THE PHASE RETRIEVAL PROBLEM AND ITS APPLICATIONS IN OPTICS

A Thesis

by

RUSSELL EDWARD TRAHAN III

Submitted to the Office of Graduate Studies of
Texas A&M University
In partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Approved by:

Chair of Committee, David Hyland
Committee Members, Suman Chakravorty
Tom Pollock
Alexey Belyanin

Head of Department, Rodney Bowersox

December 2014

Major Subject: Aerospace Engineering

ABSTRACT

In this dissertation the various forms and applications of the phase retrieval problem in imaging are discussed. The phase retrieval problem in general refers to the estimation of the phase of a complex-valued function based on knowledge of its magnitude. Here the phase retrieval problem is applied to the estimation of the phase of an electromagnetic wave field based on knowledge of its magnitude. The magnitude (not the phase) can be measured using several devices as discussed.

There are many applications of phase retrieval which have been explored where the mapping between the detected wave field magnitude and the light source is the Fourier transform. Within these applications phase retrieval solutions are used to estimate the phase of the Fourier transform so as to obtain the image or the shape of the light emitter. These solutions necessitate a model of the propagation of the wave field, a method of detecting the field's magnitude, and a method of estimating the phase of the observed field. The first two considerations are discussed here historically and reference many significant scientific discoveries, namely the Huygens-Fresnel principle, the Van Cittert-Zernike theorem, and the Michelson interferometer.

Within the field of interferometry where the Fourier transform is the mapping between the light source and observed wave field, most solutions utilize a discrete Fourier transform. The estimate of the light source takes the form of a two-dimensional pixelated image. These solutions have been explored for many years and have many variations for particular applications. Not many of these solution methods, however, have confronted the

problem of measurement noise. Measurement noise here refers to noise in the quantification of the magnitude of the wave field at the observed locations. Within this dissertation the negative effects of noise are analyzed and a method of filtering the noise from the data is derived, tested, and shown to be effective. In a separate analysis the use of the discrete Fourier transform as opposed to the continuous Fourier transform is questioned. A phase solution is proposed which is capable of estimating the source of the observed wave field and takes discrete magnitude data and outputs a continuous image function formed from Gaussian bases. This method is beneficial from an analytical point-of-view since it is not an iterative solution. It also has an error metric which definitively determines whether the true solution has been found—unlike the traditional solution methods.

The phase retrieval problem is also explored in the case where the Fourier transform is not the mapping between the image and the observed wave field. Particularly, the case of a small asteroid occulting a star is analyzed with the goal of characterizing the shape of the asteroid's silhouette. A solution is formulated capable of resolving the asteroid silhouette based on time histories of the intensity of the wave field measured at multiple spatial locations. The solution is based on an analysis of the shadow that the occulter casts.

The phase retrieval problem is present in many current fields of imaging and remains a prominent source of inquiry. Although many solution methods exist, there are still many improvements that can be made. This dissertation addresses some potential improvements to existing solutions and proposes new applications and formulations of the phase retrieval problem.

ACKNOWLEDGEMENTS

I sincerely thank Dr. David Hyland for his guidance, encouragement, and support while working on my master's and doctoral degrees. It has been an honor and a pleasure to work with him. I would also like to thank my committee, Dr. Belyanin, Dr. Chakravorty, and Dr. Pollock, for their time and advice over the last several years. I'd also like to acknowledge the help of Dr. Hyland's graduate students, namely Greg Kelderman, Daniel Fitch, and Micaela Landivar, who all have aided me greatly in my time in graduate school.

I would also like to thank my parents for their encouragement to pursue my education in these many years of college. Their support and example have been priceless in the completion of my education.

Finally, I thank my fiancé Brookelynn Russey who has stood with me and kept me going despite my late nights and weekends working. Words cannot express how grateful I am to have had her with me through this time.

NOMENCLATURE

ω	Angular frequency of light
ρ	Data coverage ratio
R	Distance from source to observer
z	Distance from object to observer
$\langle \ldots \rangle$	Ensemble average operator
$U(t)$	Electromagnetic field
$E(t)$	Electric field
$I(t)$	Electric field intensity
S	Far field source object's plane
$\underline{\theta} = (\theta_x, \theta_y)$	Far field source object/image angular view coordinate
$\underline{x} = (x, y)$	Far field source object/image spatial coordinate
$\Delta I(t)$	Fluctuation in electric field intensity
$\underline{u} = (u, v)$	Fourier domain coordinates
$F[x]$	Fresnel integral
F	Fresnel number
$I(\underline{x})$	Image pixel value
ICI	Intensity correlation interferometry

λ	Light wavelength
γ	Normalized mutual coherence
$J(\underline{u})$	Mutual coherence
MTF	Modulation transfer function
O	Observation plane
$\underline{\xi} = (\xi, \psi)$	Observation plane coordinates
OTF, $\hat{O}(...)$	Optical transfer function
φ	Phase of a complex value
RMS	Root Mean Squared
$\Gamma(\underline{\theta})$	Silhouette Function
c	Speed of light
SNR	Signal-to-noise ratio

TABLE OF CONTENTS

Page

LIST OF FIGURES

FIGURE Page

LIST OF TABLES

TABLE Page

1. INTRODUCTION

The phase retrieval problem in general is the estimation of the phase of a complex-valued function based primarily on its known magnitude. This problem can be found in many areas of physics, but most attention has been placed on its various forms within the recovery of a wave field's properties using measured diffraction information. This field of study is commonly referred to as interferometry. Within most of these applications, an electromagnetic wave field is emitted by some objective, and this field's amplitude is measured some distance away. In order for the observer to determine the field at the objective, the measured amplitude is used to devise an estimate of the field's phase at the observer's location. Based on the knowledge of the field's propagation, observed amplitude, and estimated phase, the field's origin at the objective can be determined. There are thus three areas to explore within the phase retrieval problem: a wave field's propagation based on a physical model, an experimental means of observing a wave field's amplitude, and the estimation of the phase of the observed field. In this work the first two are discussed in a historical context to introduce the latter. Many approaches to the estimation of the phase have been devised for various applications. The methods that are associated with the applications discussed here are explained and compared.

The underlying theme of this work is the various formulations of the phase retrieval problem applied to imaging. Almost all research in phase retrieval is based on the early work of Gerchberg and Saxton—explained in detail later. The premise of their methods is that an illuminated object emits light (electromagnetic radiation) towards an

observer. The vector components of the electromagnetic field are described by complex-valued functions, and measurement devices are typically only sensitive to the electric field. Since all components of the electric field satisfy the wave equation, scalar diffraction theory is used to describe the light's wave field. Scalar diffraction theory represents the wave field as a complex-valued function in spatial position and time. In this work, the absolute magnitude of the complex wave field is referred to as the "amplitude." The amplitude or amplitude-squared of the light's wave field is detected by some means at some known positions relative to the objective. Knowledge of the wave field's amplitude is used in conjunction with the knowledge of the propagation of light to reconstruct an image of the object. The three components of the phase retrieval process are evident. The detection of the wave field amplitude is performed by some apparatus—an intensity detector. The phase of the observed field at the detector must be estimated by some means. Lastly, the wave field amplitude and phase at the observer, combined with knowledge of the dynamics of the wave field propagation, are used to determine a fine-resolution image of the objective. Here, these three components are described and used to formulate and solve three distinct problems in phase retrieval. These three problems are namely recovery of a pixelated image of an object, recovery of a continuous image of an object, and recovery of the silhouette of an object which is occulting a light source. Each of these methods is suited for particular applications. In particular, the first two topics are primarily associated with interferometry while the last is unique.

Within this dissertation, section 1 contains discussions on the wave nature of light. This discussion contains both historical and mathematical models for light propagation

which has led to the current understanding of light. Next, Section 1 discusses methods of detecting a light field and gives the mathematical rationale for an interferometer. The Michelson interferometer and intensity correlation interferometer designs are discussed in detail. Several practical applications are also discussed with the most attention placed on the intensity correlation interferometer. Section 2 presents some of the noteworthy methods of solving the phase estimation problem. The techniques used here to compare phase retrieval algorithms are also presented. Section 3 gives an original method of solving the phase problem that builds on the current state-of-the-field methods for handling noise in the measurement of the light field's amplitude. Its derivation is discussed in detail, and its performance is explored in various scenarios. Section 4 presents a new, unique formulation of the phase retrieval problem which most accurately captures the physics of using telescopes to detect the wave field emitted from distant objects. Section 5 presents progress on characterizing the silhouette of asteroids by observing their shadow when occulting a distant star. Finally, the Conclusion highlights the contributions of this research to the physics and engineering communities. The work presented in Section 3 has been previously published in [1, 2], Section 4 has been published in [3], and Section 5 has been published in [4]. The material discussed here expands on the discussions in the previous publications.

1.1. Light as a Wave Field

The description of light has been the object of inquiry for millennia across many cultures. In classical Greece Empedocles in the fifth century BC held that light was emitted by the human eye which contained the four elements: fire, air, earth, and water. The first

mathematical approaches were made by Euclid around 300 BC who described light as traveling in straight lines and described reflection. Ptolemy described refraction in the second century in his book *Optics*. In ancient Indian Hindu philosophy schools taught that light was one of the five fundamental elements. The Vaisheshika School posed light as fire atoms. Most ancient descriptions of light contributed to setting the precedent of describing light as a particle [5].

In more modern times, the particle theory of light was further advanced by René Descartes. In his 1637 publication, he described refraction as being caused by changes in the speed of light. He erroneously concluded that light propagation is analogous to sound propagation which increases in speed in thicker media. In fact, the opposite is true for light. Pierre Gassendi also published a particle theory of light which was embraced by Isaac Newton. Newton published his *Hypothesis of Light* in 1675 in which he speculated that light is composed of corpuscles [6]. In his 1704 publication *Opticks*, he was able to describe refraction but did so incorrectly by attributing the acceleration of corpuscles when changing media to differences in gravitational pull in materials of differing density. Concurrently, Robert Hooke was working on his 1665 publication *Micrographia* in which he suggested the first wave theory of light by concluding that light vibrates perpendicular to the direction of propagation [7, 8]. He observed interference phenomena in soap bubbles and oil on water. He was not able to fully explain these phenomena, but he attributed them to light being a wave. Christiaan Huygens published a mathematical wave theory for light in 1690 called the *Treatise on Light*. He held that light waves are independent of gravity and slow when entering dense mediums. Huygens correctly postulated that every point

illuminated by light can be thought of as a point source of light. Leonhard Euler published his support for the wave theory of light in 1746 citing diffraction as evidence of light being a wave field. The wave theory was later validated by Thomas Young around 1800 using his double-slit diffraction experiment [9]. Augustin-Jean Fresnel developed a mathematical wave theory of light which built upon Huygen's work and employed a pure transverse wave [8]. He was able to explain reflection, refraction, and double refraction. Huygens and Fresnel's works are commonly combined and referred to as the Huygens-Fresnel principle. There were many contributors to the description of light, but the aforementioned are the key contributors to the models needed here [10].

The culmination of the various theories of light give the modern Huygens-Fresnel model which is used in developing the phase retrieval problem. In the following work, the time dependence of the wave field is assumed to be nearly periodic at frequency ω. This is the well-known quasi-monochromatic assumption. Further, we adopt the common practice of embedding the wave propagation formulation in the complex domain. In this model, the most basic result is that a point source of light creates a spherically-spreading wave field. Assuming the wave field is created with strength U_0, wavelength λ, and speed c, the complex value of the wave field at a point $\mathbf{P}$ due to the source at point $\mathbf{Q}$ is [11]

$$U(\mathbf{P}) = \frac{U_0}{\|\mathbf{P}-\mathbf{Q}\|}\exp\left(i\omega t - i\frac{2\pi c}{\lambda}\|\mathbf{P}-\mathbf{Q}\|\right). \tag{1.1}$$

where $\mathbf{P}$ and $\mathbf{Q}$ are the position vectors of the observation and source points, respectively.

Since a true point source of light cannot be realized, this idea needs to be generalized to describe a region emitting light. Using the principle of superposition, the field due to an infinitesimal area can be summed over a spatial region to describe a finite source of light in the form

$$U(\mathbf{P}) = \iint \frac{U_0}{\|\mathbf{P}-\mathbf{Q}\|} \exp\left(i\omega t - i\frac{2\pi c}{\lambda}\|\mathbf{P}-\mathbf{Q}\| \right) d\mathbf{Q} \tag{1.2}$$

A simple example for utilizing the wave field idea is the Fraunhofer Single Slit experiment. In this experiment monochromatic planar waves arrive at a wall, the image plane, which has a small opening. The light passes through the small slit and propagates until it reaches a second solid wall, the observation plane. The slit has width w and the two walls are separated by the distance D. The goal of the example is to compute the intensity of the wave field across the observation plane. The diagram is shown in Fig. 1.

