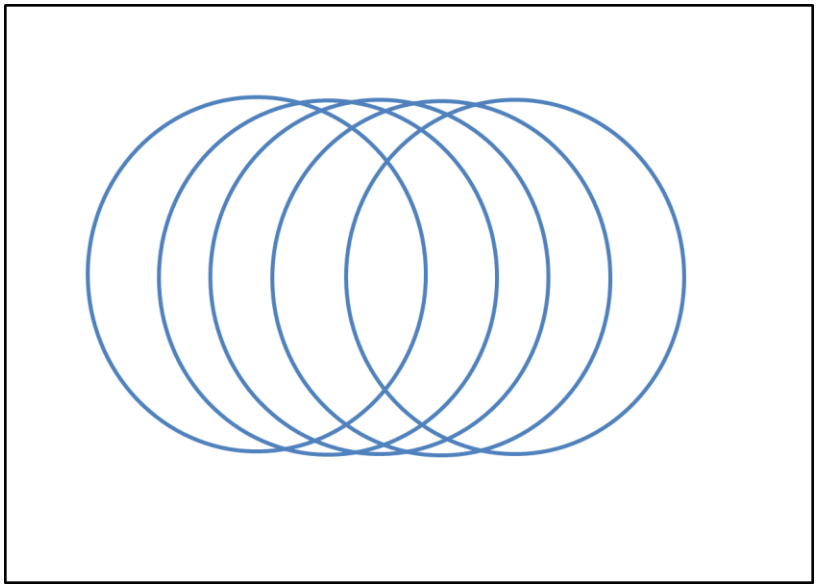

Figure 19 Diagram HAL mosaic with the circles representing the individual apertures. The amount that the aperture was offset was governed by the rotation of the system or target and determined by correlation

Issues with Turbulence

The time-evolving nature of turbulence can cause a problem for constructing HAL images. The evolution in the turbulence, in addition to causing blurring with individual apertures, makes the neighboring apertures not correlate well if they were captured at different times. This makes the phase information not well correlated, which makes determining the offset and achieving resolution improvement with HAL difficult. The presence of turbulence can be identified by the image jumping around in the image plane. In the Fourier plane, the turbulence actually changes the speckle that is recorded. Turbulence can be expressed as a random phase screen being added to the phase of the signal. This occurs where the signal passes through the phase screen and then is propagated onto the next element along the signals path.

The ability to correct for turbulence is a goal for any high-resolution imaging system including synthetic aperture imaging. The next sections explore different computational methods for doing so, including the method that is the subject of this thesis.

Anisoplatic Imaging

Turbulence can be represented as a series of phase screens in Fourier imaging as discussed previously. The Anisoplatic Imaging method was first created as a computational simulation by Fienup et al. [24, 25] as a way to help do phase corrections through turbulent phase screens. This initial simulation made use of simulated phase screens to act as different areas of turbulence. Signals were then propagated through the phase screens. The distorted signal was propagated back to the simulated phase screen location, and a correction applied to the signal. Then the signal would be propagated back to the image plane to observe the correction. This worked quite well for Fienup et al. [24,25] in simulation, so they proceeded to try it in real lab experiments. Tippie [26] further developed the method for the real-world experiments. The method involved propagating the wavefront from the image plane back to one of the turbulence phase screens, adding a correction and propagating back to the image plane. Though unlike the simulation where the phase screen was known, in the lab the phase screen was unknown and so the correction was less certain to be successful. The sharpness was checked to see if there was an improvement made. This moving back and forth between the phase screen and the image plane to check to see if the correction was improving the sharpness took a

lot of computational time. However, this process did maintain the phase information between apertures well enough to allow for HAL imaging.

The most pressing issue in this approach was the need to have computational resources to do the necessary calculations, which Tippie pointed out as one of the faults with this technique [26]. She stated that a supercomputer was needed to do the correction, and that the creation of the mosaic for HAL took over a day, although it was never stated if that was due to the large nature of her mosaic or the process of doing the corrections to said mosaic.

Phase Gradient Autofocus

Another method that has been explored for correcting aberrations induced into digital holography by turbulence is the phase gradient autofocus algorithm. The original method, as explained by Ghiglia and Jackowatz [27] was used to correct synthetic aperture radar data. The process began by taking a complex image and cutting it into a series of cells. The phase gradient estimates were calculated for each cell and then used in a Poisson solver to find the phase gradient error. The phase gradient error was then removed from the image after doing some interpolation to get back to whole image size. The estimator used to find the reconstructed phase was found to be linear [28]. The key assumption for the PGA is that the phase error from the turbulence was common to every portion of the image, and that the scene contained bright glints or edges that should be sharp and could act as references.

Ghiglia with Romero [29] further developed this method to deal with sheared phase that can be present in some complex images. This further development has been about

smoothing out the phase difference, but not actually dealing with phase noise like that generated by turbulence.

Stephen Crouch [30] applied PGA to synthetic aperture lidar (SAL). He was mostly looking at a way to correct the phase error occurring in SAL data that he had collected. He made the point that, at least for SAL, to make good use of this method he had to have bright targets to achieve pleasing results. This method was described in greater detail with several examples of how to build a SAL system by Barber and Crouch [31].

PGA has been used by another group to correct digital holograms [32]. PGA can require a lot of computational power because you need to cut up each image into a series of cells, and if you have a large image that can mean a lot of cells that need to have their gradients calculated. Nevertheless, PGA is probably more computationally light than Tippie's method.

Phase Gradient Averaging

The concept for phase gradient averaging was a fairly straightforward one, with two basic assumptions. The first assumption was that the target is static, not evolving in time. The second assumption was that the turbulence is dynamic and random, that is, it evolves in time in a random way. These assumptions lead to the understanding that the phase from the turbulence should average out over time, while the target phase should not. The main issue with this is that digital holography as a coherent imaging method tends to have large piston phase errors in time, which causes the target phase to average out also. To get around this issue, we instead averaged the gradient of the phase because

it was not affected by the piston phase. The time average of these phase gradients was expected, to send the contribution from the turbulence to zero. After calculating these averaged gradients, it was possible to use a phase reconstructor to make a new phase that contained the signal and not the turbulence.

Taking the Phase Gradient

How to take the gradient of the aperture plane was a point of discussion for this method since there were several ways to go about it. Ghiglia described a way to do this using Fourier methods [29]. With this method, one starts with the complex image $g(x, y)$ and its Fourier plane is defined as $G(u, v)$ [29]. The gradient of G was found by multiplying g in x and y by a ramp function using the theorem

$$F\left[\frac{d}{dt}f(t)\right] = (-i\omega)f(\omega). \tag{4.7}$$

This product was then forward Fourier transformed to be the gradient of $G(u, v)$ for both $G_x(u, v)$ and $G_y(u, v)$. To get the phase gradient, the complex gradient was multiplied by the complex conjugate of G and the imaginary part taken. Then the product was normalized by the intensity of the $G, i.e. |G|^2$.

I also tried other methods to see if I could get better results. By making use of the angle function in MATLAB, I found the angle of G, and then calculated the gradient by means of the gradient function in MATLAB. The gradient function produced gradients in the x and y directions.

Both of these methods could produce pixels with not-a number (NaN) as values, so a check was used to look for and set NaN values to zero. This prevented an issue with the reconstructors returning all NaN if even one element of an array was NaN.

Reconstructor

The reconstructor was an important step in getting the phase gradient. We found two reconstructors on the Mathworks file sharing site: intgrad2 [33] and the ImageRecH [34]. These two reconstructors have some different requirements but intgrad2 is the simplest, since it only requires two gradients in the x and y direction. These two gradients need to be the same size and also not contain NaN elements.

The ImageRecH has an additional requirement that the gradients be Hudgin gradients. Hudgin gradients are gradients where the last column for the x gradient is zero and the last row being zero in the y gradient.

Test of Reconstructor and Gradient Methods

To test these different reconstructors and gradient methods, I made use of a test pupil that contained a random wrapped phase. The goal was to see if I could return to this same phase after taking the gradient and reconstructing from the phase gradients (see Figure 19).

Figure 20 shows the different combinations of reconstructors and phase gradient. Using the phase gradient method given by Ghiglia it can be seen that there is something strange going on with the reconstruction. This is due to how the phase is unwrapped in this method and how the constructor treats the unwrapped phase

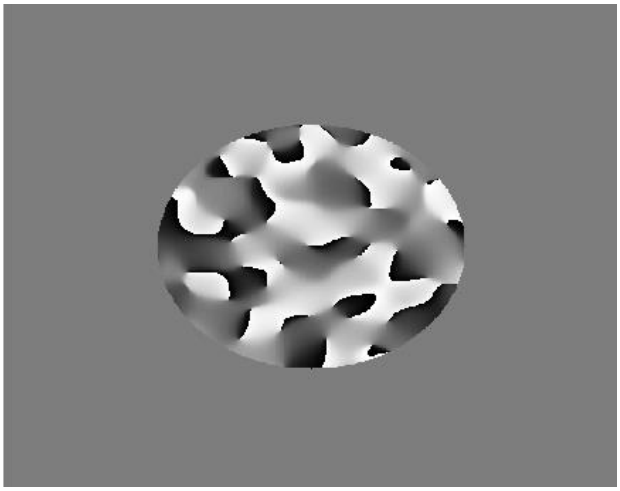

Figure 19 The base pupil that is the goal to return to in reconstruction.

The angle method for phase gradient extraction seemed to be better suited for the constructors that we found. The intgrad2 with the angle method seemed to be the closest to returning to the original test pupils' phase (see Figure 20). There was some minor pixelation occurring around the edges where the phase changed quickly. The ImageRecH constructor gave even more problems around the regions of quickly changing phase when used with the angle gradient method (see Figure 20). This is seen by looking at the edges where it is more pixelated than the intgrad2. This pointed toward the use of the angle method with intgrad2 being the one to be used in the reconstruction of the actual data that were collected.

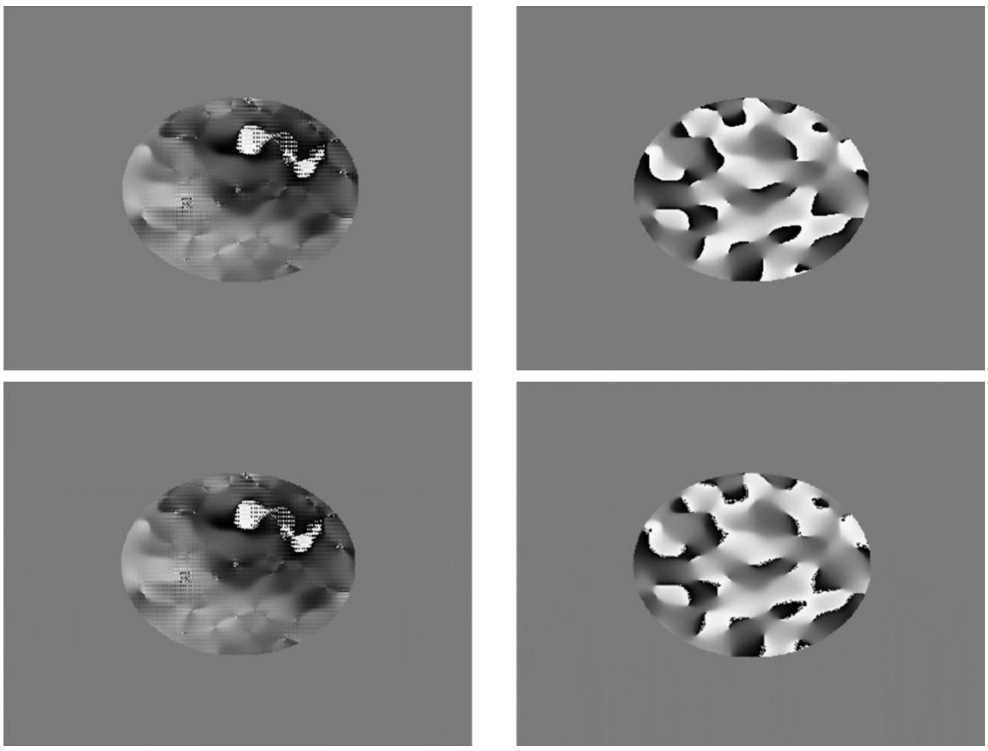

Figure 20 The Ghiglia with intgrad2 reconstruction (Top Left), Ghiglia with ImageRecH reconstruction (Bottom Left), Angle with Intgrad2 reconstruction (Top Right), Angle with ImageRecH reconstruction (Bottom Right). It is clear that the angle method returns better results, and the Intgrad2 constructor doing the best job getting back to that original pupil.

EXPERIMENTAL SETUP

With all the theoretical background presented, it is now possible to talk about the physical experiment that I conducted. This gives me an opportunity to write about some decisions I made and their motivations. The most obvious difference from the theoretical discussion, is that in my experiment I used an imaging system and placed the camera at the image plane, as shown in Figure 22. This was done for a few reasons, but the most pressing was because I had a diffuse target, which returned a small amount of light to the HAL system, and the sensor was so small it was hard to collect much light. I needed to maximize the amount of signal because of the low power of the returning signal captured by, the 50-mm diameter lens.

Another important reason for placing the camera in the image plane is that when a forward Fourier transform is performed on the image, one gets the hologram of the far field, which is restricted by the aperture of the lens system. This allows for a speckle field that has a defined shape limited by the aperture. A benefit of this is that the correlation for creating the HAL mosaic is well behaved.

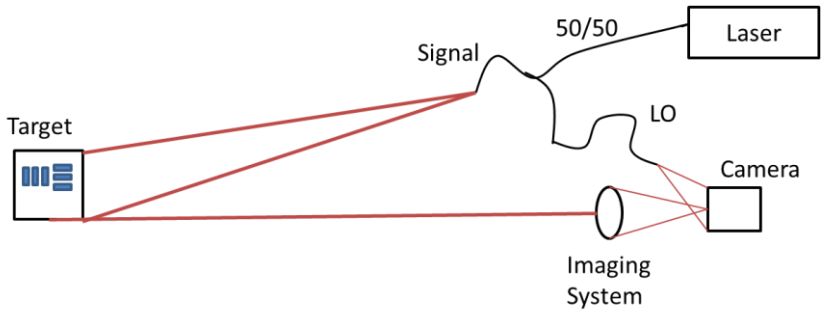

Figure 21 Basic setup of the HAL imaging system.

Camera

The requirements for the camera was a low exposure time for "freezing" the turbulence, a high data transfer speed for recording frames quickly, and a high pixel count for the sensor. These requirements led me to choosing a camera from Point Grey: the Flea3 monochromatic camera (see Figure 22). The camera is now available from Flir who bought Point Grey. The camera had a 1328 x 1048 pixel sensor with a pixel size of 3.63 µm. The Flea3 is a usb3 camera with a high data transfer rate to the computer. The exposure time can be set between 0.008ms to 1s. We went with 1 ms to allow for the evolution of turbulence to occur between shots. The camera had dimensions of 29 mm x 29 mm x 30 mm, a common size for scientific cameras that fit well on the optics bench. A practical note was that the camera got very hot to the touch while running at full transfer rate.

Figure 22 Flea 3 camera [35] connects to a computer using a usb3 connection for fast data transfer. This camera sensor is monochromatic, with a 1328 x 1048 pixel sensor.

Target

For a target, I used an Edmunds Optics USAF 1951 resolution target mounted onto a mirror mount with magnetic tape. This is a bar target that lets you determine image resolution by seeing what is the smallest resolved line element. Table 2 shows the line elements that were on the target, in cycles per millimeter.

Table 2 The table of resolution [36] found on the back of the case that the target came in. This means that some targets may not have exact same spacing in groups if not using the USAF standard. Our target follows the USAF standard.

Resolution Values for 1951 Resolution Test Pattern (Cycles per millimeter)								
	Groups							
Elements	+0	+1	+2	+3	+4	+5	+6	+7
1	1.00	2.00	4.00	8.00	16.0	32.0	64.0	128
2	1.12	2.24	4.49	8.98	17.9	35.9	71.8	143
3	1.26	2.52	5.04	10.1	20.1	40.3	80.6	161
4	1.41	2.83	5.66	11.3	22.6	45.3	90.5	181
5	1.59	3.17	6.35	12.7	25.4	50.8	101	203
6	1.78	3.56	7.13	14.3	28.5	57.0	114	228

The target that I used had the 0 and higher numbered groups for checking the resolution. The part of the actual target that contained the features of interest was 2.5 cm x 2.5 cm. I spray painted the back of the target with white enamel paint so that the returning signal I received would be higher. This was after I tried to do some captures and not having a very high return. The target was made of glass with etched-in markings (see Figure 23).