A point $\mathbf{P}$ on the observation plane receives light from every point within the slit, but the phase is shifted by α due to the different distances across the slit to $\mathbf{P}$. The light taking the shortest path from the slit to $\mathbf{P}$ is considered a reference and has $\alpha = 0$. The maximum phase difference which is associated with the longest path from the slit to $\mathbf{P}$ is

$$\alpha_{\max} = \frac{\pi w y}{\lambda D}. \tag{1.3}$$

According to the Huygen-Fresnel principle, every illuminated point within the slit acts as a point source. By adding the wave field emitted by these point sources using equation (1.2), the spatially dependent part of the wave field at point $\mathbf{P}$ is expressed as

$$E(y) = \frac{U_0}{D}\mathrm{Re}\left\{\int_0^{\alpha_{\max}} \exp(i\alpha)\, d\alpha\right\} = \frac{U_0}{D}\mathrm{sinc}\,\alpha_{\max} \tag{1.4}$$

Hereon, the sinc function is defined as $\mathrm{sinc}(\alpha) = \sin(\alpha)/\alpha$. Intensity is defined as the square of the amplitude of the field. The total intensity at a point $\mathbf{P}(y)$ on the observation plane as shown in Fig. 1 is thus

$$I(y) = \left(\frac{U_0}{D}\mathrm{sinc}\,\frac{\pi w y}{\lambda D}\right)^2. \tag{1.5}$$

The intensity pattern shown in Fig. 1 at the observation plane corresponds to equation (1.5). Fig. 2 is a photograph from an experiment using a laser and a single slit to visually confirm this derivation. This example shows the correlation between the Huygen-Fresnel principle's suggested behavior of light going through a single slit and the reality observed in an experiment.

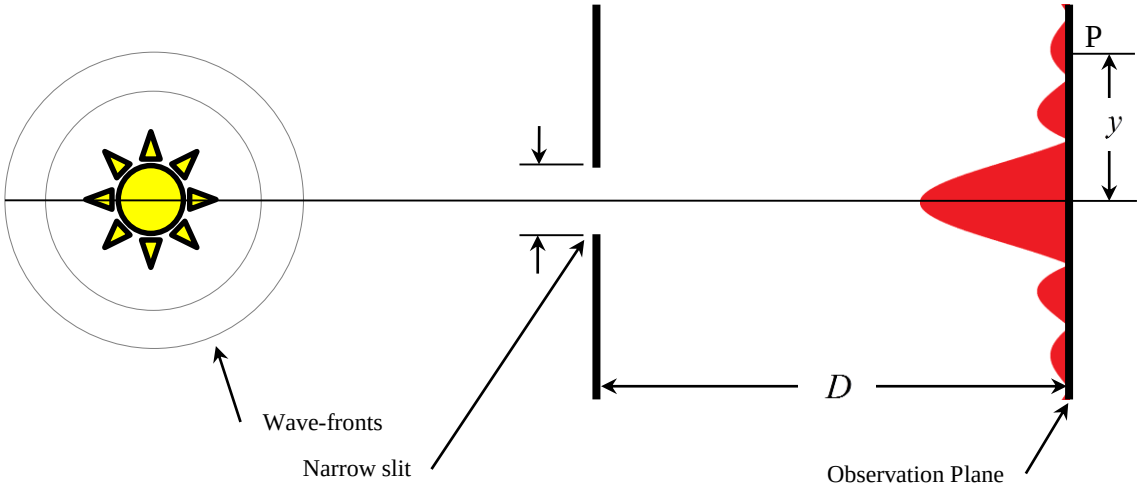


Fig. 1. Schematic of the Fraunhofer single slit experiment.

Fig. 2. Photograph of the interference pattern resulting from a laser beam going through a single slit [12].

The double-slit experiment is a continuation of the Fraunhofer single-slit experiment as shown in Fig. 3. After passing through the single-slit, the field passes through the second wall which has two equally sized slits. The field emanating from these two slits is them made to interfere and project an interference pattern on the observation plane. The propagation from the wall with the double-slits to the observation plane is the region of interest. The double-slit experiment was first performed by Thomas Young and is one of the first forms of conclusive evidence that light indeed is a wave [9].

The double-slit experiment can be analyzed in the same manner as the double slit experiment. The path length from the top slit to a point $\mathbf{P}$ is

$$r_1^2 = r^2 + \left(\frac{L}{2}\right)^2 - Lr\sin\theta \tag{1.6}$$

and the path length for the bottom slit is

$$r_2^2 = r^2 + \left(\frac{L}{2}\right)^2 + Lr\sin\theta \tag{1.7}$$

where D is the distance between the walls, L is the distance between the slit centers, and $\theta = \sin^{-1} y/D$. When $D \gg L$, the difference between the two path lengths is approximated as

$$\delta = r_2 - r_1 \approx L\sin\theta \tag{1.8}$$

The wave field is thus

$$
\begin{aligned}
E(y) &= \frac{U_0}{D}\mathrm{Re}\left\{\int_0^{\alpha_{\max}} \exp\left(i\alpha - \frac{2\pi\delta}{2\lambda}\right) + \exp\left(i\alpha + \frac{2\pi\delta}{2\lambda}\right) d\alpha\right\} \\
&= \frac{U_0}{D}\mathrm{Re}\left\{\frac{\exp(i\alpha_{\max})\exp\left(-\frac{\pi\delta}{\lambda}\right)}{i\alpha_{\max}} + \frac{\exp(i\alpha_{\max})\exp\left(\frac{\pi\delta}{\lambda}\right)}{i\alpha_{\max}}\right\} \\
&= \frac{2U_0}{D}\cos\left(\frac{\pi Ly}{\lambda D}\right)\frac{\sin\left(\frac{\pi wy}{\lambda D}\right)}{\frac{\pi wy}{\lambda D}}
\end{aligned}
\tag{1.9}
$$

The resulting intensity is

$$
I(y) = \frac{4U_0^2}{D^2}\cos^2\left(\frac{\pi Ly}{\lambda D}\right)\left[\frac{\sin\left(\frac{\pi wy}{\lambda D}\right)}{\frac{\pi wy}{\lambda D}}\right]^2 \tag{1.10}
$$

This interference pattern is shown in the schematic in Fig. 3. Notice that the computed pattern matches the interference pattern viewed in an experiment as shown in Fig. 4. Most notable a high frequency oscillation is mixed with a low frequency sinc function.

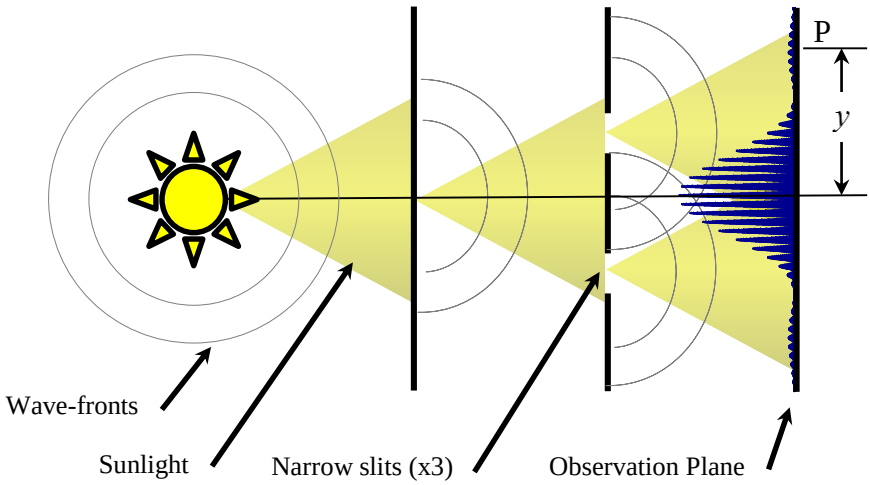


Fig. 3. Schematic of the double-slit experiment.

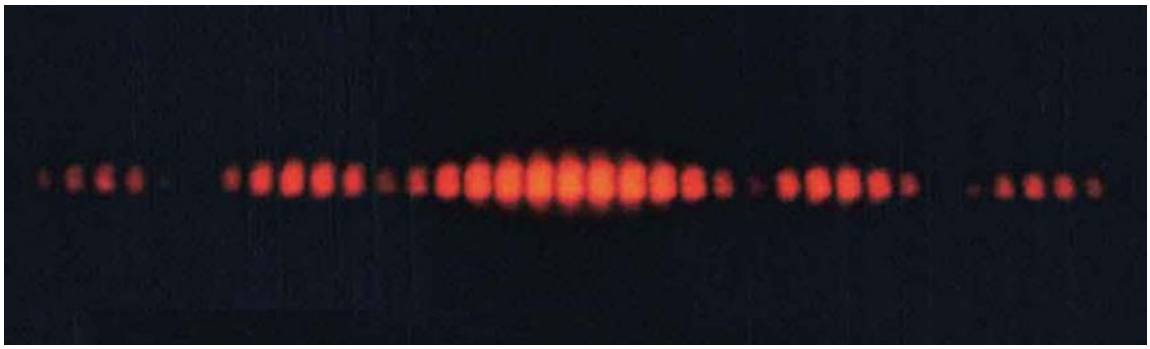


Fig. 4. Photograph of the diffraction pattern resulting from a laser going through a double slit [17].

1.2. The Van Cittert-Zernike Theorem

Section 1.1 gave the history and an example of the wave theory of light. In the example, the wave field was propagated from the source to the observation plane through a simple slit. The propagation process as previously derived can be quite cumbersome for an arbitrarily shaped source. The Van Cittert-Zernike theorem addresses this issue and creates a mapping between the source geometry and the intensity at the observation plane which turns out to be the Fourier transform. The traditional phase retrieval problem is based entirely on this mapping so its derivation will be shown [13, 14]. The theorem expresses the cross-correlation of the field measured at two points on the observation plane in terms of the intensity distribution of the radiant object to be imaged. It is assumed that the object is an incoherent source, i.e. any two source points radiate with phases that are random and uncorrelated. The exact form of the theorem is complicated, but usual conditions yield some algebraic simplifications and Taylor series expansions which lead to the Fourier transform as the mapping between the source and observation. Due to the simplicity of its final form and especially the fact that it relates quantities that can be measured even in the high frequency optical regime, the Van Cittert-Zernike theorem is more commonly used to predict the propagation of light than the Huygen-Fresnel principle alone.

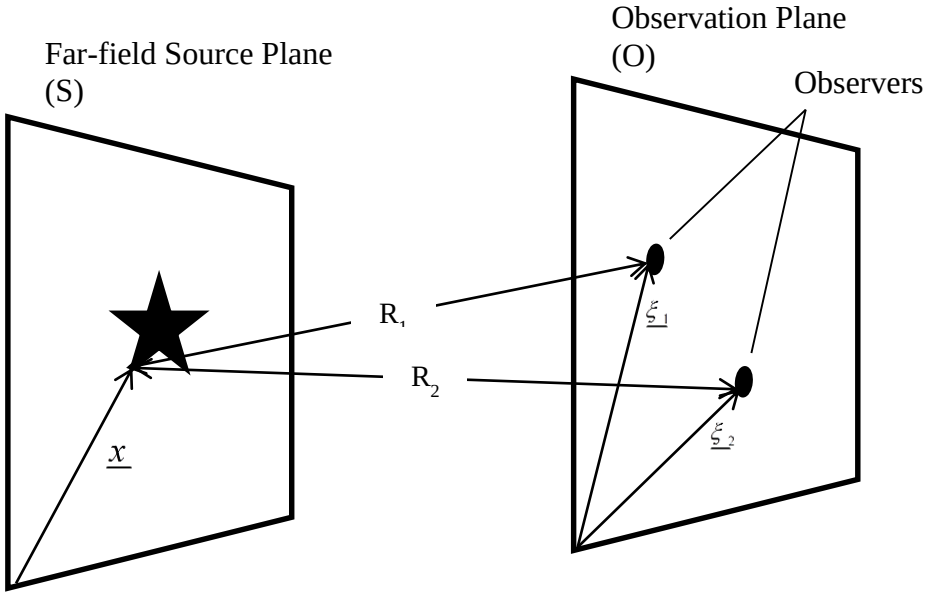


Fig. 5. Two observer in an observation plane and a light source in the far-field plane.