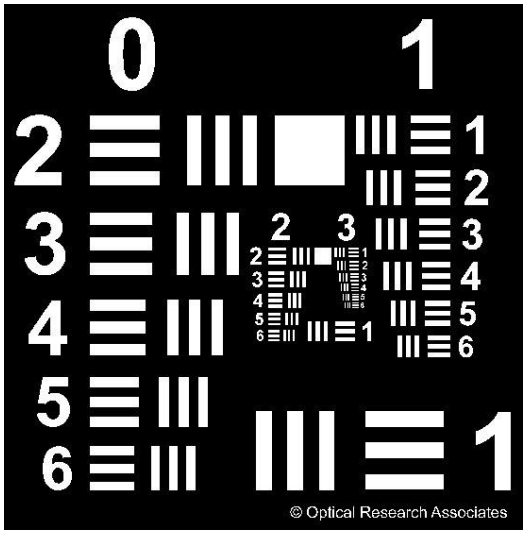

Figure 23 Example of image of the USAF 1951 resolution target [37], that was used in my experiments.

Rotation Stage

To rotate the target in the way needed to produce the HAL images, I placed the target on a rotation stage. The rotation stage also had the benefit of consistent rotation that would only move by the given amount with a very small amount of error. I used a Galil RSR 100 direct drive rotary stage (see Figure 24), which had a minimum rotation of 0.021 arc-seconds, or 5.8×10^{-6} degrees. The rotation stage was connected to the computer by means of an Ethernet cable. To successfully communicate to the stage, which had a built-in IP address of 10.10.10.2, the computer needed to have its IP address changed to 10.10.10.1.

Figure 24 Rotation stage [38] that required a special controller (not shown here) to make use of at Ethernet connection.

C++ Software Controller

To control the rotation stage and the camera at the same time it became necessary to write a software controller. I wrote this controller in C++ because Blackmore, the company that subcontracted the work for this master thesis, wanted to have a proprietary-free controller. The process of making the controller was one of loading in the correct libraries and header files for each piece of equipment. This process involved detailed reading the documentation provided, as well as looking at example code that Point Grey and Galil provided. Implementing the commands for these devices was where the example software became especially helpful, as they provided the correct syntax for connection and communication.

The place where documentation seemed to fall short was in controlling the camera's properties. There were no clear examples of how to turn off autofocus and auto-brightness. For example, to find a clear example I had to search online until I found

documentation for the SDK for Point Grey that was several years old and was no longer on their main website.

The controller, when started, asked the user to name the images, the number of images to be captured per aperture position, the number of rotations at a given rate, and the exposure time for the camera in milliseconds. The controller then opened three windows. One of these windows was a histogram that showed how many pixels were at the levels from 0 to 256, which is from black to white in a monochromatic scale. This was used to make sure that the image was not being over or underexposed when illuminated by the LO and the target signal. The LO placement has been discussed in terms of its angle, but an issue that has not been covered was the need for the LO to not overpower the scene and cause saturation of the pixels. This was done by finding the maximum intensity of the LO on the camera sensor at full power, then air gapping the fiber connection to bring the power down. Air gapping is when a fiber-to-fiber coupler is slightly unscrewed, which produces an air gap between the fibers, which reduces the power coupled from one fiber to the other. Using the histogram window from the software controller to see if the LO was overexposing the pixels was one of the main uses of the histogram.

The image from the camera also was viewed to checked LO placement and target focus. The Fourier transform of the image plane helped with setting the angle of the LO so that the hologram appeared in the quadrants and was not aliased. When the “c” button was hit, the three windows closed and the images were captured in a PNG file format. The file naming convention was file name root, position number, and the image number

for that position. The rotation stage rotated after capturing a number of images for each position. After capturing, the three windows were reopened and could be viewed again. This software controller is included in Appendix A.

Laser

The light source for this project was provided by a 793-nm laser. This means that the laser is very close to being in the infrared range. The light for the target beam and the LO came from the laser and were split by a fiber coupler/splitter that had a 50/50 ratio. I used this laser because it was available in the lab and the camera sensor could see this wavelength of light.

The target beam was sent to a fiber collimator (see Figure 25) that had its focus adjusted so that the spot from the laser light was roughly the size of the bar target once the beam had reached the target. The LO emitted from the fiber so it could be treated like a point source, with no collimator. The fiber connector was held by a plastic cap with a small hole drilled in it for the light to escape (see Figure 25).

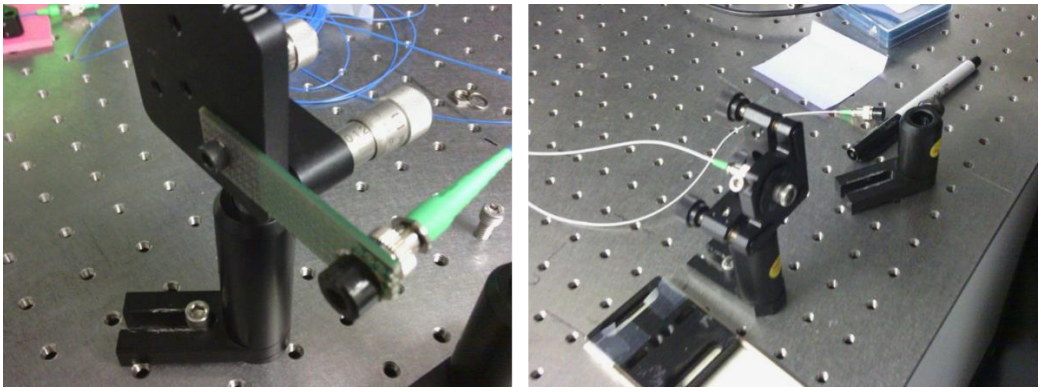

Figure 25 LO setup (Left), Transmitter (Right). The LO needed to have a custom stand made for it to allow it to reflect at the desired angle and not hit the lenses. The transmitter has a collimator to help control the spot size on the target.

Turbulence

To create atmospheric turbulence in the lab, I used a space heater. With the exception of the 750-mm lens setup, where it was placed underneath the beam, the heater was placed off to the side. The heater was placed 122 cm away perpendicularly from the beam path. The heater was placed 60 cm vertically from the beam path by being placed on a stool. This case was called the turbulence case, due to the turbulence being present. The turbulence moved across the aperture in a horizontal direction.

Optical Setups

The need to image a small, far away target meant I needed to build an imaging system. Several setups were tried to for collecting the data, building the HAL images, and testing the phase gradient averaging method.

750-mm lens, 12m Away

The first optical setup tried was the longest, with the target being placed 12.1 m from the receiver. This was only possible because of the space provided by Blackmore Sensor and Analytics, who opened up their basement lab space for me to use. The optical system that I initially used had a single 750-mm focal length lens. This lens had a diameter of 50 mm, so it created a large aperture in the Fourier or aperture plane. The distance from the lens to the camera was 825.5 mm. The LO was placed 76.2mm perpendicular to beam path and 381 mm away from the camera (see Figure 26).

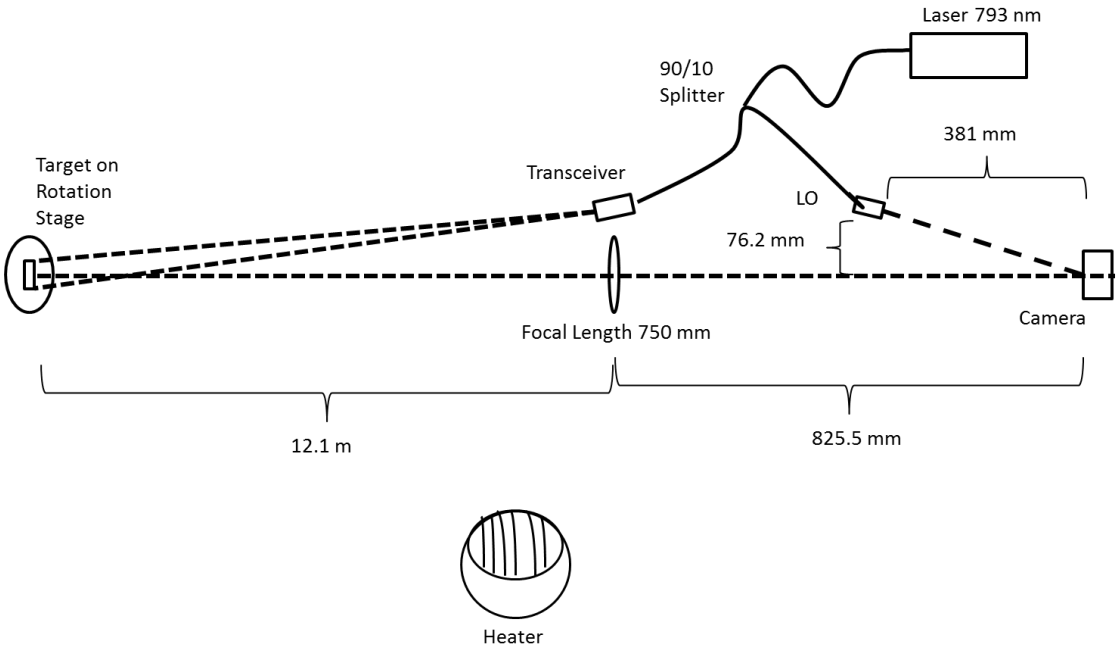

Figure 26 Schematic of 750 mm lens setup, the longest-range setup that was tried. Even though we got useable results the magnification of the target made it difficult to see if the resolution had been improved by the phase gradient averaging method.

A major flaw of this design was that the field of view was too large. If we look at the magnification we see that the target became very small and did not fill our sensor. This

led to poor resolution individual images. This is demonstrated below with s_{image} being the distance from the lens to the image plane and, s_{object} the distance from the object to the lens.

$$M = -\frac{s_{image}}{s_{object}} = -\frac{825.5mm}{12.1m} = -0.0682 \quad (5.1)$$

$$-My_{object} = y_{image} = 0.1705 \text{ cm} \quad (5.2)$$

This means that the target only took up about one third of the camera sensor, which was 3.8 mm on its shortest side. This led me to develop an imaging system that would fill the camera sensor.

Telephoto 5m Away

Another imaging system was developed with the specific purpose of having the target area of interest fill the sensor. The solution that I came to was to build a telephoto lens system with lenses that were available in the lab. I was able to use a 200-mm positive focal length lens with a diameter of 50 mm, and a -100-mm focal length lens with a diameter of 25 mm for building the system. By looking at the magnification calculation, Eq. 5.5, I determined that I needed a system with a 659-mm focal length. To help the return signal to be higher, we moved the target closer to the receiver. The new distance was set to be 5 m, as illustrated Figure 28.

To setup the distance needed to achieve the goal of getting a 659-mm effective-focal-length system, I used Zemax Optical Studio to design the system (see Figure 28). I set the distance between the two lenses to be the only variable and also constrained it such that the concave lens would not jump in front of the positive lens. For a 659-mm effective focal length, the Zemax optimizer places the lenses 12.7 cm apart. This produced a back focal length of 22.6 cm. In this setup, the LO was placed 13 cm away from the camera and 2 cm away from the beam axis (see Figure 27).

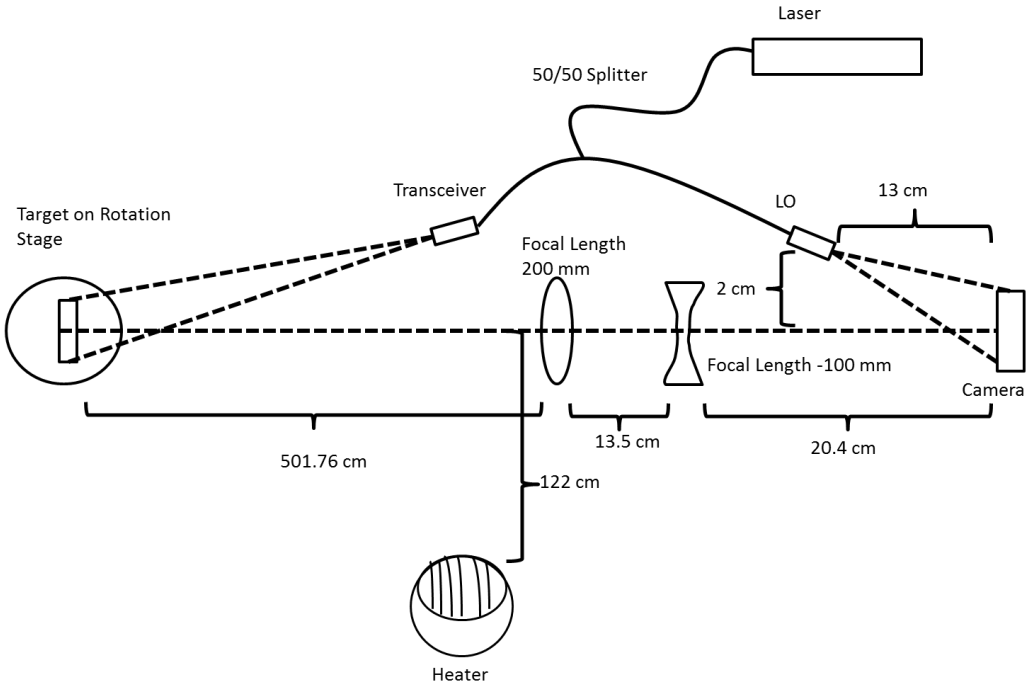

Figure 27 Telephoto schematic. This design did not produce useable results; it seems to be that the LO was placed to close so that it still had too much curvature when it reached the sensor and the telephoto lens had large aberrations.

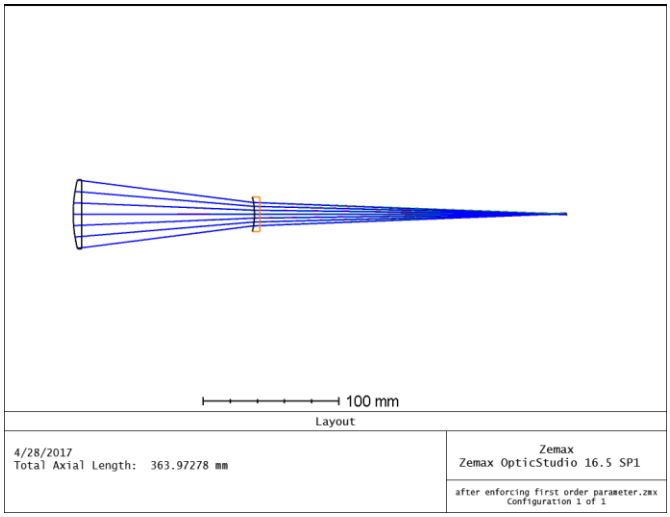

Figure 28 Zemax schematic of the telephoto lens design showing the distance to the image plane.

The advantage of using Zemax for this application was the ability to look at aberrations that were present in the optical system. Looking at the Zemax spot diagrams (see Figure 29) and the Seidel coefficients (see Figure 30) it is easy to see that this design had spherical aberration. Since spherical aberration depends on ρ, which is how far from center of the lens that the light is entering the lens system, one could put in a physical aperture or inscribe a smaller aperture in post processing to improve the images.

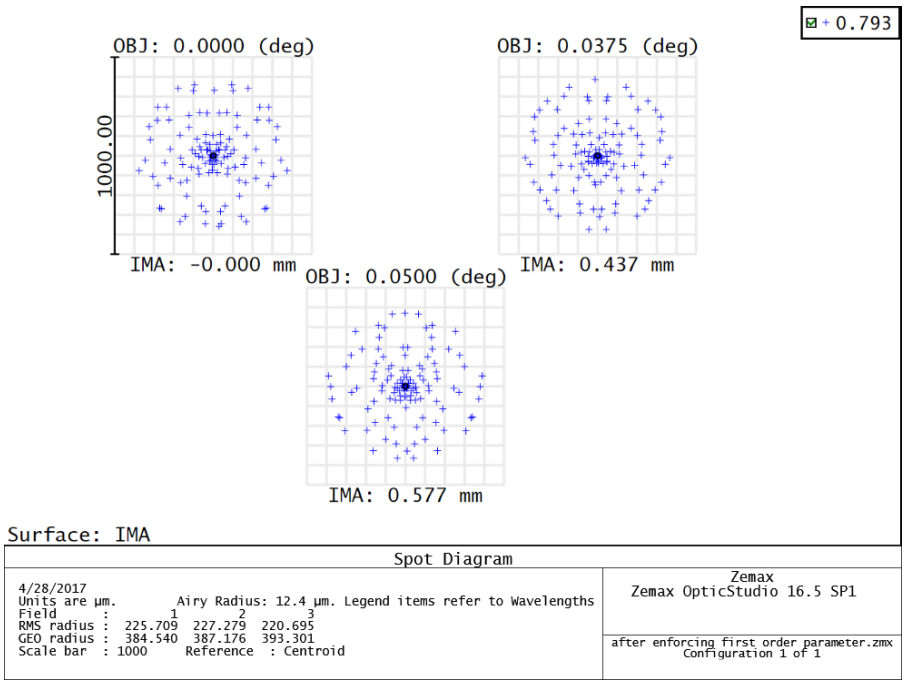

Figure 29 Spot diagram of telephoto lens system. It is possible to see that the spherical aberrations are significant by the very small circle representing the diffraction limit present in the spot diagrams.