Consider a point light source on the far-field parameterized by position $\underline{x} = [x, y]$ and time t. The field at some point $\underline{\xi}_j = [\xi, \psi]$ on the observation plane is thus

$$E_j(\underline{x}, t) = A(\underline{x}, t)\frac{1}{R_j}\exp\left(-i2\pi\frac{ct - R_j}{\lambda}\right) \tag{1.11}$$

where A is the strength of the source field, R_j is the distance from the source position $\underline{x}$ to the observation position $\underline{\xi}_j$, λ is the mean wavelength, and c is the speed of light. The mean wavelength refers to the center of a narrow band of wavelengths. This is denoted the quasi-monochromatic assumption. Consider that the field is observed at two spatial positions $\underline{\xi}_1$ and $\underline{\xi}_2$. The time average of the product of the fields from the same point source is the first-order correlation [15],

$$\left\langle E_1\left(\underline{x},t\right)E_2^*\left(\underline{x},t\right)\right\rangle$$
$$=\left\langle A\left(\underline{x},t-\frac{R_1}{c}\right)A\left(\underline{x},t-\frac{R_2}{c}\right)\right\rangle\frac{\exp\left(-i2\pi\,\frac{ct-R_1}{\lambda}\right)}{R_1}\frac{\exp\left(i2\pi\,\frac{ct-R_2}{\lambda}\right)}{R_2}. \tag{1.12}$$

The time shifts represent the phase shift of the wave as it travels over some distance. Time can arbitrarily be shifted to simplify which gives

$$\left\langle E_1\left(\underline{x},t\right)E_2^*\left(\underline{x},t\right)\right\rangle=\left\langle A\left(\underline{x},t\right)A\left(\underline{x},t-\frac{R_2-R_1}{c}\right)\right\rangle\frac{\exp\left(i2\pi\,\frac{R_1-R_2}{\lambda}\right)}{R_1R_2}. \tag{1.13}$$

This time shift complicated one of the amplitude terms. Under the condition $c\Delta t \gg \left|R_1-R_2\right|$ where Δt is the measurement time interval, the expression further simplifies to

$$\left\langle E_1\left(\underline{x},t\right)E_2^*\left(\underline{x},t\right)\right\rangle=\left\langle A\left(\underline{x},t\right)A\left(\underline{x},t\right)\right\rangle\frac{\exp\left(i2\pi\,\frac{R_1-R_2}{\lambda}\right)}{R_1R_2}. \tag{1.14}$$

The intensity of the field is defined as the square of the amplitude so $\left\langle A\left(\underline{x},t\right)A\left(\underline{x},t\right)\right\rangle$ is the average intensity of the source at $\underline{x}$ which gives

$$\left\langle E_1\left(\underline{x},t\right)E_2^*\left(\underline{x},t\right)\right\rangle=I\left(\underline{x}\right)\frac{\exp\left(i2\pi\,\frac{R_1-R_2}{\lambda}\right)}{R_1R_2}. \tag{1.15}$$

Equation (1.15) is the mutual coherence of the field emitted by the source at $\underline{x}$ between the observations points $\underline{\xi}_1$ and $\underline{\xi}_2$. Integrating this equation over the entire intensity distribution of the source gives the coherence equation

$$J\left(\underline{\xi}_1,\underline{\xi}_2\right)=\iint I\left(\underline{x}\right)\frac{\exp\left(i2\pi\,\frac{R_1-R_2}{\lambda}\right)}{R_1R_2}d\underline{x}. \tag{1.16}$$

A position $\underline{x}$ in the source plane can be defined in terms of the direction cosines $\underline{\theta}=\left[\theta_x,\theta_y\right]=\left[\frac{x}{R},\frac{y}{R}\right]$ for $R\gg x$ and $R\gg y$. Applying this change of variables requires a change in the variable of integration which gives

$$\frac{d\underline{x}}{R_1R_2}=d\underline{\theta} \tag{1.17}$$

The integral can thus be rewritten as

$$J\left(\underline{\xi}_1,\underline{\xi}_2\right)=\iint I\left(\underline{\theta}\right)\exp\left(i2\pi\,\frac{R_1-R_2}{\lambda}\right)d\underline{\theta}. \tag{1.18}$$

The $\frac{R_1-R_2}{\lambda}$ term can be simplified with a few steps. Let R be the normal distance from the source plane to the observation plane. If $\underline{\chi}_i=\left[\xi_i,\psi_i\right]$ it follows that

$$\frac{R_1-R_2}{\lambda}=R\sqrt{1+\frac{\xi_1^2}{R^2}+\frac{\psi_1^2}{R^2}}-R\sqrt{1+\frac{\xi_2^2}{R^2}+\frac{\psi_2^2}{R^2}}. \tag{1.19}$$

Expanding the square root in a Taylor series gives

$$\begin{aligned}\frac{R_1-R_2}{\lambda}&\approx\frac{R}{\lambda}\left[1+\frac{1}{2}\left(1+\frac{\xi_1^2+\psi_1^2}{R^2}\right)\right]-R\left[1+\frac{1}{2}\left(1+\frac{\xi_2^2+\psi_2^2}{R^2}\right)\right]\\&\approx\frac{1}{2R\lambda}\left[\left(\xi_1-\xi_2\right)\left(\xi_1+\xi_2\right)+\left(\psi_1-\psi_2\right)\left(\psi_1+\psi_2\right)\right].\end{aligned} \tag{1.20}$$

With the definition of the wavenumber plane, also referred to as the "u-v plane,"

$$u\equiv\frac{\xi_2-\xi_1}{\lambda},\ v\equiv\frac{\psi_2-\psi_1}{\lambda} \tag{1.21}$$

and with the far-field plane expressed as

$$\theta_x = \frac{\xi_1 + \xi_2}{2R}, \ \theta_y = \frac{\psi_1 + \psi_2}{2R}, \tag{1.22}$$

it follows that

$$\frac{R_2 - R_1}{\lambda} = u\theta_x + v\theta_y \ . \tag{1.23}$$

The mutual coherence is thus

$$J(u,v) = \iint I(\theta_x, \theta_y) \exp\left(-i2\pi\left(u\theta_x + v\theta_y\right)\right) d\theta_x d\theta_y \tag{1.24}$$

which is the form of a standard two-dimensional Fourier transform [16].

Fig. 6 shows a graphical relationship between the angular picture frame and angular resolution quantities in the image and Fourier domains. This figure serves as a visual for the concept that information in the Fourier domain far away from the origin represents the detail in the image, whereas information near the origin in the Fourier domain represents the large features in the image. This diagram assumes a finite aperture with a finite field of view viewing an infinite scene.

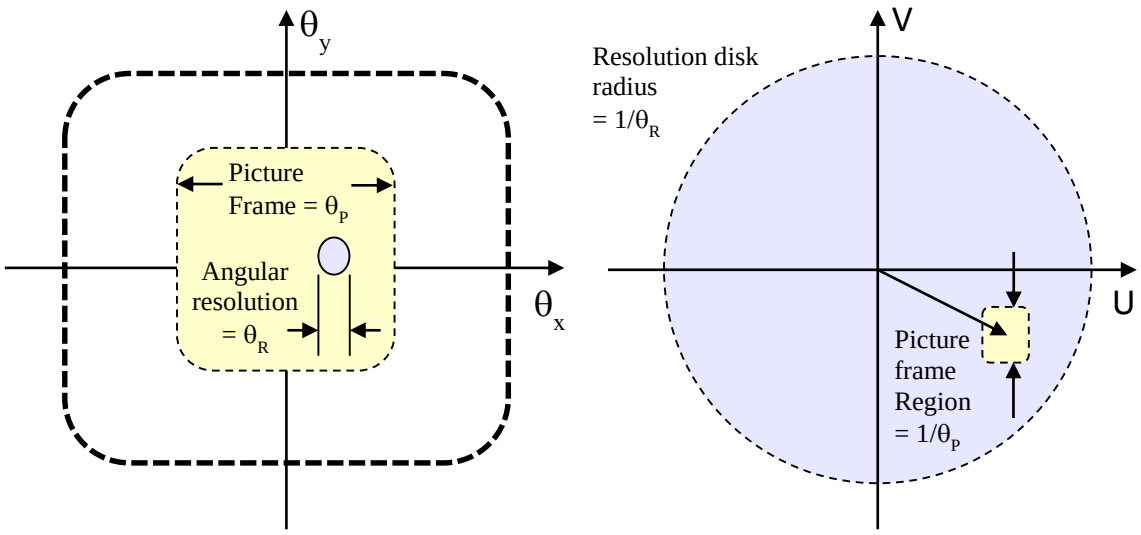


Fig. 6. Graphical representation of the θ angular spatial plane which contains the image and the Fourier UV wave number plane.

1.3. Optical Interferometry

Along with the maturation of the wave theory of light came applications of the theory to various fields of science, particularly astronomy. In 1868 Fizeau devised a method of measuring the angular dimensions of a star using two slits in front of a telescope. In 1890 Michelson successfully implemented Fizeau's idea to measure the diameter of the moons around Jupiter which marked the birth of stellar interferometry which has since taken several forms [8].

In general, an interferometer splits a wave field into two or more paths and recombines the paths in a controlled way. For example, the double-slit experiment takes the light from the first slit, splits it by passing it through the double slit mask, and combines the light from the double-slits on the observation plane. There are many types of interferometers that demonstrate various phenomena, many of which serve as technical demonstrations. Michelson devised the first interferometer with the purpose of making astronomical observations which marked the beginning of the stellar interferometry field.

The Michelson stellar interferometer, being one of the earliest, is quite simple [18]. Light from a star is collected by two separated apertures. The lights is brought from the two apertures to a central combiner. The combiner is essentially a telescope which focuses the light from the two apertures onto an observation plane where the interference pattern is visible. A major complication with the Michelson interferometer and other comparable interferometers is the precision required along the light paths. The two paths must have lengths accurate to within a small fraction of the wavelength of the light to prevent phase shifting which essentially blurs the interference pattern. This problem has been handled

satisfactorily but still poses significant technical and financial burdens [19]. The schematic is shown in Fig. 7. A Michelson interferometer was added to the Hooker telescope in 1920 shown in Fig. 8. A modern 437 meter optical Michelson interferometer is shown in Fig. 9 and is one of the largest in the world. It combines six beams from apertures in a Y configuration. There are dozens of types of optical interferometers that build upon the idea within the Michelson interferometer. Each has its own benefits, complications, and applications—enough that an exhaustive list here is unnecessary.

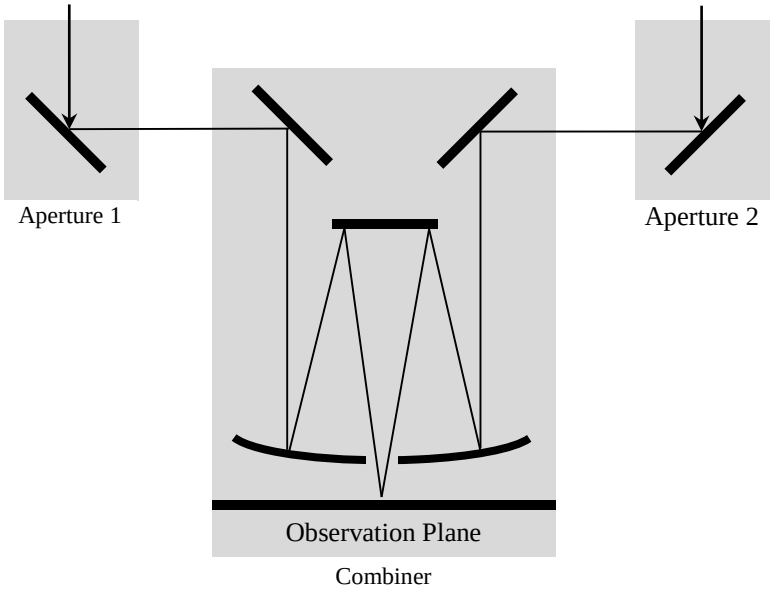


Fig. 7. Schematic of a Michelson stellar interferometer showing the incoming lights path to the focal point.

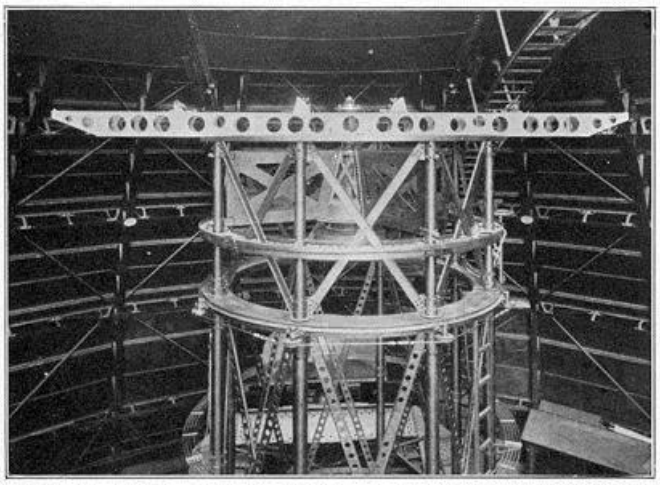


Fig. 8. Twenty-foot Michelson interferometer for measuring star diameters, attached to upper end of the skeleton tube of the 100-inch Hooker telescope [20].

Fig. 9. Navy Prototype Optical Interferometer, Anderson Mesa, Flagstaff [21].

A field of interferometry that aims to mitigate the necessity of precision optics as is needed for a Michelson interferometer is intensity correlation interferometry. These optically simpler interferometers capture incoming light at two apertures which use photodetectors to quantify the intensity of incident light. The photodetector measurements are then combined computationally instead of physically. This concept allows for apertures to be moved arbitrarily without the need for precise measurement of the path length. This technique was pioneered by Robert Hanbury Brown and Richard Twiss and was published in 1954 [22].

The Hanbury Brown and Twiss effect correlates the arrival of photons at two spatially separated photodetectors as shown in Fig. 10. The correlation between photon arrival times is proportional to the coherence of the light source [23]. This means that if laser light is used there is no correlation between the photon arrivals at the two detectors. If light from a thermal source is used, however, the correlation is non-zero and positive. The interferometer as shown in Fig. 10 is merely an autocorrelation bench-top apparatus for demonstrating the Hanbury Brown and Twiss effect.

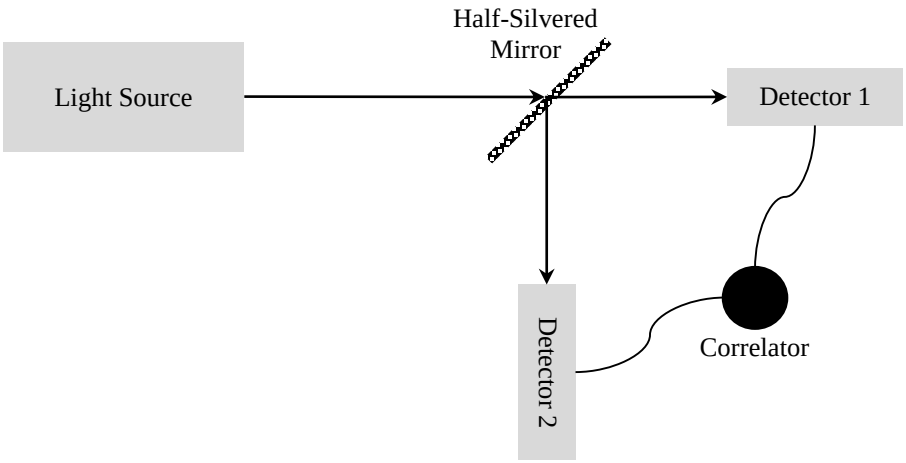


Fig. 10. Schematic of an intensity correlation interferometer.

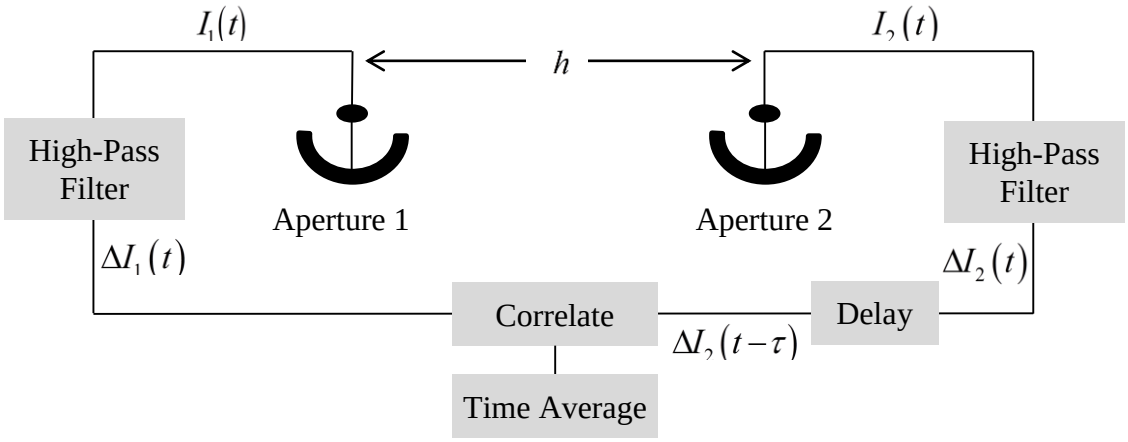


Fig. 11. Schematic of the stellar intensity correlation interferometer.

Fig. 12. Photograph of the Narrabri stellar interferometer used by Hanbury Brown and Twiss [24].

For application to stellar interferometry, the intensity interferometer is modified from its bench-top form as shown in Fig. 10. Two separated light collectors measure the total intensity of the incoming light using photodetectors [25, 26]. The intensities are the square of the complex field at the two spatial locations:

$$I_1(t) = E_1(t)E_1^*(t) \tag{1.25}$$

and

$$I_2(t) = E_2(t)E_2^*(t). \tag{1.26}$$

The intensity interferometer includes equipment to compute the time-averaged cross-correlation of the fluctuations of the two intensity measurements about their average values. Brown and Twiss' significant discovery was that the cross-correlation is proportional to the magnitude of the mutual coherence of the wave field at the relative position of the two telescopes. There is also a proportionality constant that relates the cross-correlation with the mutual coherence which is not discussed here. The mutual coherence is the magnitude of the quantity $J(\underline{u})$ in the Van Cittert-Zernike theorem. If the coherence magnitude is measured at sufficiently many relative positions of the telescopes (positions in the u-v plane), and if the phase of $J(\underline{u})$ can be estimated, an inverse Fourier transform can yield the image of the object. This explains the role of phase retrieval in intensity interferometry.

1.4. Applications of Phase Retrieval

The phase retrieval problem has applications in many fields and is not limited to imaging; however, only imaging applications are considered here. One of the most

common applications of phase retrieval is in crystallography. Crystallography refers to a method of observing the crystalline structure of a material. Crystallography has become commonplace in the analysis of large biomolecules such as proteins and for inorganic crystalline materials. Different wave fields can be used for crystallography such as x-rays, neutrons, and electrons. These are considered because the wavelength of these fields is less than the length of the atomic bonds in the specimen. The wave field is passed through the material where it interacts with the atoms leaving a diffraction pattern. The diffraction pattern is often observed with a film or array of discrete detectors. The phase retrieval process can then be used to determine the two or three dimensional arrangement of the atoms in the structure.

Coherent Diffractive Imaging (CDI) is another method of imaging which employs the phase retrieval process. This imaging method is typically applied to nano-scale structures which include nanotubes [27], nano-crystals [28], and material defects [29]. CDI uses either electromagnetic or electron beams to produce a diffraction pattern which requires phase retrieval to recover the shape of the structure.

A Transmission Electron Microscope (TEM) is typically capable of producing an image on a detector; however, when observing a crystal structure the signal-to-noise ratio can be increased by focusing the TEM to produce a diffraction pattern [30]. It then functions in a similar fashion as the CDI method.

Astronomical interferometry also presents an application for phase retrieval and can be posed as a macro-scale analog to the crystallography method. A distant source of coherent light produces a diffraction pattern via the propagation of light over long

distances as described by the Van Cittert-Zernike theorem. Intensity interferometers thus require a solution to the phase problem to recover the shape of the light source. Application of this idea is difficult because where the diffraction pattern in crystallography is very small, the usable diffraction pattern for astronomical imaging can be many kilometers in size. The practicality of building a data collection device is thus a topic of ongoing research.

Phase retrieval has also been implemented in the analysis of aberrations in optical devices. As an example, the aberrations in the Hubble space telescope were diagnosed and characterized through a phase retrieval process [31, 32]. This analysis was used to determine the optical transfer function of the Hubble telescope prior to its servicing. By finding the true optical transfer function, which differs from the designers' intended optical transfer function, some adaptive methods were devised to mitigate the blurriness induced by the lens' aberrations.

Within the work described here in Section 5 and in [4], the phase retrieval problem has been applied to stellar occultation. Within this problem, light from a distant star passes around an occulting asteroid. After passing the asteroid the resulting wave field is subject to diffraction. The resulting wave field can be observed by multiple observers as a diffraction pattern. Using this diffraction pattern to determine the geometry of the occulting asteroid yields a phase retrieval problem. There are many other forms of the phase retrieval problems but the ones discussed here serve as a brief, representative list.

2. EXISTING PHASE RETRIEVAL METHODS

Through research in the various applications of intensity interferometry, several methods of phase retrieval have been devised. Most require some form of *a priori* knowledge. If not to actually perform the computation, some knowledge of the image is required to know what algorithms are applicable. Researchers continue to seek a globally applicable algorithm which requires no *a priori* knowledge. This sections presents many of the current algorithms for phase retrieval and gives brief explanations of some of them. Some of these algorithms will be referred to in later sections.

To fully understand the difficulties in the phase retrieval problem an exact solution is considered in the following [33]. In this example consider a pixelated $N \times N$ image where the Fourier domain magnitude $\left|A(u,v)\right|$ is known. Since every pixel in the image is a square, the Fourier domain is the convolution of the Fourier transform of each pixel's independent Fourier transform. The Fourier transform of a single pixel at the origin with intensity I is

$$J(u,v) = I\,\mathrm{sinc}(u)\,\mathrm{sinc}(v). \tag{2.1}$$

The measured quantity $\left|A(u,v)\right|$ is thus a sum of all of the pixel's Fourier transforms taking into account the translation of each pixel relative to the origin of the image. The sum is thus

$$\left|A(u,v)\right|\exp\left(i\varphi(u,v)\right) = \sum_{k=1}^{N}\sum_{m=1}^{N}\left|A(k,m)\right|\exp\left(i\varphi(k,m)\right)\operatorname{sinc}(u-k)\operatorname{sinc}(v-m). \tag{2.2}$$

Evaluating the square of this expression gives a system of non-linear algebraic equations

$$\left|A(u,v)\right|^2 = \left[\sum_{k=1}^{N}\sum_{m=1}^{N}\left|A(k,m)\right|\operatorname{sinc}(u-k)\operatorname{sinc}(v-m)\cos\left(\varphi(k,m)\right)\right]^2 + \left[\sum_{k=1}^{N}\sum_{m=1}^{N}\left|A(k,m)\right|\operatorname{sinc}(u-k)\operatorname{sinc}(v-m)\sin\left(\varphi(k,m)\right)\right]^2. \tag{2.3}$$

This system has N^2 unknowns and N^2 solutions. Any one pixel can be constrained such that its phase is a specified value. As an example consider $\varphi(1,1) = 0$. The system of equations simplifies to

$$\left|A(u,v)\right|^2 = \left[\sum_{\substack{k=1\\k\neq 1,m\neq 1}}^{N}\sum_{m=1}^{N}\left|A(k,m)\right|\operatorname{sinc}(u-k)\operatorname{sinc}(v-m)\cos\left(\varphi(k,m)\right) + \left|A(1,1)\right|\operatorname{sinc}(u-1)\operatorname{sinc}(v-1)\right]^2 + \left[\sum_{\substack{k=1\\k\neq 1,m\neq 1}}^{N}\sum_{m=1}^{N}\left|A(k,m)\right|\operatorname{sinc}(u-k)\operatorname{sinc}(v-m)\sin\left(\varphi(k,m)\right)\right]^2. \tag{2.4}$$

This system of equations does have a unique solution but is still difficult to solve numerically.

The primary purpose of this discussion is to call attention to the non-uniqueness of a solution. Solving equation (2.3) is tantamount to factoring a polynomial in two variables. This means the Fundamental Theorem of Algebra does not hold, and thus non-unique solutions can exist. Fortunately the probability of encountering Fourier magnitude data in

field observations that permits these ambiguities is low. The most common ambiguities correspond to rotations, translations, and super-positions of solutions. Due to the high dimensionality, non-linearity, and non-uniqueness of a solution, the direct computation of the phase from a relationship such as equation (2.3) is infeasible.