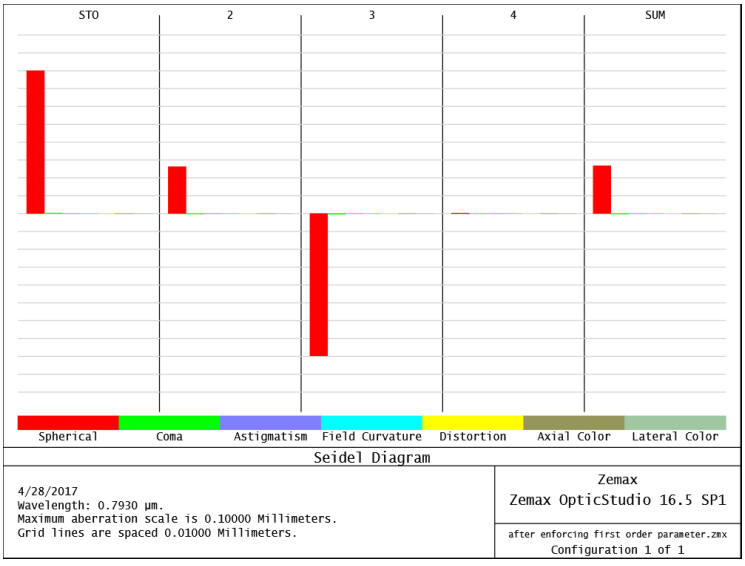

Figure 30 Seidel diagram with the spherical aberration shown again to be the dominate aberration.

The large spherical aberration made the digital holography images have very large radial blurring. I tried to correct this using an additional 4^{th} order correction term in addition to digital focus correction, but this was not sufficient to make the imagery look good. So, in consultation with Dr. Barber, I decided to not use this setup for further data.

500mm lens at 5m Distance

To see if we could get better resolution with a single lens, I made another lens system with a 500mm focal length, 50mm diameter lens. This imaging system had magnification of -0.11, according to Eq. 5.5 which meant that the target size was 3mm on the 4.8mm side of the camera sensor. The image of the target filled most of the sensor area, which was one of our goals for the system. The lens was 55 cm away from the camera, and the LO was placed at two different locations: one 13 cm away, similar to what was used for the telephoto system (see Figure 31). The other position was located 54 cm away from the camera and 4.5 cm from the beam axis (see Figure 32). The two positions revealed that the LO position had a significant effect on the resulting image.

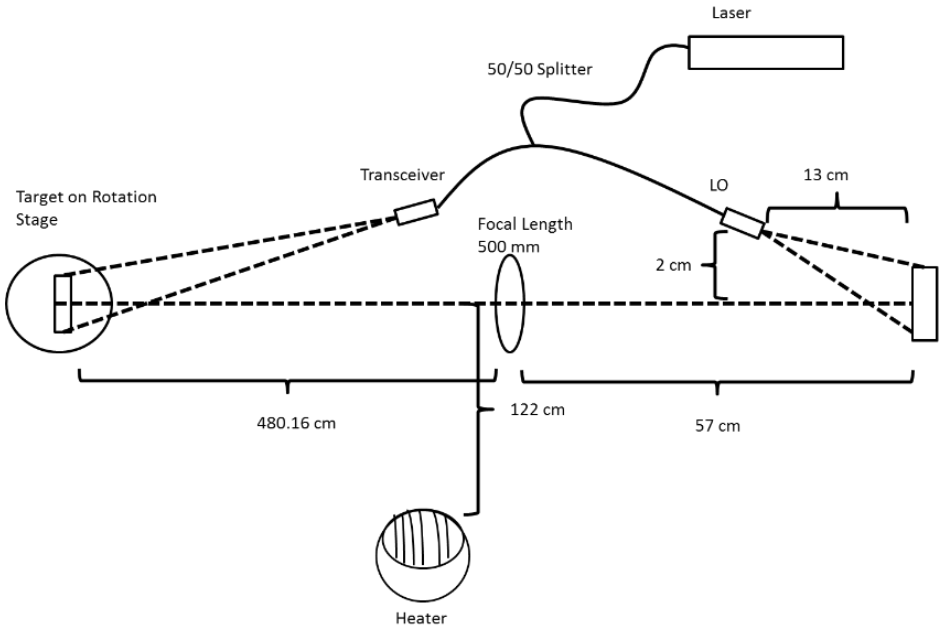

Figure 31 Schematic of 500 mm lens imaging system with short LO distance. This setup produced unusable images compared to the setup with the longer LO distance.

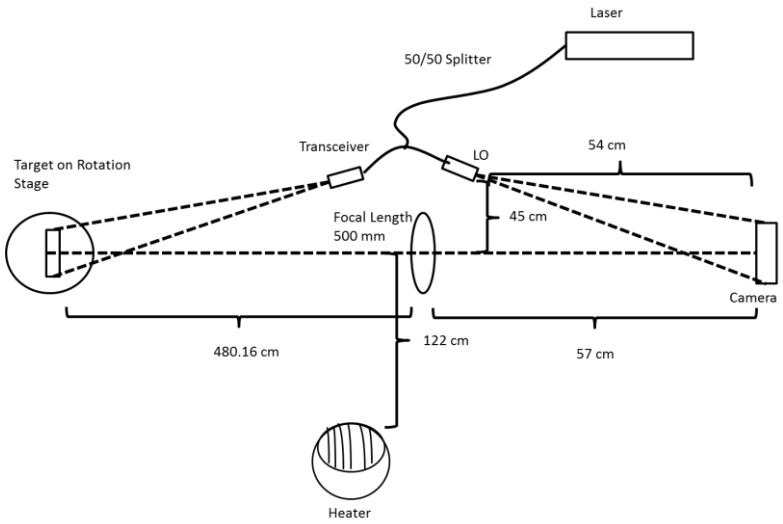

Figure 32 Schematic of the 500-mm lens imaging system with long distance LO. This produced the best results and the one that was used for most of the analysis

In general, to construct good images, the optical system needed to have a magnification that allowed for the target to fill most of the sensor. The system also

needed to allow for the LO to have as flat of a wavefront as possible on the camera. This was due to the LO defocus adding to the requirements for aberration correction to the image.

To understand this better, let's look at the curvature of the LO beam at the two distances for the 500-mm focal length imaging system. A laser beam emitted from a fiber (like the LO) can be modeled as Gaussian beam for determining its radius of curvature (see Eq. 2.6). To make use of this model, we needed to know the waist location z_o,which was at the face of fiber in this case. The waist size was about 2 µm when it left the fiber. This means that the Gaussian beam becomes a spherical wave since looking at Eq. 2.6 if z_o is 0 since it was at the surface of the fiber meaning that $\left(\frac{z_o}{z}\right)$ was also 0, thus making the Gaussian a spherical in terms of wavefront curvature. This led to the spherical wave model for the wavefront curvature, which is the distance from the origin point of the laser to the camera sensor. It is important to note that this is not a true spherical wave, since it is only part of the wave and not the complete sphere. This means that the radii of curvature are 13 cm and 54 cm for the short and long distances, respectively.

This means that the amount of aberration that the LO was contributing depended on the distance from the camera at which the LO was placed. The relative distance, Δz, between the spherical wavefront and the flat wavefront a distance, Δx, from the center of can be estimated by:

$$\Delta z \approx \frac{1}{2}\frac{\Delta x^2}{R} \tag{5.3}$$

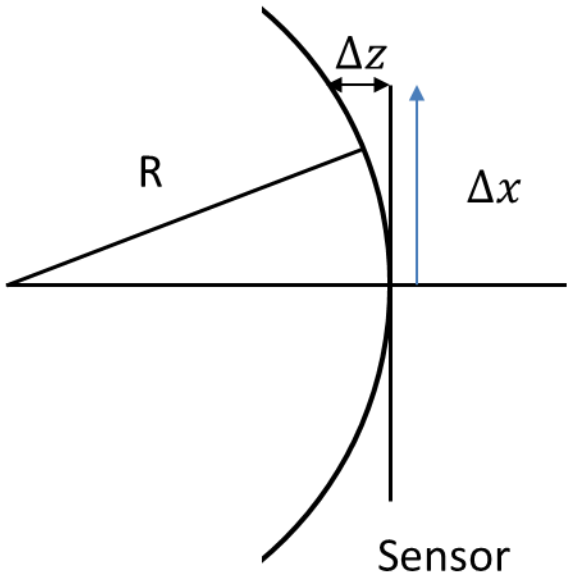

Figure 33 The visualization of the circular fringes.

This means that the total fringe depth for the 13 cm LO was 13.9 µm. The total fringe depth was 3.3 µm for the 54-cm setup of the LO. This clearly shows the fringes are significantly flatter for the 54-cm case.

TESTING PHASE GRADIENT AVERAGING FOR SINGLE APERATURES

Digital Holography Processing

To create digital holographic images, it was first necessary to capture the LO and the target beam together on the camera sensor. After capturing the image, a Fourier transform was performed on the image. This generated the Fourier plane with “images” of the lens aperture being contained in opposite quadrants. This aperture placement was determined by the LO angle. We chose to cut out the upper left quadrant “image” out of preference, as both apertures contained the same information. Next, I zeroed out the quadrant except for the aperture by using a circle mask consisting of an array of zeros and ones. At this point we performed an inverse Fourier transform and got the single aperture images. To improve the focus, a quadratic correction was applied to the aperture plane and the sharpness was calculated to see if the correction improved the image quality. This quadratic correction was then applied to all the images for that system to see if the image resolution was improved. A visualization of this process can be seen in Figure 34.

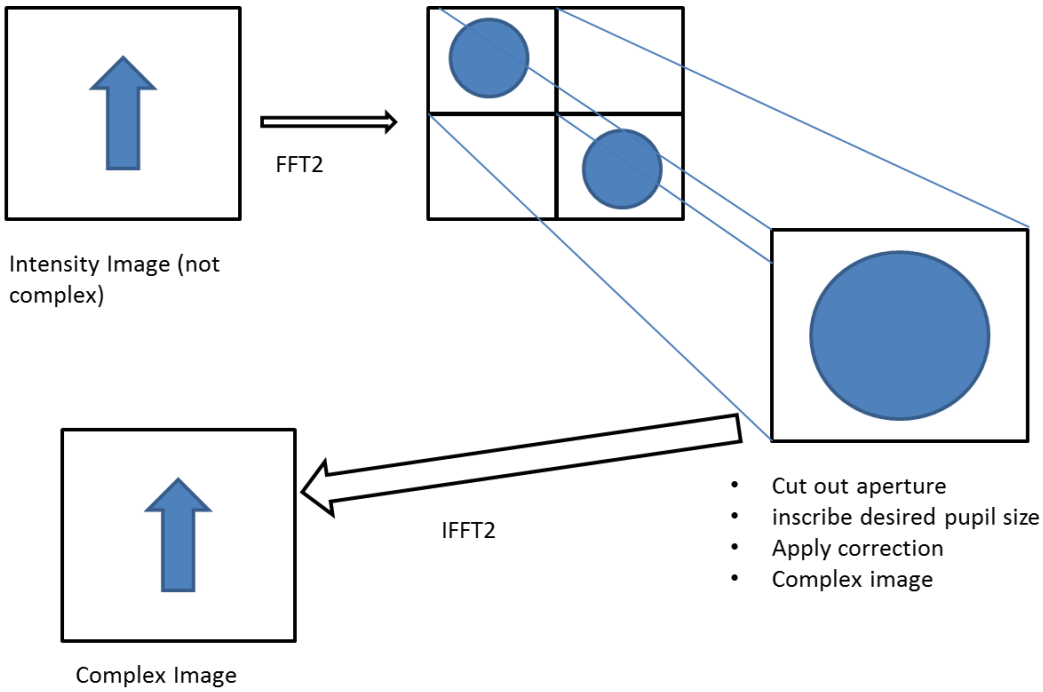

Figure 34 Digital holography processing procedure

Single Aperture Images

Looking at a single-aperture image allowed me to judge the initial quality of the images and decide if I should continue taking data with that setup into the HAL and turbulence experiments. All the following images had the quadratic phase corrections applied, appropriate to each system. All the images used an inverse gray scale for printing so brighter pixels showed up darker when printed in black and white.

750-mm lens at 12 m System

The first system developed to test this method was the 750-mm focal length system, located 12 m from the target. This resulted in a tiny hologram of the target in terms of the field of view. One could view the 0-1 element, so in consultation with Dr.

Barber, I decided to proceed with the turbulence and HAL imaging experiments (see Figure 35). Though usable, we thought it would be better to have an increased amount of the field of view being taken up by the target, so I developed other systems.

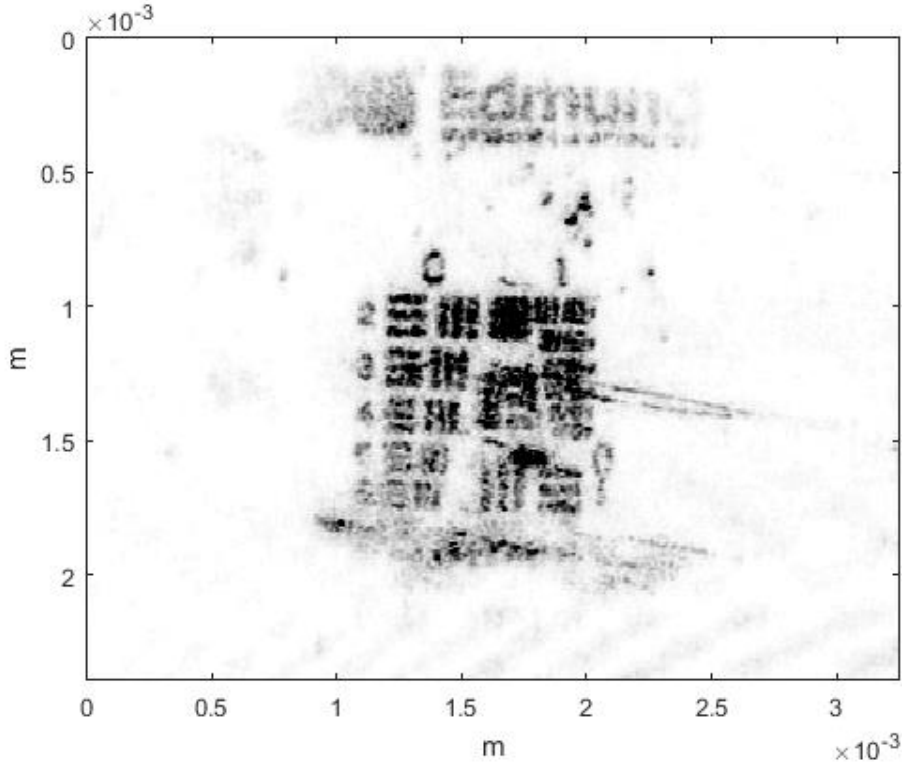

Figure 35 Speckle average of single aperture. This shows the initial resolution that we achieved with the 750mm lens system. The group that we can see is the 0-1 element with 1-1 element being questionable.

Telephoto at 5m

After building and taking some initial images with the telephoto lens, I realized there was quite a bit of spherical aberration present. This was confirmed by the calculations in Zemax as discussed in Chapter 5. This, combined with the issue with LO curvature, made it very difficult to correct the image (see Figure 36). The images that

resulted from the system had no feature that could be well resolved. This system was not used for further experiments.

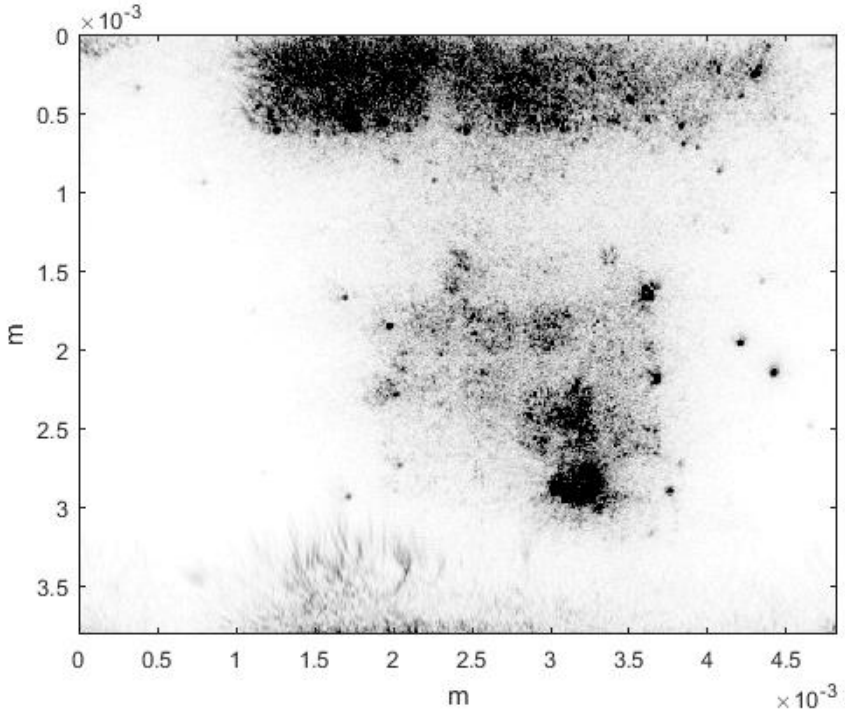

Figure 36 Telephoto 5 m single aperture speckle average, which shows that the image is not useable for analysis since nothing can be clearly seen.

500-mm lens at 5m

The 500-mm focal length lens system had two setups, with the only change being the LO location. The two LO locations were at 13 cm and 54 cm away from the camera. The LO position at 54 cm from the camera produced the best results. This was due to the smaller curvature of the LO beam. Since the images (see Figure 37) at 13 cm LO distance were unusable, I did not use this setup to collect data for the turbulence and phase gradient reconstruction case. These two previous cases could be potentially

corrected in post processing, but I ran out of time to develop a correction procedure. This would have involved correcting for spherical and other aberrations, not just due to the lenses but also due to the curvature of the wave front of our LO. These aberrations would most likely be higher order as well, needing a more complex correcting equation than previously discussed. This correction then would be applied to the turbulent data set and then reconstruction would be applied after wards.

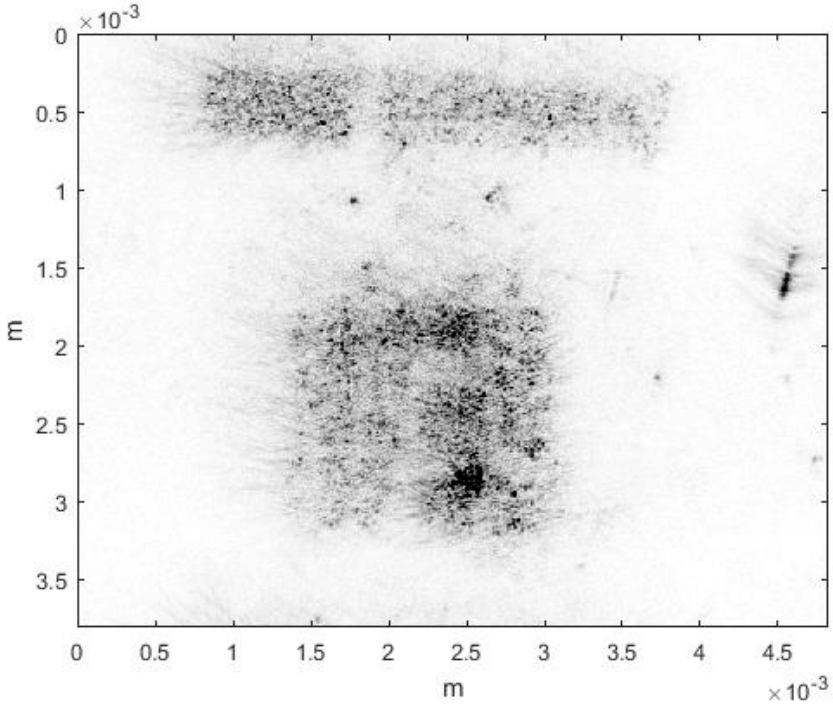

Figure 37 500mm single aperture with speckle averaged LO at 13 cm

The long-distance LO produced the best image quality of the different lens setups that we tried. This points to a very important factor of building a holographic image capturing system. The LO needs to be as close to a plane wave as possible to make the

processing straight forward. In this case, the last group that is resolvable in the USAF resolution target was the 0-5 group (see Figure).

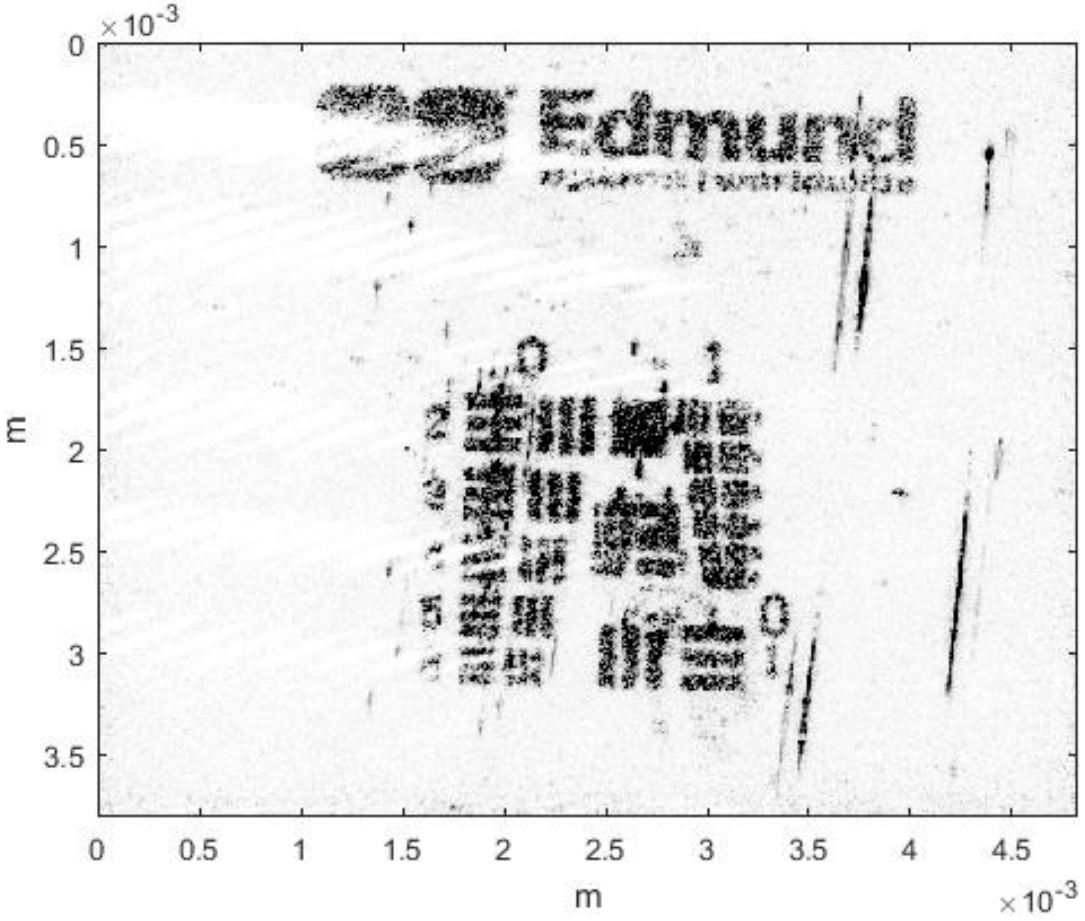

Figure 39 500-mm single aperture, speckle averaged with the LO distance at 54 cm showing the affect that the LO's wave curvature can have on the final hologram image quality.

Phase Gradient Reconstruction Method

The major difference in the processing of the turbulent case was the phase gradient reconstruction. This was done by using the method described in Chapter 4. Upon first attempt of the reconstruction, I ran into a problem. There was an overlay on the

image. This overlay was the image inversed about both axes (see Figure 38). This was not usable for HAL images, so the problem needed to be remedied.

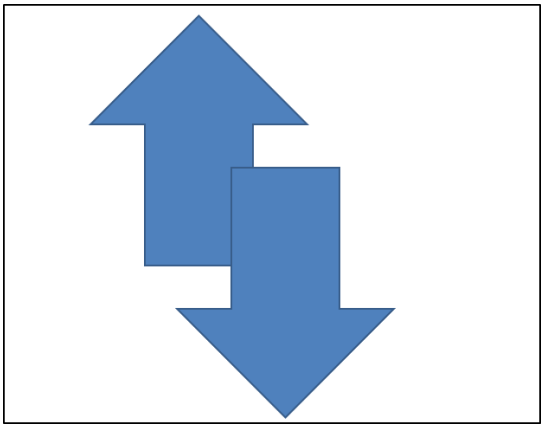

Figure 38 Image overlap issue that was occurring during the initial attempts at reconstruction. This was where an inverted flipped image was showing up after reconstruction.

Figure 39 shows a flow chart for the complete process for doing the phase gradient averaging. I first had to align the images by using correlation since the turbulence added tip-tilt aberration that moved the image around on the sensor. I then found that to correct the issue of the overlap I needed to create an image similar to the Fourier Plane before doing the reconstruction. I took the image from the hologram and placed it in a quadrant of a large grid. Then I performed a forward Fourier transform and then processed the Fourier plane data as we have previously described in the initial discussion of the method in Chapter 4, in the section titled “Taking the Gradient.” This was done by calculating the angle of the Fourier plane and then calculating the gradient, which resulted in the phase gradient. I took the phase gradient of 30 images for that aperture position. The phase gradients were then averaged over the set for that position.

The construction was done using the intgrad2 function, which constructed the phase component for the aperture, which was then added to the averaged intensity of that aperture position. Finally, an inverse Fourier transform on the Fourier plane produced an image with two copies of the image in each quadrant. I used the top quadrant image for the process of making the apertures to be used in HAL (see Figure 39).

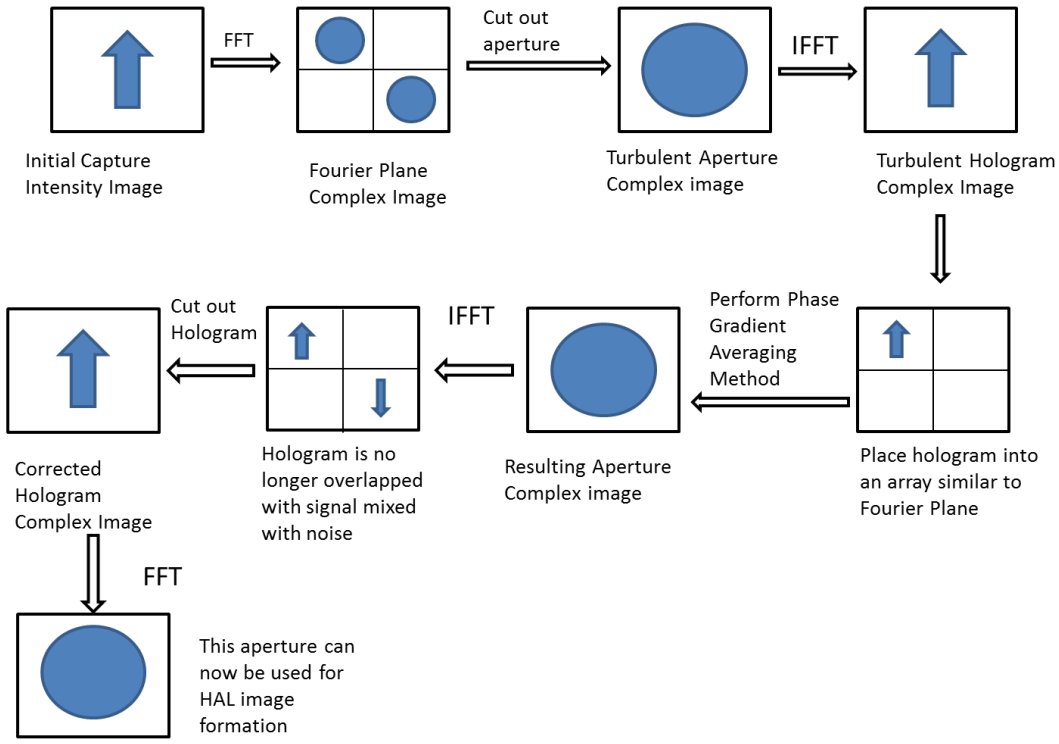

Figure 39 The flow chart of how to do the processing for phase gradient averaging method. This removes the issues of the overlapping holograms after reconstruction.

Testing Phase Gradient Averaging

As part of the phase gradient averaging process, a question to investigate is: exactly how many averages are needed and how to tell if there was actual improvement in the image. As can be seen in Figure speckle can make it difficult to pick out details

visually. This led me to developing a standard of looking at the normalized intensity cross-section of a line element. The cross-section I chose to look at was the area around the 0-1 element of the vertical lines, which measured resolution in the horizontal direction. Figures 42-44 show the cross-section and include a box of the area that was sampled for the cross-section. The contrast between the dark and light lines provide a measure of our resolution, with a larger contrast indicating a better resolution. All the cross sections had extra space along their outside edges to try to show what the background noise was.

To test the amount of averaging needed, I examined 10 averages, 30 averages and 100 averages. This showed a good range of what improvement was possible. To compare the tests, I looked at the cross section of the 0-1 element on the resolution chart. The x-axis on the cross-section graphs was the dimension in image space.

To test the method, I used the reconstruction of the turbulence case for the imaging system with 500 mm focal length and the LO located at 54 cm. I tested the averages of 10, 30, 100 turbulent reconstructions. Looking at the averaged image, it was hard to tell if there was improvement from the cross-sections, except that maybe 30 and 100 were better than 10, but the difference between 30 and 100 was difficult see. In the trade-off of time versus quality we chose to use the 30 averages for the creation of the HAL images. This was done to give us a good chance of being able to create HAL images through correlation and to not have to use a super computer to process all the images.

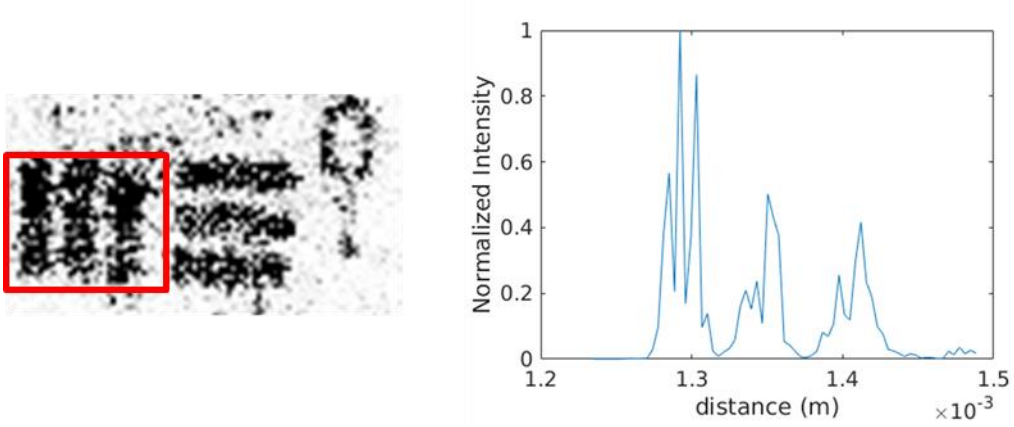

Figure 40 The 10 averages image (left) with the cross section (right), the general region used enclosed in the red box. We can see from the cross-section that there is a lot of speckle still present due to there not being on consistent peak for each line

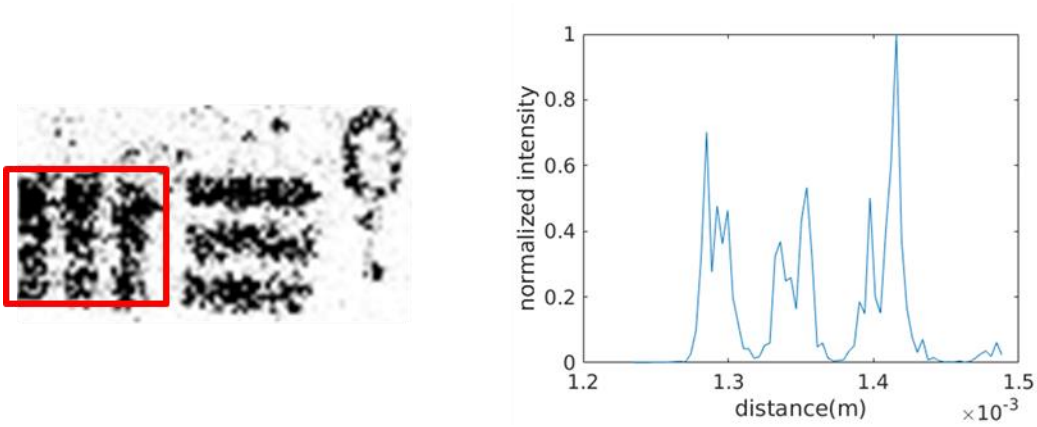

Figure 41 The 30 averages image (left) with the cross-section (right), the lines are becoming less affected by the speckle with the difference between gap minimum and maximum of the least intense line has improved.