2.1. The Error-Reduction and Hybrid Input-Output Methods

An iterative algorithm was developed by Gerchberg and Saxton which was later improved by Fienup called the error-reduction (ER) method [34, 35]. This algorithm, shown graphically in Fig. 13, exploits the discrete nature of an image and the ease with which a discrete 2-d image can be Fourier transformed. In this work the 2-d discrete Fourier transform has the form

$$G(u,v) = \frac{1}{MN}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1} g(x,y)\exp\left(-i2\pi\left(\frac{ux}{M}+\frac{vy}{N}\right)\right) \tag{2.5}$$

and the inverse is

$$g(x,y) = \sum_{u=0}^{M-1}\sum_{v=0}^{N-1} G(u,v)\exp\left(i2\pi\left(\frac{ux}{M}+\frac{vy}{N}\right)\right). \tag{2.6}$$

An initial guess of the image g_0 is formulated. The image g_k at iteration k is Fourier transformed to arrive at its representation in the Fourier domain G_k. The Fourier domain pixel-wise magnitude is constrained to the given, measured value F giving

$$G'_k = |F|\exp\left(i\arg\left(G_k\right)\right). \tag{2.7}$$

The result contains the phase value from the image and the measured modulus values. This result is inverse Fourier transformed back to the image domain yielding g_i'. Some region

of the image γ is known *a priori* to be background, i.e. zero pixel values, and the image pixels are real and positive. This knowledge is applied via the constraint

$$g_{k+1}(x,y) = \begin{cases} g'_k(x,y), & (x,y) \notin \gamma \\ 0, & \text{otherwise} \end{cases} \tag{2.8}$$

which gives the image for the next iteration.

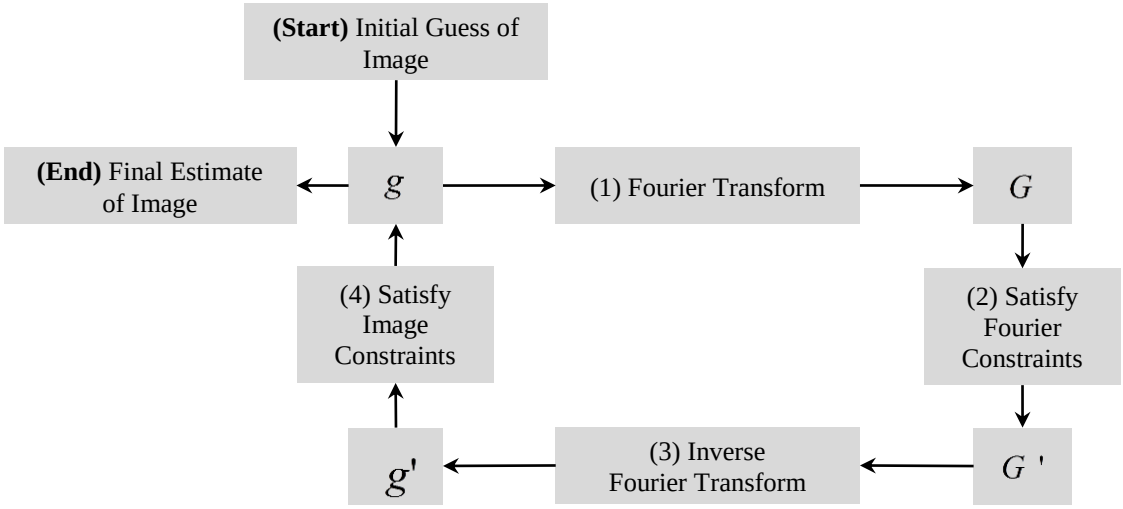


Fig. 13. Block diagram of the error-reduction method.

Fienup defined an error metric which quantifies the violations of the background nullity in the form [35]

$$e^2 = \frac{\int_\gamma \left[g_k' \right]^2 d\mathbf{x}}{\int \left[g_k' \right]^2 d\mathbf{x}}. \tag{2.9}$$

It can be shown that this error quantity cannot increase at each iteration which gives the method its name [34, 36].

Using Parseval's theorem the Fourier domain error at iteration $k+1$ can be stated as

$$\begin{aligned} E_{F,k+1}^2 &= N^{-2} \int \left| G_{k+1} - G'_{k+1} \right|^2 d\mathbf{x} \\ &= \int \left| g_{k+1} - g'_{k+1} \right|^2 d\mathbf{x}. \end{aligned} \tag{2.10}$$

Similarly, the image domain error at iteration k can be stated as

$$\begin{aligned} E_{0,k}^2 &= \int \left| g_{k+1} - g'_k \right|^2 d\mathbf{x} \\ &= N^{-2} \int \left| G_{k+1} - G'_k \right|^2 d\mathbf{x}. \end{aligned} \tag{2.11}$$

Because the image is constrained in the transition from g'_k to g_{k+1}, it holds that

$$\left| G_{k+1} - G'_{k+1} \right| \leq \left| G_{k+1} - G'_k \right| \tag{2.12}$$

which implies

$$E_{F,k+1}^2 \leq E_{0,k}^2. \tag{2.13}$$

Using the equalities resulting from Parseval's theorem in equations (2.10) and (2.11), this expression can be expanded to

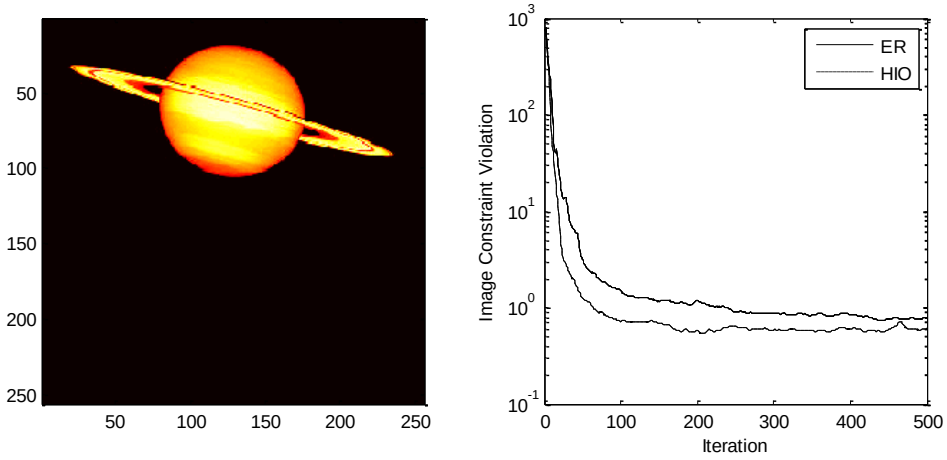
$$E_{0,k+1}^2 \leq E_{F,k+1}^2 \leq E_{0,k}^2 \leq E_{F,k}^2. \tag{2.14}$$

This expression shows the apparent convergence of the algorithm because the error cannot increase. The equalities, however, are the caveat. Experience shows that although the algorithm is typically convergent, convergence can take an impractical number of iterations. The algorithm is said to stagnate at local minima of this error metric which the equalities allow.

To overcome the stagnation problems encountered by the ER method, Fienup devised some modifications to the method which culminated in the Hybrid Input-Output method (HIO). This method employs a feedback parameter β to push the pixel values in γ to zero without rigidly constraining them. The image constraint has the form

$$g_{k+1}(x,y) = \begin{cases} g'_k(x,y), & (x,y) \notin \gamma \\ g_k(x,y) - \beta g'_k(x,y), & (x,y) \in \gamma. \end{cases} \tag{2.15}$$

In a diagnostic cases, the HIO typically outperforms the ER method. Given a proper estimate of γ the HIO error will typically decrease more quickly than the ER error. A comparison between the ER and HIO is shown in Fig. 14 for a noiseless test case which shows that the HIO converges more quickly. Since the Fourier modulus data is noiseless, both converge to about the same error magnitude.



a) Sample image of Saturn b) Error results for the ER and HIO

Fig. 14. Comparison of the error in a test case of the ER and HIO on the 256x256 image of Saturn with noiseless Fourier modulus data.

The HIO requires knowledge of the support of the true image, the region which has zero pixel values, γ. The exact background region in practice is not known and must thus be estimated. There are several methods which can assist in this estimation.

Fienup, soon after developing the HIO, devised a method of estimating the support of the image from the support of its autocorrelation [37] but not uniquely [38]. He concedes if the support is estimated nearly exactly, the HIO will likely stagnate more often than if a conservative estimate is made in which the foreground is assumed larger than is actually is. He explains that if the partially converged image happens to be translated such that part of the foreground lies in γ it will be clipped which will cause stagnation. He suggests that leaving the foreground larger than needed typically provides the best convergence rates. He later stated that the autocorrelation is more useful for defining an initial guess of the image rather than the support region. In [38] he says to take the autocorrelation, decrease its size by a factor of two, threshold the values, and multiply by a uniform random number. This would define an initial image for the HIO and the threshold would give a conservative estimate of the γ region.

Another method of determining the support region was proposed by [39] which adapts to the image at each iteration. The algorithm blurs the image and determines γ based on a threshold of the blurred image. The threshold is slowly decreased with subsequent iterations until the image is adequately reconstructed. An advantage to this type of estimation lies in the existence of multiple foregrounds. This can be advantageous in many fields such as crystallography, x-ray imaging, and astronomy.

It has been shown that the HIO method's performance can be theoretically quantified based on the size of the background region. This can be called the amount of 'over sampling.' With the definition of the ratio

$$\sigma = \frac{\text{total pixel number}}{\text{unknown-valued pixel number}}, \tag{2.16}$$

the HIO can be shown to be convergent in theory for $\sigma > \sqrt{2}$ in each dimension of the 2-d image or $\sigma > 2$ for the entire image [40]. This requirement imposes constraints on the non-linear system in equation (2.3), i.e. some pixel values in the system are known to be zero. It is also shown in [40] that oversampling aides in handling noise in the Fourier modulus data as will be investigated later.

In practice the Fourier modulus data contains some amount of noise. A model of this noise,

$$\left|F(u,v)\right| = \left|F(-u,-v)\right| = \left|F_{True}(u,v)\left(1 + N_1[0,\sigma] + N_2[0,\sigma]\,i\right)\right|, \tag{2.17}$$

corrupts the true Fourier modulus with Gaussian noise of standard deviation σ and scaled by the true modulus [1]. It should be noted that the two Gaussians, $N_1[\cdot]$ and $N_2[\cdot]$ are statistically independent. This noise model is based on that used in intensity correlation interferometry when analyzing the statistics of the coherence magnitude measurements based on the Hanbury Brown and Twiss work [41, 42]. The model is similar to those used in [35, 38, 43, 44, 45]. With the addition of Fourier modulus noise with $\sigma = 0.05$, the HIO achieves about an order-of-magnitude smaller error than the ER achieves as shown in Fig. 15. With larger amounts of noise, however, the HIO error oscillates and does not converge

[1]. This oscillation behavior will be further explained in the next section. This shows the necessity for a means of filtering the noise to achieve convergence.

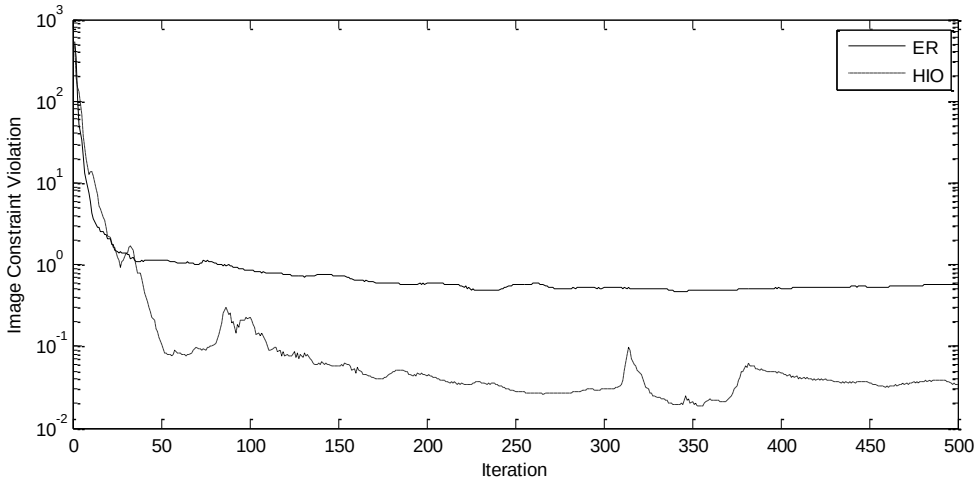


Fig. 15. Comparison of the error in a test case of the ER and HIO on the image of Saturn with noisy Fourier modulus data, 5%.