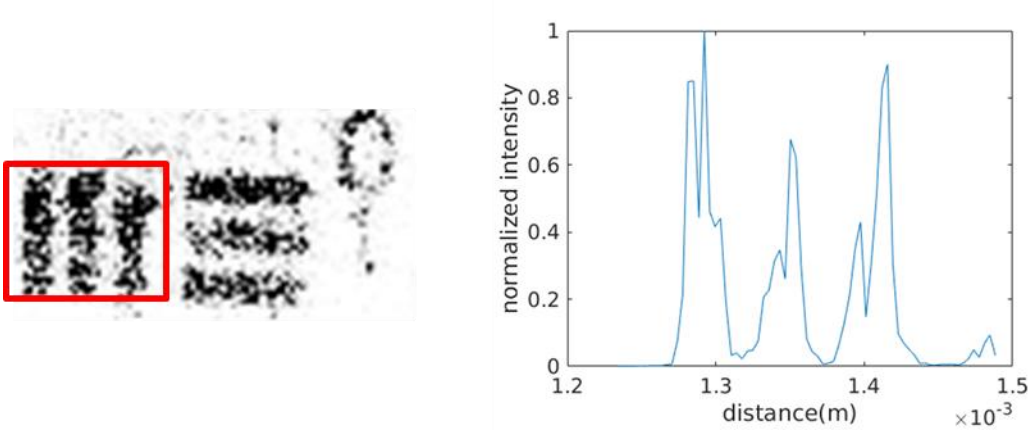

Figure 42 The 100 averages image (left) and the cross-section (left), the cross-section shows the improvement to the contrast between the line and the group. The speckle averaging is starting to show with the lines having less dips in them due to speckle. The speckle average is also the reason it looks like the gap is filling in.

Phase Gradient Averaging Data

To be able to examine how well the phase gradient averaging method worked we examined two cases: one with no turbulence and one with turbulence. The no turbulence case was to establish a base line for comparison for the reconstruction. The turbulence case was to test the phase gradient averaging method and to see if it was possible to return to the no-turbulence case. All the images that are to follow were taken using the 500-mm focal length system that was 5 m away from the target.

The no-turbulence case resulted in showing what I wanted to return to with the reconstruction method. Figure 43 shows the cross section of the 0-6 element with the cross section being an average of the rows of that area in the red box. This will establish the base line for our reconstruction.

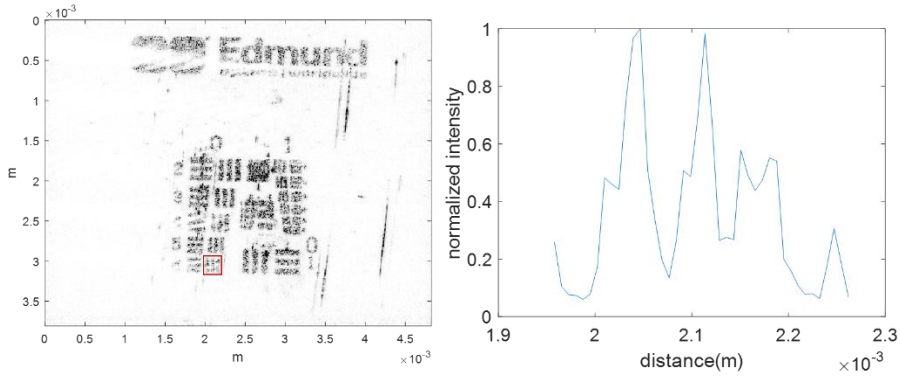

Figure 43 The cross-section and image of the no turbulence case

The most notable feature of the turbulence cases shown in Figure 44 was t the increased amount of background noise. This led to degraded contrast between the line groups. The line element for the turbulence case looked similar to the no-turbulence case. This points to an issue of the way that I was trying to simulate turbulence was not as strong as I thought it would be. The reconstruction seems to have made the line shape become worse, though the reconstruction did improve the contrast.

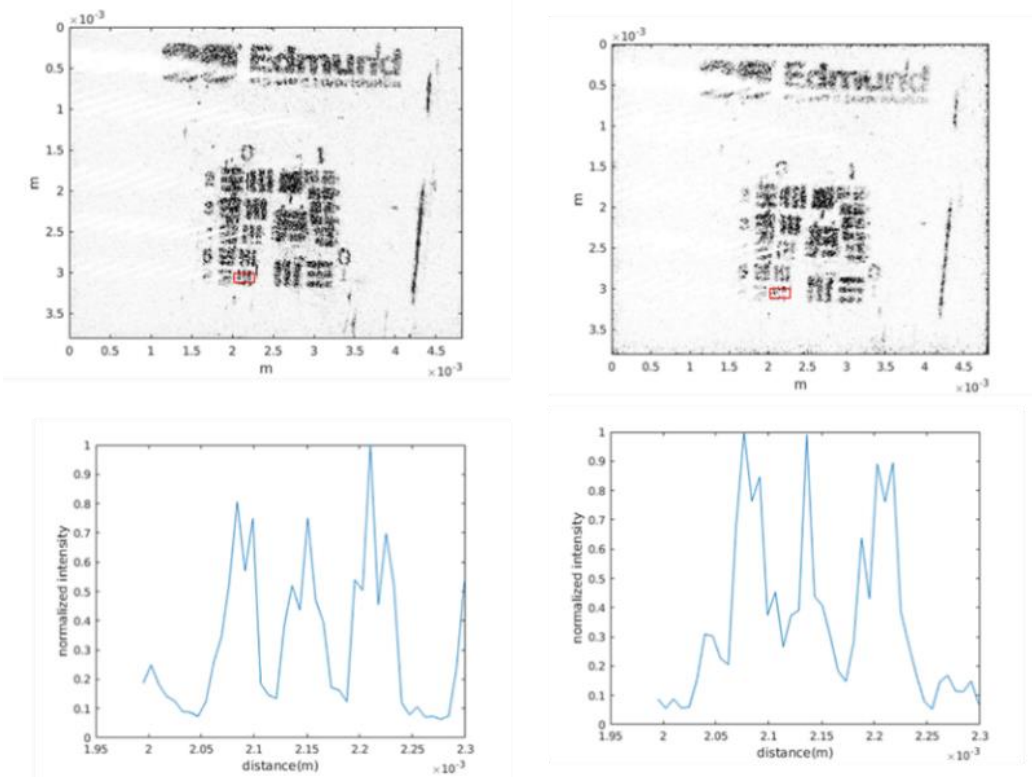

Figure 44 The non-reconstructed turbulence image (left) and the reconstructed image (right), The background noise is the most noticeable difference with the reconstruction seeming to get this to be less of a prominent feature in the image.

HOLOGAPHIC APERTURE LIDAR IMAGING EXPERIMENT

HAL Imaging Process

The process of getting a HAL image was the same for non-turbulent and turbulent cases, except the latter used the phase gradient averaging reconstruction. To create HAL images, I needed to correlate the apertures from different rotation positions together. The correlation occurred between apertures that are overlapping each other. If the apertures did not overlap, they were not correlated to each other and could not be used to produce an offset, as discussed previously.

The HAL image was created by using the offset and phase to add the pupils together coherently. To make the HAL image, I needed a way to add the pupils together. This was done by adding the pupils into another larger grid, called the HAL mosaic. The position of the apertures in the mosaic was the offset of the correlation point. The mosaic was inverse Fourier transformed back to the image plane and then speckle averaged with other HAL sets of the same target. A visual map of the process can be seen in Figure 45.

The expected resolution increase was dependent on how much larger than the original pupil size the mosaic was. In this experiment, I tried to make the mosaic three times the original aperture size.

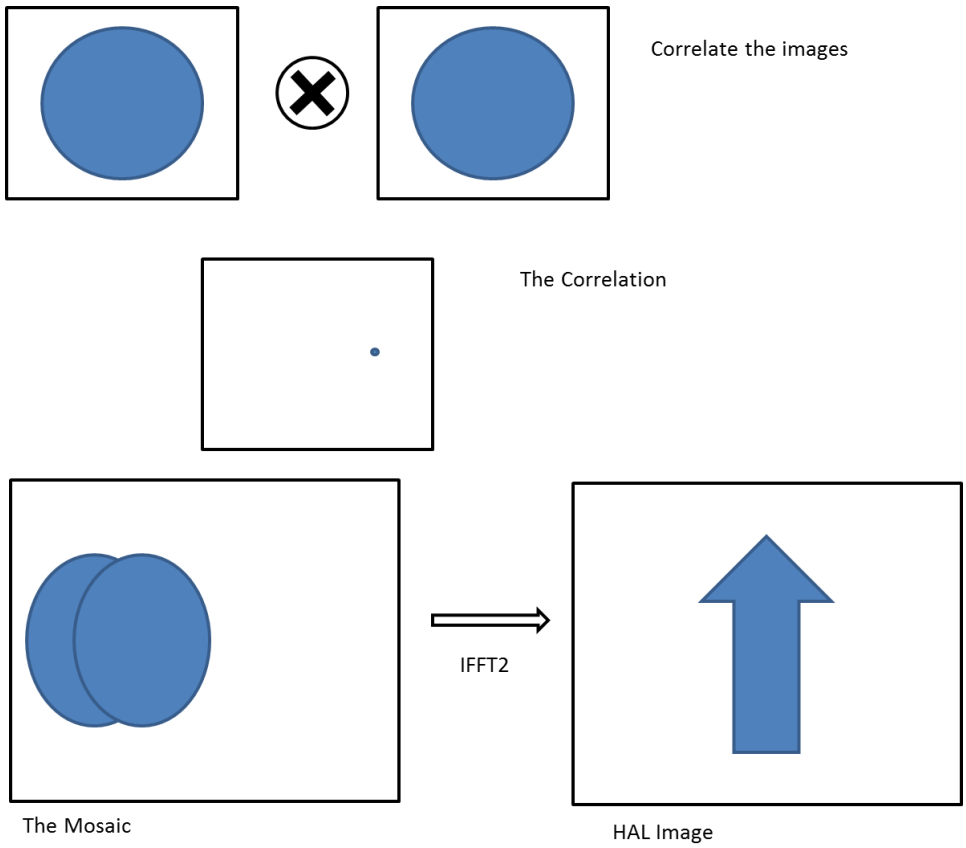

Figure 45 HAL processing flow chart with a representation of what each stage looks like visually.

To bring all the information on how to form a HAL image altogether into one place I have created a list of steps as follows:

1. Capture intensity image of target with LO also being captured onto the camera. If multiple captures of the same position are desired then capture these now.
2. Rotate the target using the rotation stage and controller.
3. Repeat steps 1 and 2 until desired number of positions have been captured.
4. For each intensity image:
    a. Perform a Fourier transform to get to the pupil plane.

   b. Cut out the quadrant containing the pupil desired. Either quadrant will work, but signs of phase corrections will change as they are complex conjugates.
   c. Apply a mask to zero out background and constrict pupil to desired size.
   d. Apply phase corrections to pupil for imaging system aberrations.
5. If not performing the Phase Gradient Averaging skip to step 8
6. Inverse Fourier transform the pupils for one position back to the image plane and register the images and remove tip-tilt error by:
   a. Using correlation to find the offset from the first image,
   b. Shift the images so they are registered together,
   c. Fourier transform all images back to the pupil plane.
7. For each rotation position:
   a. Take the gradient of the phase of all the pupils by using the angle and gradient functions in MATLAB.
   b. Average the phase gradients of the pupils.
   c. Reconstruct the averaged phase gradient using intgrad2 reconstructor.
   d. Average the pupil intensities.
   e. Combine the averaged intensity and the averaged phase back together in the form of $A_{averaged}e^{i\varphi_{averaged}}$.
8. Form the mosaic of pupils by:
   a. Correlate the pupil of the next rotation position with the one previous.

b. Find offset of correlation by finding the max value.

c. Use the offset to place pupils into mosaic. The mosaic is a coherently summed together pupils of rotated positions.

d. Repeat for all rotation positions.

9. To improve image quality one can incoherently sum non-correlated HAL images (i.e. non-overlapping rotation positions) to speckle average.

One issue that came up sometimes is that the center of the correlation represents the correlation of the circular aperture out of the pupil, and sometimes this can be larger than that from the overlapping speckle in the aperture which is the correlation used to calculate the offset. The point of interest is the one encircled by red in Figure 46.

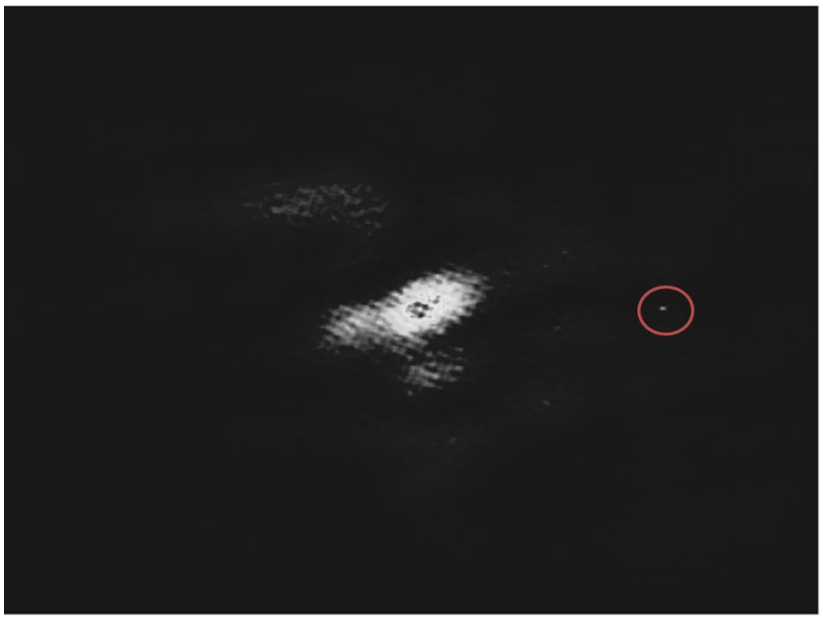

Figure 46 The correlation between two pupils. The offset is the bright point contained in the red circle. This offset from center was used to place the pupil in the mosaic.

The resulting mosaic for one set of the 750-mm system's mosaic can be seen in Figure 49. It should be noted that a correction to the pupils' amplitude must be applied since certain sections were brighter than others due to them being combined in an additive process (see Figure). If the correction was not applied, streaks appeared in the final HAL image.

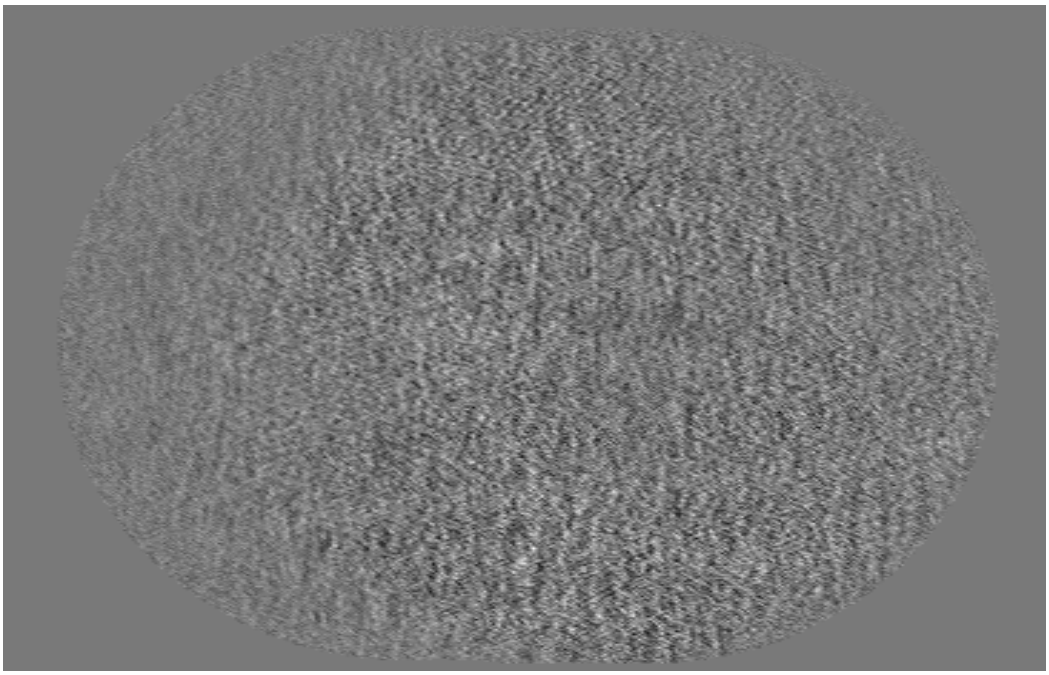

Figure 49 The mosaic of a HAL image after it has its intensity normalized.

Another issue was that when we looked at any image, it had some assigned scale to its color. In traditional photography, this was dictated by the chemical reaction that happened between the film and the incident light, but also how the image prints were processed played a huge part in this scaling. In the digital age, it is the sensor picking up photons that acts like the film but the making of image prints is what can be done in post processing. This scaling in software lets us change the scale to fit each set of data

independently, but can make it difficult to compare two images visually next to each other (see Figure 47).

Another problem I ran into was that the rotation stage stopped rotating partway through a capture. The rotation stage stopped rotating after 30 steps, but kept telling the computer that it had moved. This led to only having really only one and a half HAL image. This was for the 500-mm focal length imaging system.

750-mm lens at 12 m

The 750-mm system was my first attempt at building a HAL system. The image in Figure 49 shows that I was able to build a system that could produce HAL images. There is a clear improvement in the HAL image over the single-aperture image, clearly visible in the 1-1 element becoming resolved in the HAL.

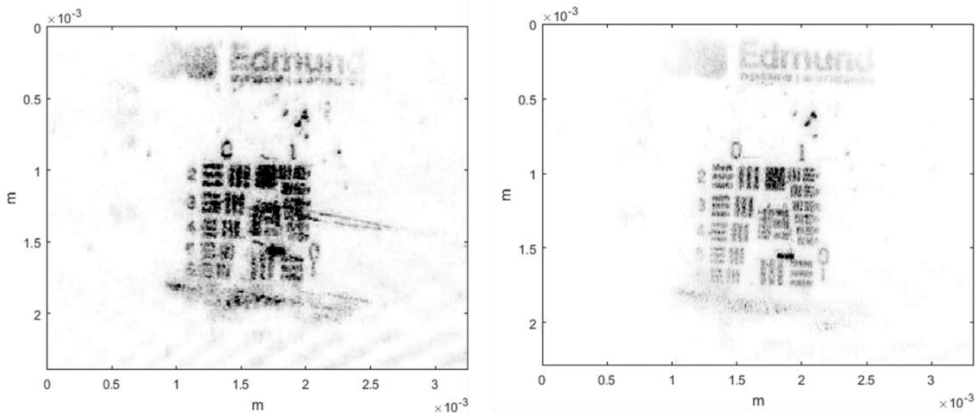

Figure 47 The single aperture image (left) and the HAL image (right) for the 750-mm lens, 12 m range experiment. We can see that the 1-1 element has become resolved in the HAL image while it is not in the single aperture. This visual example show how synthetic aperture can be used to increase the resolution of images.

500 mm 5 m

The turbulence case (see Figure 48) looks better than the single aperture case but I was unable to say by visual inspection if the turbulence-corrected HAL image had been returned to the same level of resolution as the no turbulence experiment. The comparison required further analysis to decide if there had been improvement.

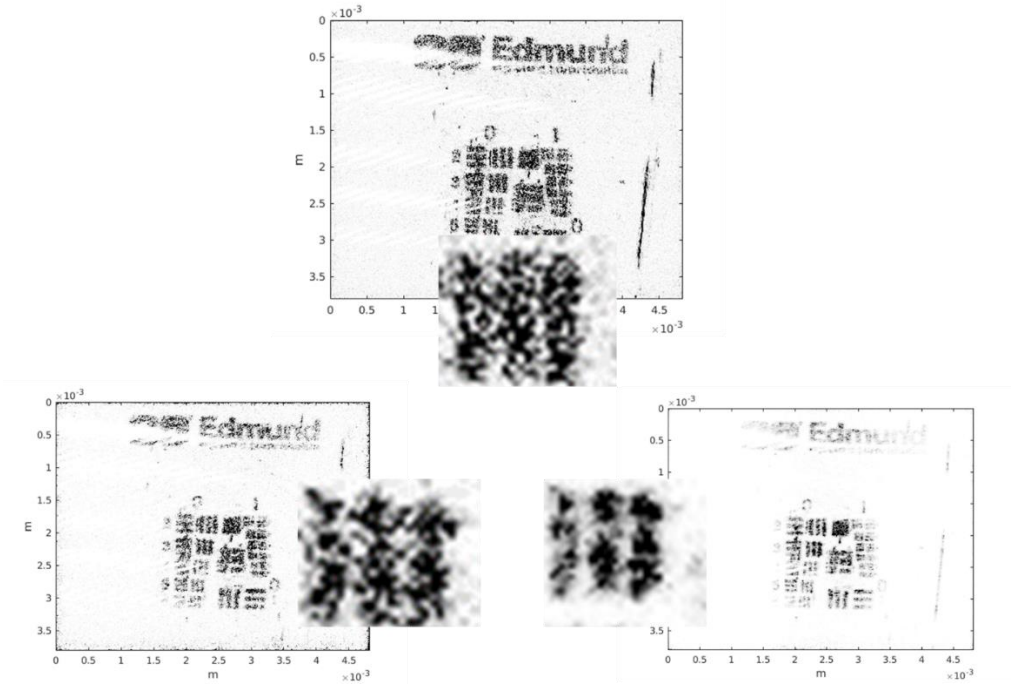

Figure 48 The 500mm turbulence data set with the non-reconstructed image (top), single aperture reconstructed image (left) and the HAL image (right). The 0-3 line element has been cropped and enlarged so we can see the difference in the contrast that each step provided.

When I looked at the comparison between the uncorrected HAL image of high turbulence and the corrected image, I only saw a small improvement to the shape of the lines of the 0-6 element (Figure 49). The contrast had a very minimal improvement, to the point of almost being negligible. This might mean that we needed more averages for the turbulence case. This need for more averages seems also to be true when comparing

the non-turbulent and the turbulence HAL image. This result points to the method working but needing more study to improve.

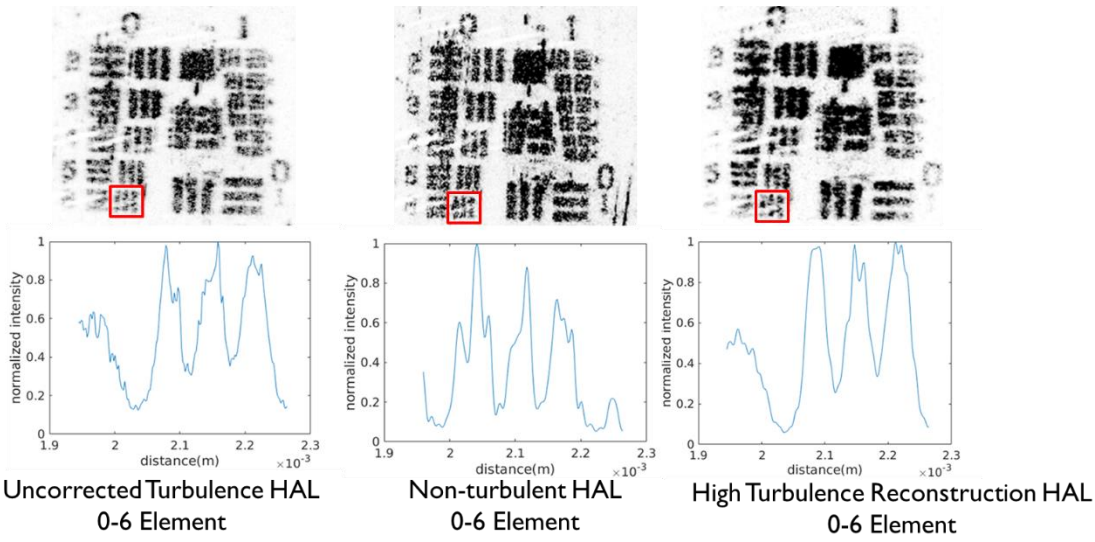

Figure 49 The uncorrected HAL image and cross-section of 0-6 element(left), the non-turbulent HAL image and cross-section of element 0-6 (center) and the turbulence HAL image and cross-section of element 0-6 (right). The improvement from the uncorrected to the corrected state can be seen to be a slight improvement in shape and a very slight change in the contrast.

DISCUSSION OF RESULTS AND FUTURE RESEARCH

LO Location

Looking at all the data sets that we collected, the LO location appears to make a huge difference to the overall image quality, as seen in the 500-mm lens system case. The close LO had a more curved wavefront, which led to a more defocused final image, while the LO at long distance had much clearer images. This all points directly to one improvement that could be made for future study is moving the LO a long distance back or processing to take out LO curvature. With improved image quality by having flatter LO wavefronts, it would be possible to make a better judgment on this method for use in synthetic aperture lidar techniques.

Another point that should be emphasized is that it should be possible to remove the spherical and any other aberration that occur to the wavefront through digital means. This was not completed for this project due to a shortness of time and because it was outside the scope of this study. A robust error correction algorithm would be an excellent path for this research to be continued at a later date. This program would probably make use of the sharpness metric (see Eq.2.36) and apply more of the aberration theory that is used in optical design algorithms.

Improvement of Image Through Phase Gradient Averaging

By looking at the results that were obtained it is easy to see that the reconstruction works well currently under low turbulence cases, with the line elements being resolved to

a non-turbulent level after reconstruction. This shows that the phase gradient averaging method can work at least for weak turbulence.

The high-turbulence case only showed slight improvement after reconstruction but it was not as good as the low-turbulence case. This might point to a need for a different number of averages to be used under different circumstances when capturing holograms. Also, it shows that the way we tried to simulate turbulence in the lab needs some improvement, as it seems to have been mostly in piston and tilt phase change versus the higher-order phase changes we were expecting.

In both cases, we were able to construct HAL images from the reconstructed pupils, showing that we were able to retrieve the phase information that had been contained in the signal beam prior to the interference. This shows that we were able to complete the main goals of this study.

Continued Work

Now that empirical proof has been shown that this method can correct for the error caused by turbulence, more study should be placed on the theoretical explanation of why it worked. One main area is optimizing the number of averages since as discussed in Chapter 6, I went with what seemed quick and achieved slightly improved results. A lot more organized thought could be put into how many averages are needed to achieve optimal results without unnecessary processing of extra images.

Code optimization needs help across the whole project in regard to speed. This is especially true if taking more averages is desired. Also, this could help with the aberration correction, which should be able to correct for all the errors in the imaging

system. Another route that could be taken is using the image processing toolbox of MATLAB with Zernike polynomials to correct for aberrations. This could be used with the sharpness metric to improve the image contrast. This could be done by looking for a maximum of the sharpness and using the Zernike polynomials as the correcting coefficients. We ran out of time to get this piece of code to work exactly, thus abandoning some system results that we had hoped to collect.

Different imaging systems could be developed for further study. For proving that the technique had merit, I went with simple systems to not overcomplicate the errors that could arise. Now that this method has been shown to work, more complex imaging systems could be developed to try and increase the resolution even further.

Conclusion

This thesis has covered the necessary background for a person to understand how to collect and use holograms for synthetic aperture lidar. This background covered the Fourier optics, which included discussion of the wave picture of light, and also the two ways of simplifying diffraction mathematically, these being Fresnel and Fraunhofer approximations. This background led us to the discussion of how holography works and how the Fourier transforms is fundamental to imaging and holography.

The use of optical systems in the capture of hologram has been discussed with various aspects of the design, looking at aberration in a telephoto design and a brief discussion of how to deal with these concerns. The other first-order system properties of interest were discussed, with a focus on the magnification and getting the image to fill the view of the sensor.

Experimental data were looked at with a discussion of why major differences in certain setups might be present. With the LO distance from the camera being one of the major differences found, with some systems seriously underperformed due to this issue. I was able to show a small increase in resolution with the phase gradient averaging technique. This method should be looked at more closely for further analysis to see if better results can be achieved.

REFERENCES CITED

[1] Riesland, David Walter, “Infrared cloud imaging systems characterization,” Montana State University, Bozeman, Montana, 2016.

[2] Southern African Large Telescope, Wikimedia:https://commons.wikimedia.org/wiki/File:Southern_African_Large_Telescope_720x576p.jpg, 2017

[3] Orion. Orion http://orion.scene7.com/is/image/Orion/340x340_template?$340x340$&$prod=is{Orion//09883?scl-=}, 2017

[4] L. J. Peter D. Nellist, “STEM Image Post-processing for Instability and Aberration Correction for Transfer Function Extension,” *Journal of Physics Conference Series*, pp. 1–4, 2012.

[5] Murakami, U-ichiro, *Hot road mirage “fake water.”* . Wikimedia: https://commons.wikimedia.org/w/index.php?curid=11630764

[6] Samuel M. Venable, Bradley D. Duncan, Matthew P. Dierking, and David J. Rabb, “Demonstrated resolution enhancement capability of a stripmap holographic aperture ladar system,” *Applied Optics*, vol. 51, no. 22, pp. 5531–5542, 2012.

[7] Jason W. Stafford, David J. Rabb, and Bradley D. Duncan, “Holographic aperture ladar with range compression,” *Journal of Optics Society of America*, pp. A1–A9, 2017.

[8] W. M. Brown and L. J. Porcello, “An Introduction to Synthetic Aperture Radar,” *IEEE Spectrum*, pp. 52–56, 1969.

[9] Jean-Luc Starck, Fionn Murtagh, and Jalal M. Fadili, *Sparse image and signal processing: wavelets, curvelets, morphological diversity*. Cambridge: Cambridge university press, 2010.

[10] J. C. Marron *et al.*, “Extended-range digital holographic imaging,” in *Proc. SPIE 7684, Laser Radar Technology and Applications*, Orlando Florida, 2010, vol. 15, p. 76841J.

[11] J. C. Marron, R. L. Kendrick, N. Seldomridge, T. D. Grow, and T. A. Höft, “Atmospheric turbulence correction using digital holographic detection: experimental results,” *Optics Express*, pp. 11638–11651, 2009.

[12] J.W. Goodman, *Statistical Optics*. Wiley Classics Library, 2000.

[13] T.S. Lewis and H.S. Hutchins, “A synthetic aperture at optical frequencies,” *Proc. IEEE*, pp. 587–588, 1970.

[14] B.E.A. Saleh and M.C. Teich, *Fundamentals of Photonics,* Second Edition. John wiley & sons Inc, 2007.

[15] Joseph W. Goodman, *Introduction to Fourier Optics,* Third. Roberts & Company Publishers, 2005.

[16] Jason D. Schmidt, in *Numerical Simulation of Optical Wave Propagation with example in MATLAB,* SPIE, 2010, p. 23.

[17] Jason D. Schmidt, in *Numerical Simulation of Optical Wave Propagation with example in MATLAB,* SPIE, 2010, p. 153.

[18] Abbie E. Tippie and James R. Fienup, "Phase-error correction for multiple planes using a sharpness metric," *OPTICS LETTERS,* vol. 34, no. 5, pp. 701–703, 2009.

[19] M. Liebling and M. Unser, "Autofocus for digital Fresnel holograms by use of a Fresnelet-sparsity criterion," *Journal of the Optical Society of America A,* vol. 21, no. 12, pp. 2424–2430, 2004.

[20] Jason D. Schmidt, *Numerical Simulation of Optical Wave Propagation with example in MATLAB.* SPIE, 2010.

[21] Jason D. Schmidt, in *Numerical Simulation of Optical Wave Propagation with example in MATLAB,* SPIE, 2010, p. 95-102.

[22] Jason D. Schmidt, in *Numerical Simulation of Optical Wave Propagation with example in MATLAB,* SPIE, 2010, p. 149-184.

[23] Samuel M. Venable, Bradley D. Duncan, Matthew P. Dierking, and David J. Rabb, "Demonstrated resolution enhancement capability of a stripmap holographic aperture ladar system," *Applied Optics,* vol. 51, no. 22, pp. 5531–5542, 2012.

[24] Abbie E. Tippie and James R. Fienup, "Multiple-plane anisoplanatic phase correction in a laboratory digital holography experiment," *Optics Letters,* vol. 35, no. 19, pp. 3291–3293, 2010.

[25] Abbie E. Tippie, Abhishek Kumar, and James R. Fienup, "High-resolution synthetic-aperture digital holography with digital phase and pupil correction," *Optics Express,* vol. 19, no. 13, pp. 12027–12038, 2011.

[26] Abbie E. Tippie, "Aberration Correction in Digital Holography," University of Rochester, Rochester, New York, 2012. https://urresearch.rochester.edu/

[27] P. H. Eichel, D.C. Ghiglia, and C. V. Jackowatz Jr, “Speckle Processing Method for Synthetic Aperture Radar Phase Correction,” *Optics Letters*, pp. 1101–1103, 1989.

[28] C.V. Jackowatz and P.H. Eichel, “Phase-gradient algorithm as an optimal estimator of the phase derivative,” *Optics Letters*, pp. 1101–1103, 1989

[29] Dennis C. Ghiglia and Louis A. Romero, “Direct phase estimation from phase differences using fast elliptic partial differential equation solvers,” *OPTICS LETTERS*, vol. 14, no. 20, pp. 1107–1109, Oct. 1989.

[30] Stephen Crouch, “Synthetic Aperture Ladar Techniques,” Montana State University, Bozeman, Montana, http://scholarworks.montana.edu/xmlui/bitstream/handle/1/1125/CrouchS0812.pdf?sequence=1.

[31] Stephen Crouch and Zeb Barber, “Laboratory demonstrations of interferometric and spotlight synthetic aperture ladar techniques,” *OPTICS EXPRESS*, vol. Vol. 20, no. No. 22.

[32] Vasily A Matkivsky, Valentine M Gelikonov, Alexander A Moiseev, Grigory V Gelikonov, Dmitry V Shabanov, and Pavel A Shilyagin, “Correction of aberrations in digital holography using the phase gradient autofocus technique,” *Laser Physics Letter*, vol. 13, no. 3, p. 35601.