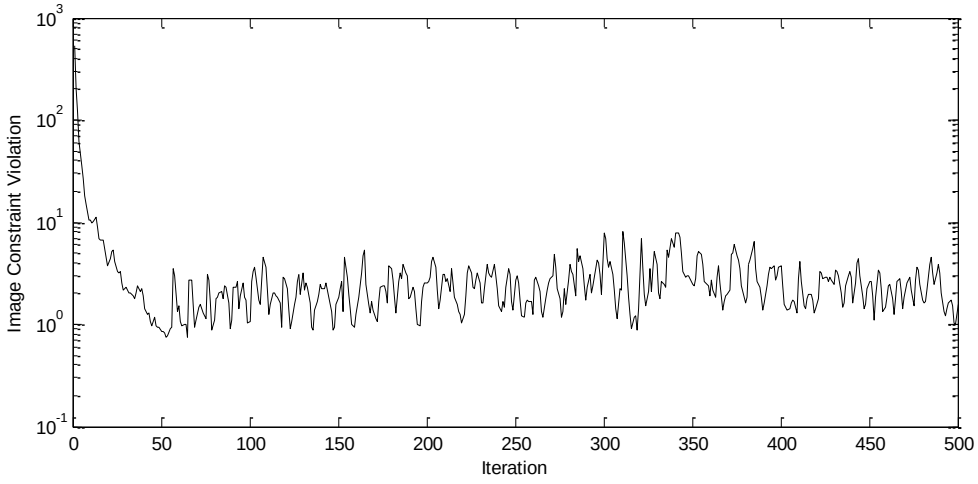


Fig. 16. Result of the HIO in a test case on the image of Saturn with noisy Fourier modulus data, 25%.

2.2. Phase Retrieval Algorithm Comparisons

While the HIO developed by Fienup is the field standard for comparison, many other algorithms have been proposed which build upon the HIO. A somewhat comprehensive list and comparison was performed by Marchesini [46]. His article follows the projection notation as used by some authors. This notation compactly represents an iteration of the algorithm using operators which can be nested. To introduce the notation the Error-Reduction method will first be shown followed by the rest of Marchesini's list plus some others.

As previously explained, the error-reduction method takes an image, Fourier transforms it, constrains the Fourier domain modulus, inverse Fourier transforms the image, and constrains some portion of the image to arrive at the next iteration's image. Each step of this process can be expressed as a projection operation. The image is denoted by $g(x,y)$ and its Fourier domain is denoted by $G(u,v)$. The Fourier domain modulus constraint can thus be expressed as

$$G_k'(u,v) = \tilde{P}_m G_k(u,v) = \left|F(u,v)\right| \exp\left(i \arg\left(G_k(u,v)\right)\right) \tag{2.18}$$

where $F(u,v)$ is the known magnitude. This operator can be combined with the Fourier transform mapping FT such that the operation P_m takes an image, Fourier transforms it, constrains the magnitude, and inverse Fourier transforms it. This gives

$$P_m = FT^{-1}\tilde{P}_m FT. \tag{2.19}$$

Similarly the image domain constraint can be formulated as

$$g_{k+1}(x,y) = P_s g'_k(x,y) = \begin{cases} g'_k(x,y) & \text{if } (x,y) \notin \gamma(x,y) \\ 0 & \text{otherwise.} \end{cases} \tag{2.20}$$

Combining these two operators gives a complete iteration of the error-reduction method as

$$g_{k+1}(x,y) = P_s P_m g_k(x,y). \tag{2.21}$$

Most of the phase retrieval methods can be expressed in this notation although some can be difficult to understand in the projection notation as opposed to the pixel-wise notation.

Table 1. List of projective phase retrieval algorithms.

Algorithm	Iteration
Error Reduction (ER)	$g_{k+1} = P_s P_m g_k$
Solvent Flipping (SF)	$g_{k+1} = R_s P_m g_k$
Hybrid Input-Output (HIO)	$g_{k+1} = \begin{cases} P_m g_k & (x,y) \notin \gamma \\ (I - \beta P_m) g_k & (x,y) \in \gamma \end{cases}$
Different Map (DM)	$g_{k+1} = \left\{ I + \beta P_s \left[(1+\gamma_s) P_m - \gamma_s I \right] - \beta P_m \left[(1+\gamma_m) P_s - \gamma_m I \right] \right\} g_k$
Averaged Successive Reflection (ASR)	$g_{k+1} = \frac{1}{2} \left[R_s R_m + I \right] g_k$
Hybrid Projection Reflection (HPR)	$g_{k+1} = \frac{1}{2} \left[R_s \left(R_m + (\beta - 1) P_m \right) + I + (1-\beta) P_m \right] g_k$
Random Averaged Alternating Reflector (RAAR)	$g_{k+1} = \left[\frac{1}{2} \beta \left(R_s R_m + I \right) + (1-\beta) P_m \right] g_k$
Levi-Stark method	$G_{k+1} = (1-\lambda) G_k + FT \left(\lambda P_s P_m g_k \right)$

Table 2. Projection operators used in phase retrieval methods.

Projection	Formula
Image support constraint	$P_s = \begin{cases} 1 & (x,y) \notin \gamma \\ 0 & (x,y) \in \gamma \end{cases}$
Fourier modulus constraint	$P_m = FT^{-1} \left[\left| F(u,v) \right| \exp\left(i \arg\left(G_k(u,v) \right) \right) \right] FT$
Image support constraint reflector	$R_s = I - 2P_s$
Fourier modulus constraint reflector	$R_m = I - 2P_m$

Shown in Table 1 is a list of phase retrieval algorithms stated in terms of the projections listed in

Table 2. This first method not discussed yet is the Solvent Flipping [47] algorithm which has the iteration

$$g_{k+1} = R_s P_m g_k . \tag{2.22}$$

Expanding the operators as

$$g_{k+1} = \left\lceil \begin{cases} 1, & \text{if } (x,y) \notin \gamma(x,y) \\ -1, & \text{otherwise} \end{cases} \right\rfloor g'_k \tag{2.23}$$

helps to see the correlation with the error reduction method. The comparison is most evident when the projections are viewed in a graphical representation of the domains and projections as shown in Fig. 17 and Fig. 18. These examples show a two degree-of-freedom case where the two dimensions (x and y) are analogous to two pixels in an image. The Fourier transform operators in the P_m projection has no visible effect in these visualizations. The figure shows the domain S which satisfies the image support constraint and the domain M which satisfies the Fourier modulus constraint. This formulation of the phase retrieval problem can employ ideas from fields unrelated to phase retrieval, such as the von Neumann algorithm in set theory [48], because the problem is reduced to finding the intersection of two sets [49].

Each of the phase retrieval methods can be graphically represented in the 2D fashion shown in Fig. 17. All of the subsequent visualizations for the phase retrieval methods discussed here are presented with the same scaling. Since the Cartesian

coordinates are meaningless, however, numerical ticks marks are not shown on the plots and units are not shown.

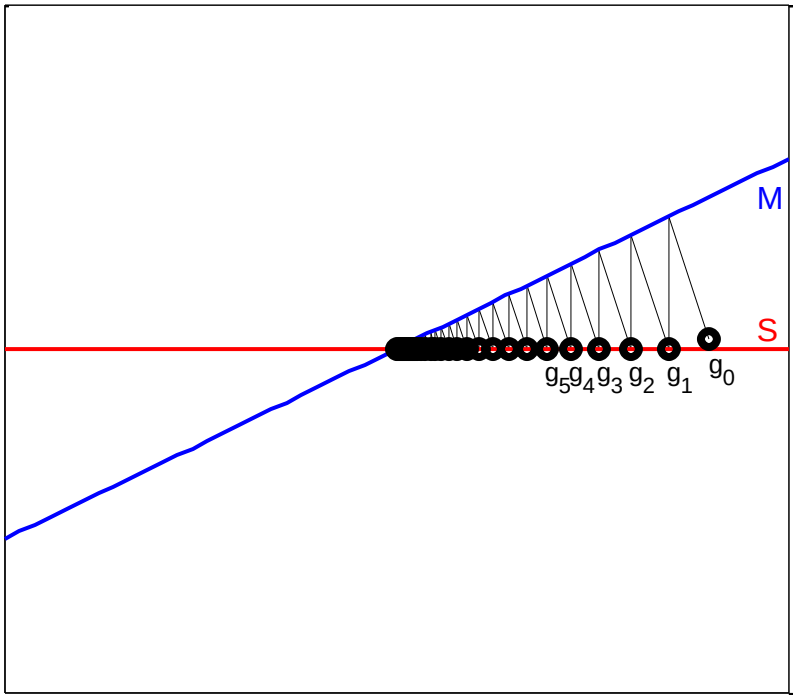


Fig. 17. Visualization of the Error Reduction method's domains and projections [39].

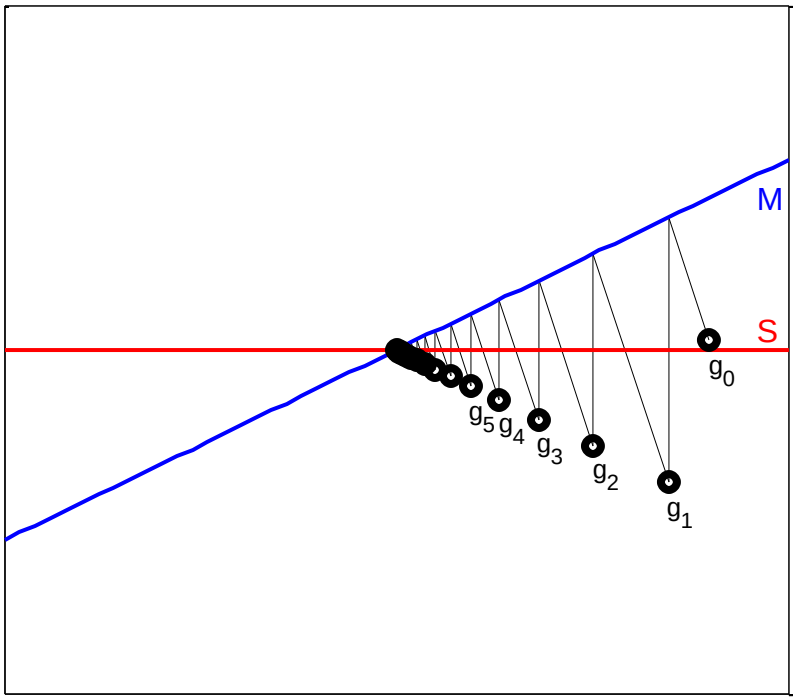


Fig. 18. Visualization of the Solvent Flipping method's domains and projections [39].

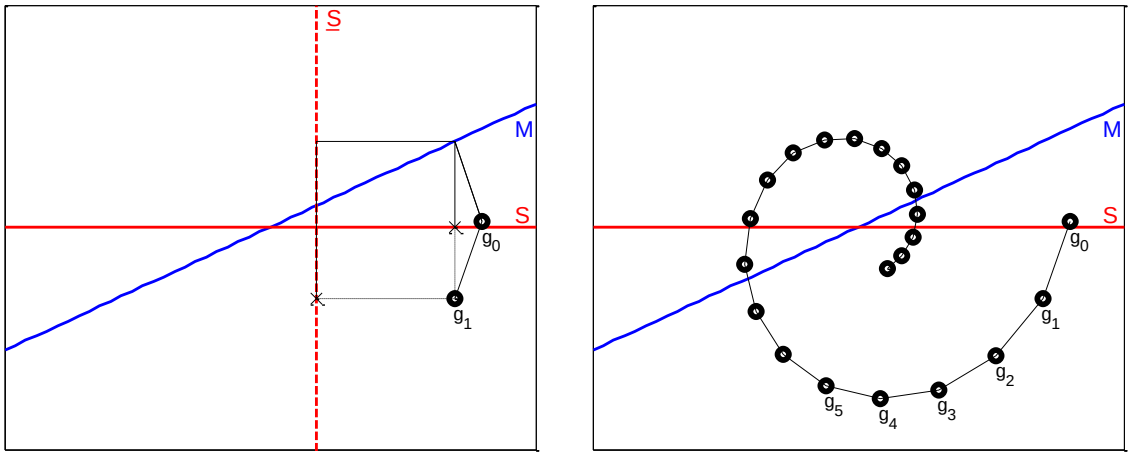
The error reduction method starts with an initial guess that satisfies neither constraint, projects onto the modulus domain, and projects back onto the support constraint domain. The resulting image, g_1, is closer to the intersection of the two domains than the initial guess. Each successive projection yields a result closer to the intersection than the previous result.

The solvent flipping method follows similar projections except the projection from the modulus domain overshoots the support domain and lands on a reflection of the modulus domain. The benefit is a larger step is taken toward the intersection of the two domains. The convergence is thus faster for the SF than the ER.