[33] D’Errico, John, *Inverse Integrated gradient*. Retrieved from Mathworks: http://www.mathworks.com/matlabcentral/fileexchange/9734-inverse--integrated--gradient

[34] Ioana S. Sevcenco, Peter J. Hampton, and Panajotis Agathoklis, “A wavelet based method for image reconstruction from gradient data with applications,” *Multidimensional Systems and Signal Processing*, vol. 26, no. 3, pp. 717–737, Jul. 2015.

[35] telescopes.net. (2017, January 15). Flea 3 camera . Retrieved from telescopes.net: https://telescopes.net/store/media/extendware/ewimageopt/media/inline/9a/6/point-grey-flea3-32mp-monochrome-usb-30-sony-imx036-camera-ab5.jpg

[36] Edmund optics. USAF resolution chart found on the back of Edmunds Optics target case R228779-11287.

[37] Edmund Optics.). USAF Resolution Target. Retrieved from Edmundoptics.com: https://www.edmundoptics.com/contentassets/c1499ee3006340bab8c768b7e799cdea/fig-9-imtf.jpg

[38] Reliant systems inc. RSR100. Retrieved from reliantsystemsinc.com: http://reliantsystemsinc.com/data/uploads/products/rsr100.jpg

APPENDICES – CODE

APPENDIX A

C++ SOFTWARE CONTROLLER

C++ Controller

```
// Library includes and header files
#include "FlyCapture2.h"
#include <fstream>
#include <opencv2/core/core.hpp>
#include <opencv2/highgui/highgui.hpp>
#include "opencv2/imgproc/imgproc.hpp"
#include <iostream>
#include <boost/date_time/posix_time/posix_time.hpp>
#include <ctime>
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <vector>
// Rotation stage includes
#include <x_examples.h>
#include <iomanip>
// Tic Toc includes
#include <stack>
#include <ctime>


using namespace FlyCapture2;
using namespace cv;
using namespace std;

std::stack<clock_t>tictoc_stack;
void tic()
{ tictoc_stack.push(clock());}
void toc()
{
	std::cout << "n\ Time Elapsed:"
		<< ((double)(clock()-tictoc_stack.top())/CLOCKS_PER_SEC);

	tictoc_stack.pop();
}




std::string path()
{
	std::string buff;
```

```
                buff="D:\\Turbulance data BB\\";
                return buff;
}


int main()
{
   Error error;
   Camera camera;
   CameraInfo camInfo;

        // Setup file name
        string filename;
        int numbi;
        cout << "Name of the file for capture : ";
        cin >> filename;
        cout<< '\n' << "Number of images to capture pr turn : ";
        cin >> numbi;
        int numbturns;
        int turns;
        turns=1;
        cout << '\n'<< "Numbr of 500 click turns for this set : ";
        cin >> numbturns;
        string filename2;
        filename2=filename;
        string filename3;
        filename3=filename;
        filename3.append("dft.png");
        filename.append(".bin");
        filename2.append(".png");

        string filepath;
        filepath = path() + filename;
        string filepath3;
        filepath3=path()+filename3;
        int fr;
        cout<< '\n' <<" Expsoure time in ms \(120 fps is 8.3\) : ";
        cin >> fr;

        int count=1;
        cout<<'\n'<<" Remeber to start capture hit c and to stop hit s"<<'\n';
```

```
// Connect the camera
error = camera.Connect( 0 );
if ( error != PGRERROR_OK )
{
   std::cout << "Failed to connect to camera" << std::endl;
   return false;
}

// Get the camera info and print it out
error = camera.GetCameraInfo( &camInfo );
if ( error != PGRERROR_OK )
{
   std::cout << "Failed to get camera info from camera" << std::endl;
   return false;
}
std::cout << camInfo.vendorName << " "
         << camInfo.modelName << " "
         << camInfo.serialNumber << std::endl;
       // Set Camera configurations
       //Gain Setting in dB
       Property prop;
       prop.type=GAIN;
       prop.autoManualMode=false;
       prop.absControl=true;
       prop.absValue=1;
       error=camera.SetProperty(&prop);

       // Shutter speed in ms
       Property shut;
       shut.type=SHUTTER;
       shut.autoManualMode=false;
       shut.absControl=true;
       shut.absValue=fr;
       error=camera.SetProperty(&shut);

       //Expoisure in eV
       Property expo;
       expo.type = AUTO_EXPOSURE;
       expo.onOff = false;
       expo.autoManualMode = false;
       expo.absControl = true;
       expo.absValue = -3.5;
       error = camera.SetProperty( &expo );
```

```
//Brightness
Property proa;
proa.type = BRIGHTNESS;
proa.onOff=false;
proa.absControl = true;
proa.absValue = 0.5;
error = camera.SetProperty( &proa );

// Shaprness
Property prob;
prob.type = SHARPNESS;
prob.onOff = false;
prob.autoManualMode = false;
prob.valueA = 1500;
error = camera.SetProperty( &prob );

// Gamma
Property proc;
proc.type = GAMMA;
proc.onOff = false;
proc.absControl = true;
proc.absValue = 1.5;
error = camera.SetProperty( &proc );

// White Balance
Property prod;
prod.type = WHITE_BALANCE;
prod.onOff = false;
prod.autoManualMode = false;
//Set the white balance red channel to 500.
prod.valueA = 500;
//Set the white balance blue channel to 850.
prod.valueB = 850;
error = camera.SetProperty( &prod );


// Setting up the rotation stage
int rc = GALIL_EXAMPLE_OK; //return code
char buf[1024]; //traffic buffer

GCon g = 0; //var used to refer to a unique connection. A valid connection is
nonzero
   x_e(GVersion(buf, sizeof(buf))); //library version
```

```
                cout << "Library version: " << buf << "\n\n";
                //Basic connections
                x_e(GOpen("10.10.10.2 --subscribe ALL --direct", &g)); //connect and
assign a value to g.

                        char* trimmed; //trimmed string pointer



   error = camera.StartCapture();
   if ( error == PGRERROR_ISOCH_BANDWIDTH_EXCEEDED )
   {
      std::cout << "Bandwidth exceeded" << std::endl;
      return false;
   }
   else if ( error != PGRERROR_OK )
   {
      std::cout << "Failed to start image capture" << std::endl;
      return false;
   }



   // capture loop
   char key = 0;
   while(key != 'q')
   {
// Get the image
      Image rawImage;

         rawImage.SetDimensions(1048,1328,1,FlyCapture2::PixelFormat::PIXEL_FOR
MAT_MONO8,FlyCapture2::BayerTileFormat::NONE);
      Error error = camera.RetrieveBuffer( &rawImage );
      if ( error != PGRERROR_OK )
      {
            std::cout << "capture error" << std::endl;
            continue;
      }

      // convert to rgb
     // convert to mono8
      Image monoImage;
      rawImage.Convert( FlyCapture2::PIXEL_FORMAT_MONO8, &monoImage );
```

```
                unsigned int rowBytes =
(double)monoImage.GetReceivedDataSize()/(double)monoImage.GetRows();

      cv::Mat image = cv::Mat(monoImage.GetRows(), monoImage.GetCols(), CV_8U,
monoImage.GetData(),rowBytes);
                // Histogram
                  int histSize = 256;
                float range[] = { 0, 256 } ;
                const float* histRange = { range };
                bool uniform = true; bool accumulate = false;
                Mat b_hist;
                calcHist( &image, 1, 0, Mat(), b_hist, 1, &histSize, &histRange, uniform,
accumulate );

                 // Draw the histograms for B, G and R
                int hist_w = 512; int hist_h = 400;
                int bin_w = cvRound( (double) hist_w/histSize );
                Mat histImage( hist_h, hist_w, CV_8U, Scalar( 0,0,0) );
                normalize(b_hist, b_hist, 0, histImage.rows, NORM_MINMAX, -1, Mat()
);
                for( int i = 1; i < histSize; i++ )
                {
                                                line( histImage, Point( bin_w*(i-1), hist_h -
cvRound(b_hist.at<float>(i-1)) ) ,
               Point( bin_w*(i), hist_h - cvRound(b_hist.at<float>(i)) ),
               Scalar( 255, 0, 0), 2, 8, 0  );

                }
 namedWindow("calcHist Demo", WINDOW_AUTOSIZE );
 imshow("calcHist Demo", histImage );

                // Use opencv to take the dft of the image
                // Padding is to help with the anti ailisaing so there is room for run off
                Mat padded;                     //expand input image to optimal size
                int m = getOptimalDFTSize( image.rows );
                int n = getOptimalDFTSize( image.cols ); // on the border add zero values
                copyMakeBorder(image, padded, 0, m - image.rows, 0, n - image.cols,
BORDER_CONSTANT, Scalar::all(0));

                Mat planes[] = {Mat_<float>(padded), Mat::zeros(padded.size(),
CV_32F)};
                Mat complexI;
                merge(planes, 2, complexI);        // Add to the expanded another plane
with zeros
```

```
                dft(complexI, complexI);            // this way the result may fit in the source
matrix

                // compute the magnitude and switch to logarithmic scale
                // => log(1 + sqrt(Re(DFT(I))^2 + Im(DFT(I))^2))
                // Switching to the log scale allows for finer details to be shown so your dc
term does not dominate.
                split(complexI, planes);                 // planes[0] = Re(DFT(I), planes[1] =
Im(DFT(I))
                magnitude(planes[0], planes[1], planes[0]);// planes[0] = magnitude
                Mat magI = planes[0];

                magI += Scalar::all(1);                  // switch to logarithmic scale
                log(magI, magI);

                // crop the spectrum, if it has an odd number of rows or columns
                magI = magI(Rect(0, 0, magI.cols & -2, magI.rows & -2));

                // rearrange the quadrants of Fourier image  so that the origin is at the
image center
                // This is the fftshift comparision for the matlab code version
                int cx = magI.cols/2;
                int cy = magI.rows/2;

                Mat q0(magI, Rect(0, 0, cx, cy));   // Top-Left - Create a ROI per quadrant
                Mat q1(magI, Rect(cx, 0, cx, cy));  // Top-Right
                Mat q2(magI, Rect(0, cy, cx, cy));  // Bottom-Left
                Mat q3(magI, Rect(cx, cy, cx, cy)); // Bottom-Right

                Mat tmp;                            // swap quadrants (Top-Left with Bottom-Right)
                q0.copyTo(tmp);
                q3.copyTo(q0);
                tmp.copyTo(q3);

                q1.copyTo(tmp);                     // swap quadrant (Top-Right with Bottom-
Left)
                q2.copyTo(q1);
                tmp.copyTo(q2);

                normalize(magI, magI, 0, 1, CV_MINMAX); // Transform the matrix with
float values into a
                                                    // viewable image form (float between values 0
and 1).
```

```
imshow("Input Image"       , image   );    // Show the result
imshow("spectrum magnitude", magI);



while(key=='c')
{
        cvDestroyAllWindows();

        while(turns<=numbturns){
        while (count<=numbi){


// Get the image

    Image rawImage;

      rawImage.SetDimensions(1048,1328,1,FlyCapture2::PixelFormat::PIXEL_FOR
MAT_MONO8,FlyCapture2::BayerTileFormat::NONE);
    Error error = camera.RetrieveBuffer( &rawImage );
    if ( error != PGRERROR_OK )
    {
        std::cout << "capture error" << std::endl;
        continue;
    }

    // convert to mono8
    Image monoImage;
    rawImage.Convert( FlyCapture2::PIXEL_FORMAT_MONO8, &monoImage );
            unsigned int rowBytes =
(double)monoImage.GetReceivedDataSize()/(double)monoImage.GetRows();

    cv::Mat image = cv::Mat(monoImage.GetRows(), monoImage.GetCols(), CV_8U,
monoImage.GetData(),rowBytes);



  // Open the file in write mode.

        FILE *pFile;
        pFile = fopen(filepath.c_str(), "a");
```

```
// Iterate through pixels.
for (int r = 0; r <image.rows; r++)
{
   for (int c = 0; c < image.cols; c++)
   {
      int pixel =image.at<uchar>(r,c);
                        fwrite(&pixel,sizeof(pixel),1,pFile);

   }

}

// Close the file.
fclose(pFile);



      //To record images just uncomment the section below
            //std::ostringstream filepath2;
            //filepath2<< path()<<count<<filename2;
            //
            //imwrite(filepath2.str(), image);
            //std::ostringstream filepathdft;
            //filepathdft<< path()<<count<<filename3;
            //
            //imwrite(filepathdft.str(), image);


            count=count+1;
                  }

      x_e(GCmd(g, "DP 0")); //define A position zero
      x_e(GCmdT(g, "TPA", buf, sizeof(buf), &trimmed));
      cout << "\nPosition: " << trimmed << '\n';
      x_e(GCmd(g, "SP 2000000")); //set up speed
      x_e(GCmd(g, "AC 1280000")); //acceleration
      x_e(GCmd(g, "DC 1280000")); //deceleration
      x_e(GCmd(g, "PR 200")); //Postion Relative.
      x_e(GCmd(g, "SH A")); //Servo Here
      cout << "Beginning independent motion... ";
      x_e(GCmd(g, "BG A")); //Begin motion
      x_e(GMotionComplete(g, "A")); //Block until motion is complete on axis A
```

```
        cout << "Motion Complete on A\n";
        x_e(GCmdT(g, "RP", buf, sizeof(buf), &trimmed));
        cout << "Position: " << trimmed << '\n';;

        turns=turns+1;
        count=1;
                        }
         break;


                }
                        count=1;
                        turns=1;
                        key = cv::waitKey(1);



    }

    error = camera.StopCapture();
    if ( error != PGRERROR_OK )
    {
        // This may fail when the camera was removed, so don't show
        // an error message
    }

    camera.Disconnect();

    return 0;
}
```

APPENDIX B

DIGITAL HOLOGRAPHY SIMULATION

Digital Holography Simulation

```
% Digital Holography
%%
clear all
close all
%%
% Sampling requirments

D1= 10e-2; % Area of target
D2= 20e-2; % Area of observation plane
Dz=14;    % distance propagated in m
wvl=793e-9; % wavelength in m
d1=1e-4; %in m
d2=1e-4; %in m

%%
k=2*pi/wvl; %
N=floor(wvl*Dz/(d1*d2)); %
%%
% Setting up the point source
arg=D2/(wvl*Dz);
x = (-N/2:(N/2-1))*d1;
y = (-N/2:(N/2-1))*d1;

[X,Y] = meshgrid(x,y);
[theta r1]=cart2pol(X,Y);
data1 = exp(-1i*k/(2*Dz)*r1.^2)*arg^2.*sinc(arg*X).*sinc(arg*Y);
figure(3)
imagesc(abs(data1))
colormap(flipud(gray))
axis off
%%
% Setup the target
ktarget=rand(N); % Generates A Grid of random values
target_phase=exp(-1i*2*pi*ktarget);
I=imread('test2.png');
I=imresize(I,[N,N]);
I=rgb2gray(I);
target=(double(I)/256).*target_phase;

%%
% Convolve the pointsource with the target
bt = ifftshift(ifft2(fft2(target).*fft2(data1)));
figure(7)
```

```
imagesc(abs(bt))
colormap(flipud(gray))
axis off
%%
% Propagate to the observation plane
[x2 y2 beam1]  = ang_spec_prop(bt, wvl, d1, d2, Dz);
beam1 = beam1*2e-7;

%%
%the hologram and retriving the
figure(9)
imagesc(real(beam1))
colormap(flipud(gray))
axis off
fx=(-N/2:N/2-1)/(N*d2);
fmax=max(max(fx));
% The local oscilator is setup
L=10*exp(-1i*2*pi*(fmax/2)*(x2+y2));
% To take into account the radius of curvature of the wavefront
Q=exp(-1i*k/2*((1-(d2/d1))/Dz+(1/Dz))*sqrt(x2.^2+y2.^2).^2);
mrt=L+beam1.*Q;

figure(46)
imagesc(abs(mrt))
colormap(flipud(gray))
axis off
%Intensisty Image of the hologram
in=abs(mrt).^2;
figure(12)
imagesc(in)
colorbar
title('intensity')
% Retriving the image
h=fftshift((fft2(fftshift(in))));
h(end/2+1,end/2+1) = 0;

figure(13)
imagesc(abs(h))
colormap(flipud(gray))
axis off
%%
% turbulence
r0 = .01; % coherence diameter [m]
L0 = 100; % outer scale [m]
```

```
l0 = 0.001;% inner scale [m]
delta=D2/N;
[phz_lo phz_hi] ...
    = ft_sh_phase_screen(r0, N, delta, L0, l0);
phz = phz_lo + phz_hi;

figure(14)
imagesc(phz)
colormap(flipud(gray))
axis off
%%
beamt=beam1.*exp(1i.*phz);
figure
imagesc(abs(beamt))
colormap(flipud(gray))
axis off
fx=(-N/2:N/2-1)/(N*d2);
fmax=max(max(fx));
% The local oscilator is setup
L=10*exp(-1i*2*pi*(fmax/2)*(x2+y2));
% To take into account the radius of curvature of the wavefront
Q=exp(-1i*k/2*((1-(d2/d1))/Dz+(1/Dz))*sqrt(x2.^2+y2.^2).^2);
mrt2=L+beamt.*Q;
figure(47)
imagesc(abs(mrt2))
colorbar
title('combined Lo')
%Intensisty Image of the hologram
in2=abs(mrt2).^2;
figure(48)
imagesc(in2)
colorbar
title('intensity')
% Retriving the image
h2=fftshift((fft2(fftshift(in2))));
h2(end/2+1,end/2+1) = 0;

figure(49)
imagesc(abs(h2))
colormap(flipud(gray))
axis off
```