At first glance in the pixel-wise formula, the HIO appears to have a very similar form to the ER. In the projection form, however, they are very different. Firstly, an exact projection onto the support domain is never made. Rather, a step is taken of the form

$$g_{k+1} = \begin{cases} P_m g_k & (x, y) \in \gamma \\ (I - \beta P_m) g_k & (x, y) \notin \gamma. \end{cases} \tag{2.24}$$

Secondly, after the projection onto the Fourier domain a split occurs. A portion of the image is projected onto the support domain and a portion is projected onto a space $\underline{S}$ orthogonal to the support domain as shown in Fig. 19a. The two projections—one onto the support domain and one onto the domain orthogonal to the support domain—determine components of the final image. In the figure, these are represented as the horizontal and vertical components. The overall progress shown in Fig. 19b shows that a spiral is formed around the intersection of S and M. The spiral somewhat explains the behavior seen previously in which the HIO oscillated between satisfying the modulus and the image constraints in the presence of noise. The spiral turns into a circle around the intersection for some noise levels. For higher noise levels the algorithm does not converge. The effect of noise will be discussed further later.



a) HIO Iteration projections

b) HIO method

Fig. 19. Visualization of the HIO iteration projections and overall projection progress.

The Difference Map (DM) algorithm is the culmination of several simpler algorithms which are applied to the phase retrieval problem which has the form

$$g_{k+1} = \left\{ I + \beta P_s \left[(1+\gamma_s) P_m - \gamma_s I \right] - \beta P_m \left[(1+\gamma_m) P_s - \gamma_m I \right] \right\} g_k . \tag{2.25}$$

The parameters γ_s and γ_m are $\gamma_s = -\beta^{-1}$ and $\gamma_m = \beta^{-1}$ to achieve the optimal step [50]. A visualization of this method is shown in Fig. 20. It has very slow convergence and spirals many times.

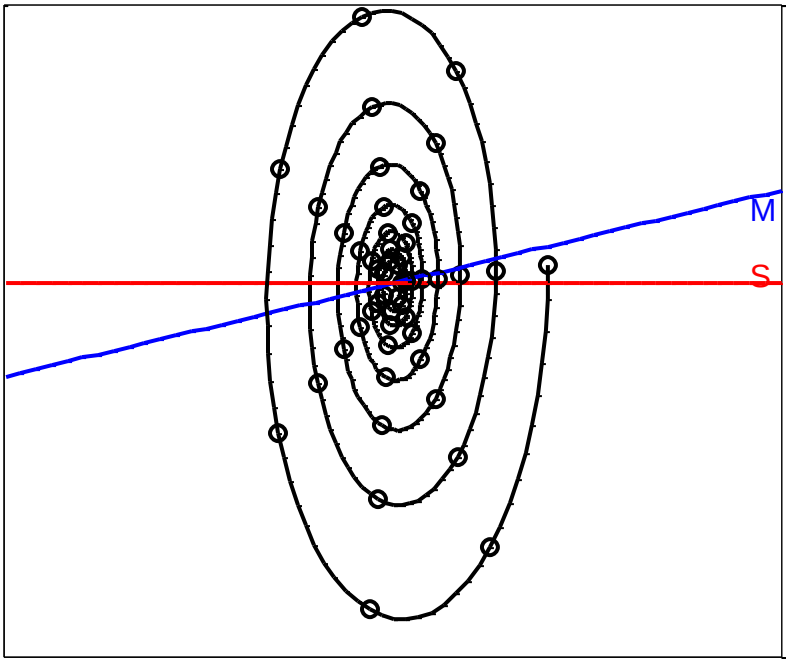


Fig. 20. Visualization of the Difference Map method's domains and projections.

The Averaged Successive Reflections (ASR) method performs the ER projections but reflects them and averages the resulting image with the original image in the iteration. Explicitly the method has the form

$$g_{k+1} = \frac{1}{2}\left[R_s R_m + I\right] g_k . \tag{2.26}$$

The visualization of the projections are shown in Fig. 21. The solid lines show the projections $R_s R_m$. The dotted line shows the region between the original image of the iterations and the projected image which are averaged yielding the new iteration on the center of the dotted line. As the iterations continue the image will make a spiral towards the intersections of M and S in a similar fashion as the HIO.

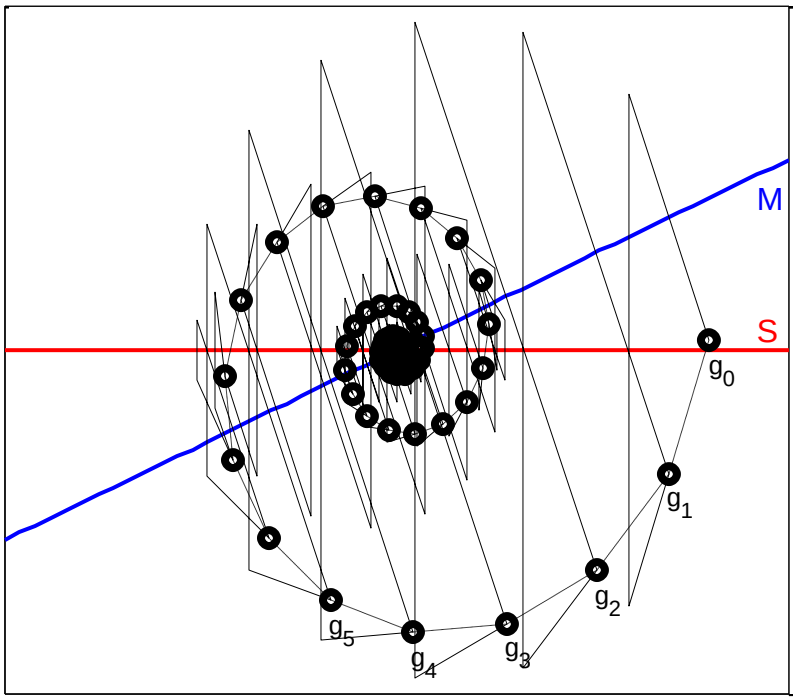


Fig. 21. Visualization of the Averaged Successive Reflections method's domains and projections.

The Hybrid Projection Reflection (HPR) algorithm,

$$g_{k+1} = \frac{1}{2}\left[R_s \left(R_m + \left(\beta - 1 \right) P_m \right) + I + \left(1 - \beta \right) P_m \right] g_k \,, \tag{2.27}$$

is based on the ASR but includes a relaxation with the parameter β . The relaxation helps to increase the convergence rate by forcing the spiral inwards faster. A typical value for β is about 0.9. This algorithm is a special single parameter relaxation of the DM algorithm [51]. Fig. 22 shows the HPR projections with an exaggerated β of 0.3 to clearly show the comparison of the convergence with the ASR.

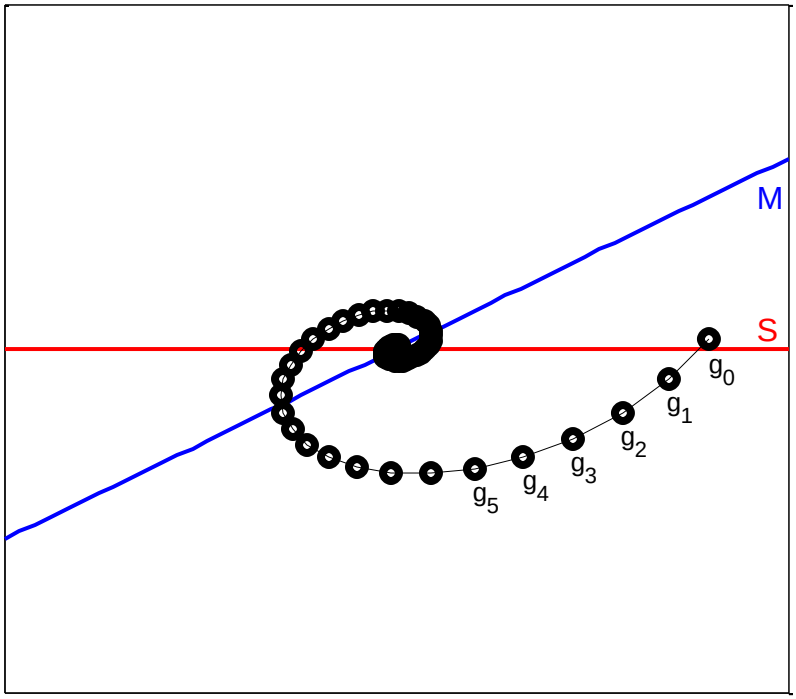


Fig. 22. Visualization of the Hybrid Projection Reflection method's domains and projections.

The Relaxed Averaged Alternating Reflectors (RAAR) algorithm,

$$g_{k+1} = \left[\frac{1}{2} \beta \left(R_s R_m + I \right) + \left(1 - \beta \right) P_m \right] g_k \,, \tag{2.28}$$

is based on the HPR [51]. The authors show some analytical justification for the RAAR which is not available for the HIO, HPR, or DM. Although the RAAR is based on the HIO, HPR, and DM, its projection takes a quite different path as shown in Fig. 23. The RAAR has much faster convergence properties than its predecessors. While the HIO is considered the basis for comparison in current literature, the RAAR appears to provide the most favorable and flexible performance of the phase retrieval methods listed here.

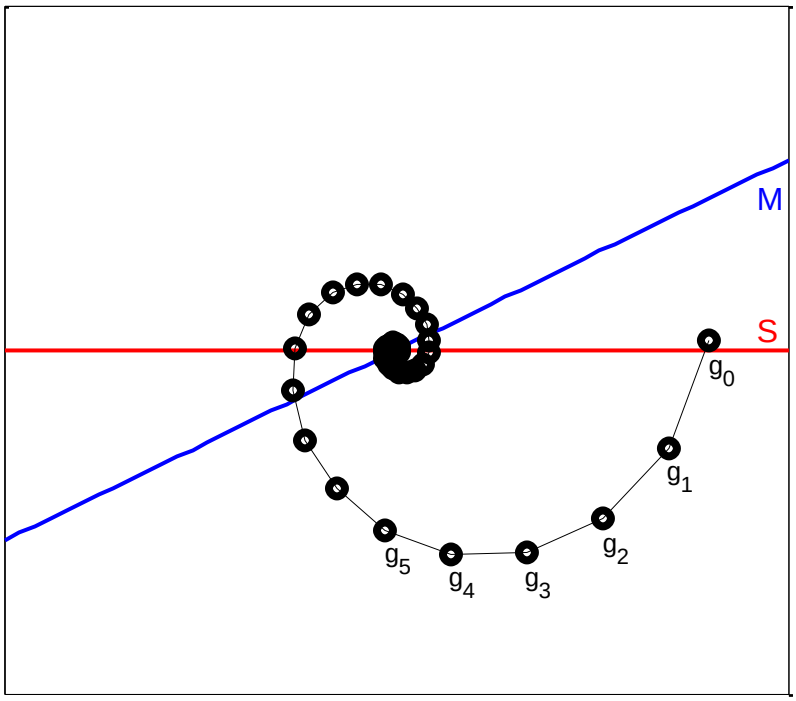


Fig. 23. Visualization of the Random Averaged Alternating Reflections method's domains and projections.

In a similar fashion as the example in [46], these algorithms can be compared side-by-side by giving all the same initial condition and comparable parameters. In Fig. 24 all of the methods have $\beta = 0.9$. Shown are the two linear domains S and M which intersect. The ER and SF exhibit the expected straight line behavior whereas the others spiral to the intersection. Fig. 25 shows the distance of each method from the origin at each iteration. The SF and ER typically show the fastest convergence; however, performance of each algorithm changes based on the parameters and initial conditions. A general trend thus cannot really be determined.

The previous comparison gives insight into the performance of the algorithms, however, the example shows an ideal case. In the phase retrieval problem the modulus domain is a non-convex set, so to easily visualize the constraint consider two semi-circles as shown in Fig. 26 [46]. The methods start near a local minimum. The ER, SF, RAAR, and Levi-Start methods stagnate at the local minimum, whereas the other methods are able to find the global minimum—the intersection. The DM shows an oscillation when transitioning from escaping the local minimum to approaching the global minimum which sometimes reveals an instability. It often stagnates in the oscillation far away from the intersection. Once the methods get close to the intersection, the spiral behavior appears as occurred in the linear domain case.

The previous examples neglected the positivity constraint which exists in the phase retrieval problem. If the support set imposes a constraint both in the vertical and horizontal axes, the algorithm's behavior changes slightly as shown in Fig. 27a. This is analogous to how the P_s operator imposes both finite support and positivity. The methods converge

slightly faster with the positivity constraint than without as shown in in Fig. 27b. Fig. 28 shows how near the intersection the trajectories follow spiral patterns similar to those shown in Fig. 24.