APPENDIX C

ANGLE SPECTRUM PROPAGATION

Angle Spectrum Propagation- Jason Schmidt, Numerical Simulation of Optical Wave Propagation,2010, pp. 102

```
function [x2 y2 Uout] ...
   = ang_spec_prop(Uin, wvl, d1, d2, Dz)
% function [x2 y2 Uout] ...
%     = ang_spec_prop(Uin, wvl, d1, d2, Dz)

   N = size(Uin,1);   % assume square grid
   k = 2*pi/wvl;    % optical wavevector
   % source-plane coordinates
   [x1 y1] = meshgrid((-N/2 : 1 : N/2 - 1) * d1);
   r1sq = x1.^2 + y1.^2;
   % spatial frequencies (of source plane)
   df1 = 1 / (N*d1);
   [fX fY] = meshgrid((-N/2 : 1 : N/2 - 1) * df1);
   fsq = fX.^2 + fY.^2;
   % scaling parameter
   m = d2/d1;
   % observation-plane coordinates
   [x2 y2] = meshgrid((-N/2 : 1 : N/2 - 1) * d2);
   r2sq = x2.^2 + y2.^2;
   % quadratic phase factors
   Q1 = exp(i*k/2*(1-m)/Dz*r1sq);
   Q2 = exp(-i*pi^2*2*Dz/m/k*fsq);
   Q3 = exp(i*k/2*(m-1)/(m*Dz)*r2sq);
   % compute the propagated field
   Uout = Q3.* ift2(Q2 .* ft2(Q1 .* Uin / m, d1), df1);
```

APPENDIX D

FT PHASE SCREEN

```
ft_phase_screen- Jason Schmidt, Numerical Simulation of Optical Wave
Propagation,2010, pp. 171
function phz = ft_phase_screen(r0, N, delta, L0, l0)
% function phz ...
%     = ft_phase_screen(r0, N, delta, L0, l0)

   % setup the PSD
   del_f = 1/(N*delta);   % frequency grid spacing [1/m]
   fx = (-N/2 : N/2-1) * del_f;
   % frequency grid [1/m]
   [fx fy] = meshgrid(fx);
   [th f] = cart2pol(fx, fy);  % polar grid
   fm = 5.92/l0/(2*pi); % inner scale frequency [1/m]
   f0 = 1/L0;           % outer scale frequency [1/m]
   % modified von Karman atmospheric phase PSD
   PSD_phi = 0.023*r0^(-5/3) * exp(-(f/fm).^2) ...
      ./ (f.^2 + f0^2).^(11/6);
   PSD_phi(N/2+1,N/2+1) = 0;
   % random draws of Fourier coefficients
   cn = (randn(N) + i*randn(N)) .* sqrt(PSD_phi)*del_f;
   % synthesize the phase screen
   phz = real(ift2(cn, 1));
```

```
ft_sh_phase_screen - ft_phase_screen- Jason Schmidt, Numerical Simulation of Optical
Wave Propagation,2010, pp. 170
function [phz_lo phz_hi] ...
   = ft_sh_phase_screen(r0, N, delta, L0, l0)
% function [phz_lo phz_hi] ...
%     = ft_sh_phase_screen(r0, N, delta, L0, l0)

   D = N*delta;
   % high-frequency screen from FFT method
   phz_hi = ft_phase_screen(r0, N, delta, L0, l0);
   % spatial grid [m]
   [x y] = meshgrid((-N/2 : N/2-1) * delta);
   % initialize low-freq screen
   phz_lo = zeros(size(phz_hi));
   % loop over frequency grids with spacing 1/(3^p*L)
   for p = 1:3
      % setup the PSD
      del_f = 1 / (3^p*D); % frequency grid spacing [1/m]
      fx = (-1 : 1) * del_f;
      % frequency grid [1/m]
      [fx fy] = meshgrid(fx);
      [th f] = cart2pol(fx, fy);  % polar grid
      fm = 5.92/l0/(2*pi); % inner scale frequency [1/m]
      f0 = 1/L0;           % outer scale frequency [1/m]
      % modified von Karman atmospheric phase PSD
      PSD_phi = 0.023*r0^(-5/3) * exp(-(f/fm).^2) ...
         ./ (f.^2 + f0^2).^(11/6);
      PSD_phi(2,2) = 0;
      % random draws of Fourier coefficients
      cn = (randn(3) + i*randn(3)) ...
         .* sqrt(PSD_phi)*del_f;
      SH = zeros(N);
      % loop over frequencies on this grid
      for ii = 1:9
         SH = SH + cn(ii) ...
            * exp(i*2*pi*(fx(ii)*x+fy(ii)*y));
      end
      phz_lo = phz_lo + SH;   % accumulate subharmonics
   end
   phz_lo = real(phz_lo) - mean(real(phz_lo(:)));
```

APPENDIX E

INTGRAD2

intgrad2 - John D'Errico , Mathworks, 1/27/06

```
function fhat = intgrad2(fx,fy,dx,dy,f11)
% intgrad: generates a 2-d surface, integrating gradient information.
% usage: fhat = intgrad2(fx,fy)
% usage: fhat = intgrad2(fx,fy,dx,dy)
% usage: fhat = intgrad2(fx,fy,dx,dy,f11)
%
% arguments: (input)
%  fx,fy - (ny by nx) arrays, as gradient would have produced. fx and
%        fy must both be the same size. Note that x is assumed to
%        be the column dimension of f, in the meshgrid convention.
%
%        nx and ny must both be at least 2.
%
%        fx and fy will be assumed to contain consistent gradient
%        information. If they are inconsistent, then the generated
%        gradient will be solved for in a least squares sense.
%
%        Central differences will be used where possible.
%
%    dx - (OPTIONAL) scalar or vector - denotes the spacing in x
%        if dx is a scalar, then spacing in x (the column index
%        of fx and fy) will be assumed to be constant = dx.
%        if dx is a vector, it denotes the actual coordinates
%        of the points in x (i.e., the column dimension of fx
%        and fy.) length(dx) == nx
%
%        DEFAULT: dx = 1
%
%    dy - (OPTIONAL) scalar or vector - denotes the spacing in y
%        if dy is a scalar, then the spacing in x (the row index
%        of fx and fy) will be assumed to be constant = dy.
%        if dy is a vector, it denotes the actual coordinates
%        of the points in y (i.e., the row dimension of fx
%        and fy.) length(dy) == ny
%
%        DEFAULT: dy = 1
%
%   f11 - (OPTIONAL) scalar - defines the (1,1) eleemnt of fhat
%        after integration. This is just the constant of integration.
%
%        DEFAULT: f11 = 0
%
```

```
% arguments: (output)
%   fhat - (nx by ny) array containing the integrated gradient
%
% Example usage 1: (Note x is uniform in spacing, y is not.)
%  xp = 0:.1:1;
%  yp = [0 .1 .2 .4 .8 1];
%  [x,y]=meshgrid(xp,yp);
%  f = exp(x+y) + sin((x-2*y)*3);
%  [fx,fy]=gradient(f,.1,yp);
%  tic,fhat = intgrad2(fx,fy,.1,yp,1);toc
%
%  Time required was 0.06 seconds
%
% Example usage 2: Large grid, 101x101
%  xp = 0:.01:1;
%  yp = 0:.01:1;
%  [x,y]=meshgrid(xp,yp);
%  f = exp(x+y) + sin((x-2*y)*3);
%  [fx,fy]=gradient(f,.01);
%  tic,fhat = intgrad2(fx,fy,.01,.01,1);toc
%
%  Time required was 4 seconds

% Author; John D'Errico
% Current release: 2
% Date of release: 1/27/06

% size
if (length(size(fx))>2) || (length(size(fy))>2)
  error 'fx and fy must be 2d arrays'
end
[ny,nx] = size(fx);
if (nx~=size(fy,2)) || (ny~=size(fy,1))
  error 'fx and fy must be the same sizes.'
end
if (nx<2) || (ny<2)
  error 'fx and fy must be at least 2x2 arrays'
end

% supply defaults if needed
if (nargin<3) || isempty(dx)
  % default x spacing is 1
  dx = 1;
end
```

```
if (nargin<4) || isempty(dy)
  % default y spacing is 1
  dy = 1;
end
if (nargin<5) || isempty(f11)
  % default integration constant is 0
  f11 = 0;
end

% if scalar spacings, expand them to be vectors
dx=dx(:);
if length(dx) == 1
  dx = repmat(dx,nx-1,1);
elseif length(dx)==nx
  % dx was a vector, use diff to get the spacing
  dx = diff(dx);
else
  error 'dx is not a scalar or of length == nx'
end
dy=dy(:);
if length(dy) == 1
  dy = repmat(dy,ny-1,1);
elseif length(dy)==ny
  % dy was a vector, use diff to get the spacing
  dy = diff(dy);
else
  error 'dy is not a scalar or of length == ny'
end

if (length(f11) > 1) || ~isnumeric(f11) || isnan(f11) || ~isfinite(f11)
  error 'f11 must be a finite scalar numeric variable.'
end

% build gradient design matrix, sparsely. Use a central difference
% in the body of the array, and forward/backward differences along
% the edges.

% A will be the final design matrix. it will be sparse.
% The unrolling of F will be with row index running most rapidly.
rhs = zeros(2*nx*ny,1);
% but build the array elements in Af
Af = zeros(2*nx*ny,6);
L = 0;
```

```
% do the leading edge in x, forward difference
indx = 1;
indy = (1:ny)';
ind = indy + (indx-1)*ny;
rind = repmat(L+(1:ny)',1,2);
cind = [ind,ind+ny];
dfdx = repmat([-1 1]./dx(1),ny,1);
Af(L+(1:ny),:) = [rind,cind,dfdx];
rhs(L+(1:ny)) = fx(:,1);
L = L+ny;

% interior partials in x, central difference
if nx>2
  [indx,indy] = meshgrid(2:(nx-1),1:ny);
  indx = indx(:);
  indy = indy(:);
  ind = indy + (indx-1)*ny;
  m = ny*(nx-2);

  rind = repmat(L+(1:m)',1,2);
  cind = [ind-ny,ind+ny];

  dfdx = 1./(dx(indx-1)+dx(indx));
  dfdx = dfdx*[-1 1];

  Af(L+(1:m),:) = [rind,cind,dfdx];

  rhs(L+(1:m)) = fx(ind);

  L = L+m;
end

% do the trailing edge in x, backward difference
indx = nx;
indy = (1:ny)';
ind = indy + (indx-1)*ny;
rind = repmat(L+(1:ny)',1,2);
cind = [ind-ny,ind];
dfdx = repmat([-1 1]./dx(end),ny,1);
Af(L+(1:ny),:) = [rind,cind,dfdx];
rhs(L+(1:ny)) = fx(:,end);
L = L+ny;

% do the leading edge in y, forward difference
```

```
indx = (1:nx)';
indy = 1;
ind = indy + (indx-1)*ny;
rind = repmat(L+(1:nx)',1,2);
cind = [ind,ind+1];
dfdy = repmat([-1 1]./dy(1),nx,1);
Af(L+(1:nx),:) = [rind,cind,dfdy];
rhs(L+(1:nx)) = fy(1,:).';
L = L+nx;

% interior partials in y, use a central difference
if ny>2
  [indx,indy] = meshgrid(1:nx,2:(ny-1));
  indx = indx(:);
  indy = indy(:);
  ind = indy + (indx-1)*ny;
  m = nx*(ny-2);

  rind = repmat(L+(1:m)',1,2);
  cind = [ind-1,ind+1];

  dfdy = 1./(dy(indy-1)+dy(indy));
  dfdy = dfdy*[-1 1];

  Af(L+(1:m),:) = [rind,cind,dfdy];

  rhs(L+(1:m)) = fy(ind);

  L = L+m;
end

% do the trailing edge in y, backward diffeence
indx = (1:nx)';
indy = ny;
ind = indy + (indx-1)*ny;
rind = repmat(L+(1:nx)',1,2);
cind = [ind-1,ind];
dfdy = repmat([-1 1]./dy(end),nx,1);
Af(L+(1:nx),:) = [rind,cind,dfdy];
rhs(L+(1:nx)) = fy(end,:).';

% finally, we can build the rest of A itself, in its sparse form.
A = sparse(Af(:,1:2),Af(:,3:4),Af(:,5:6),2*nx*ny,nx*ny);
```

```
% Finish up with f11, the constant of integration.
% eliminate the first unknown, as f11 is given.
rhs = rhs - A(:,1)*f11;

% Solve the final system of equations. They will be of
% full rank, due to the explicit integration constant.
% Just use sparse \
fhat = A(:,2:end)\rhs;
fhat = reshape([f11;fhat],ny,nx);
```

APPENDIX F

IMAGERECH

```
ImageRecH I. S. Sevcenco, P. Hampton, P. Agathoklis, "A wavelet based method for
image reconstruction from gradient data with applications",  Multidimensional Systems
and Signal Processing, November 2013
function  R = ImageRecH(gxH,gyH,meanval,PoissonOn,no_iter)
% Program ImageRecH.m
% ------------------------------------------------------------------
% This function reconstructs an image from its Hudgin gradient
% (please see Sec.2.2 in Ref. 1 for Hudgin gradient discretization Equations).
% The program works for both multi-channel (e.g., color),
% as well as for one channel (e.g., grayscale) images.
%
% Inputs:
%      gxH and gyH = Hudgin gradients used to reconstruct the image;
%                  should be two matrices of the same size
%                  as the image to be reconstructed
%                  (last column in gxH should be 0; last row in gyH should be 0);
%      meanval = if the image is color, should be a 3 x 1 vector storing
%              the mean value of the image we want to reconstruct
%              (one scalar per channel, meanval(1) corresponding to R,
%              meanval(2) corresponding to G, meanval(3) corresponding to B
%      PoissonOn = binary flag for Poisson solver during reconstruction
%           can be: 0: Poisson solver is not used during reconstruction
%                   1: Poisson solver is used during reconstruction
%      no_iter = number of iterations used in the Poisson solver;
%              -- if no Poisson solver is used in the reconstruction, then
%              the value of no_iter is irrelevant, and set to 0 by
%              default;
%              -- if the Poisson solver is used in the reconstruction,
%              a suggested value for the number of iterations is three
%              (default)
%              -- the user can also specify the desired number of
%              iterations (acceptable values are positive integers, greater than 1)
% Output: R = reconstructed image
%
% For Examples of how to use the function please consult the
% documentation included in our submission (DocumentationToolbox.pdf)
%
% References:
%
% [1] I. S. Sevcenco, P. Hampton, P. Agathoklis, "A wavelet based method
% for image reconstruction from gradient data with applications",
% Multidimensional Systems and Signal Processing, November 2013
%
% Earlier references using Fried gradient discretization (see sec. 2.1. in [1]):
```

```
%
% [2] P. Hampton, P. Agathoklis, C. Bradley, "A New Wave-Front Reconstruction
% Method for Adaptive Optics Systems Using Wavelets", IEEE Journal of
% Selected Topics in Signal Processing, vol. 2, no. 5, October 2008
%
% [3] P. Hampton, P. Agathoklis, "Comparison of Haar Wavelet-based and Poisson-
based
% Numerical Integration Techniques", 2010

% Written by: Ioana Sevcenco, University of Victoria%
% Last updated: August 28th, 2014
if (PoissonOn==0) && (nargin < 5),
   no_iter = 0; % not used, but defined to avoid compiling errors
end
if (PoissonOn==1) && (nargin < 5),
   no_iter = 3; % suggested number of iterations
end;
if (size(gxH,3) == 3) % i.e., if image is three channel
      [dx_rec1,dx_rec2,dx_rec3] = splitRGB(gxH);
      [dy_rec1,dy_rec2,dy_rec3] = splitRGB(gyH);
      R1 = OneChannelRec(dx_rec1,dy_rec1,PoissonOn,meanval(1),no_iter);
      R2 = OneChannelRec(dx_rec2,dy_rec2,PoissonOn,meanval(2),no_iter);
      R3 = OneChannelRec(dx_rec3,dy_rec3,PoissonOn,meanval(3),no_iter);
      R = CombineRGB(R1,R2,R3);
   else if (size(gxH,3) == 1) % i.e., if image is grayscale
            R = OneChannelRec(gxH,gyH,PoissonOn,meanval,no_iter);
       end
end
```