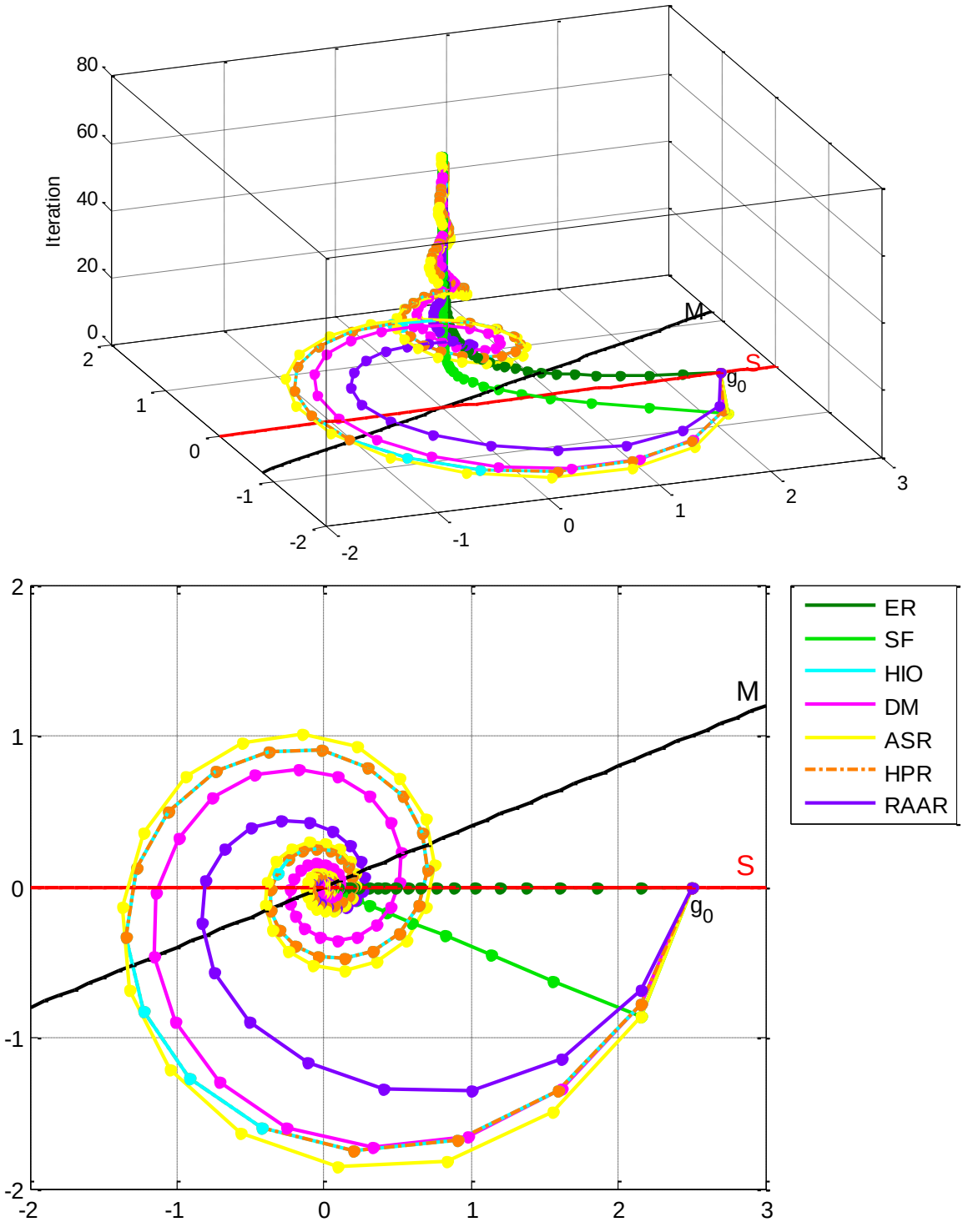


Fig. 24. Comparison of projective algorithms seeking the intersection of linear, intersecting domains. Markers are placed every 10 iterations on each path. (The HIO and HPR methods overlap.)

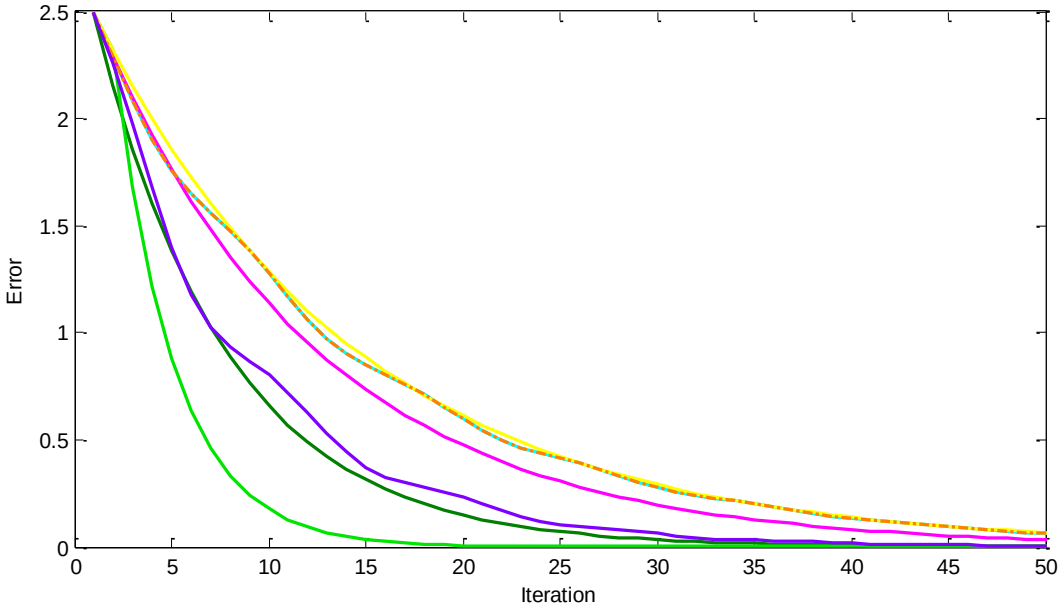
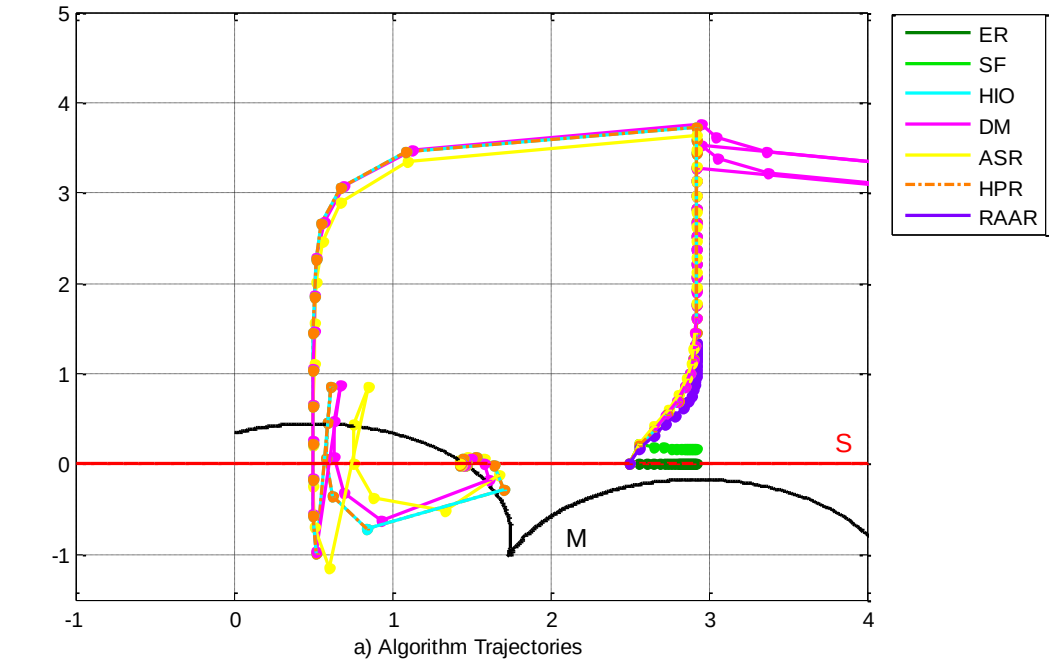
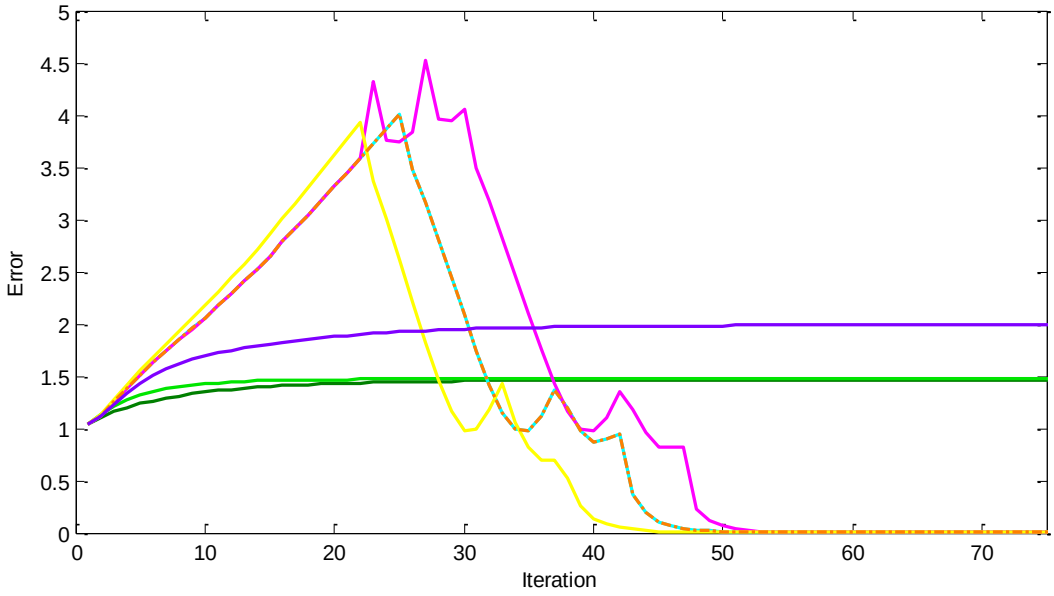


Fig. 25. Relative error, distance from intersection, of the comparisons in Fig. 24.

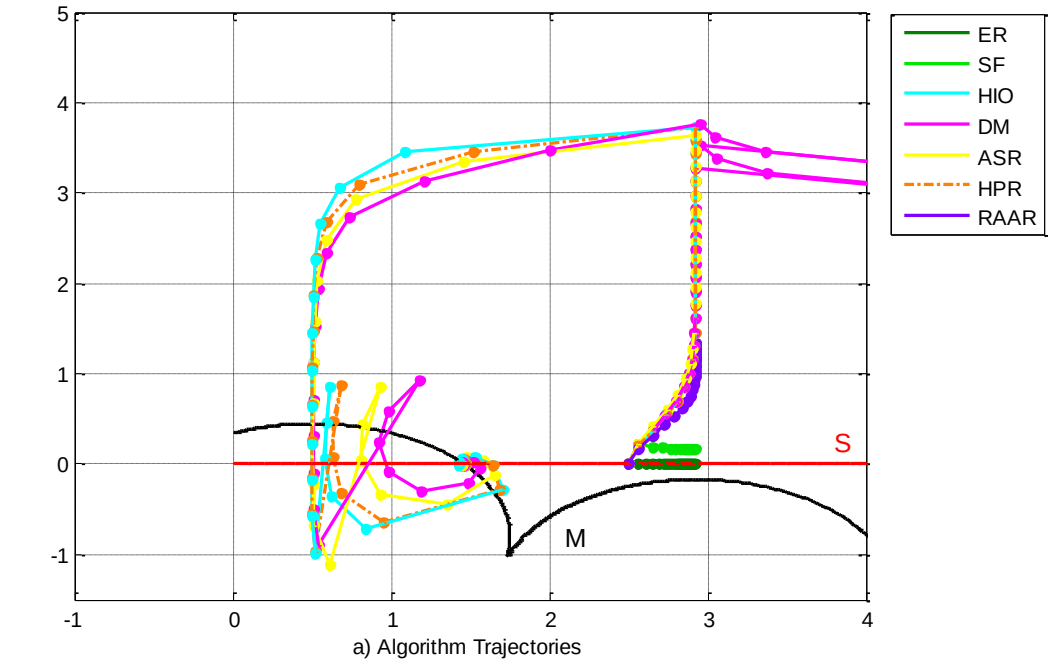


a) Algorithm Trajectories

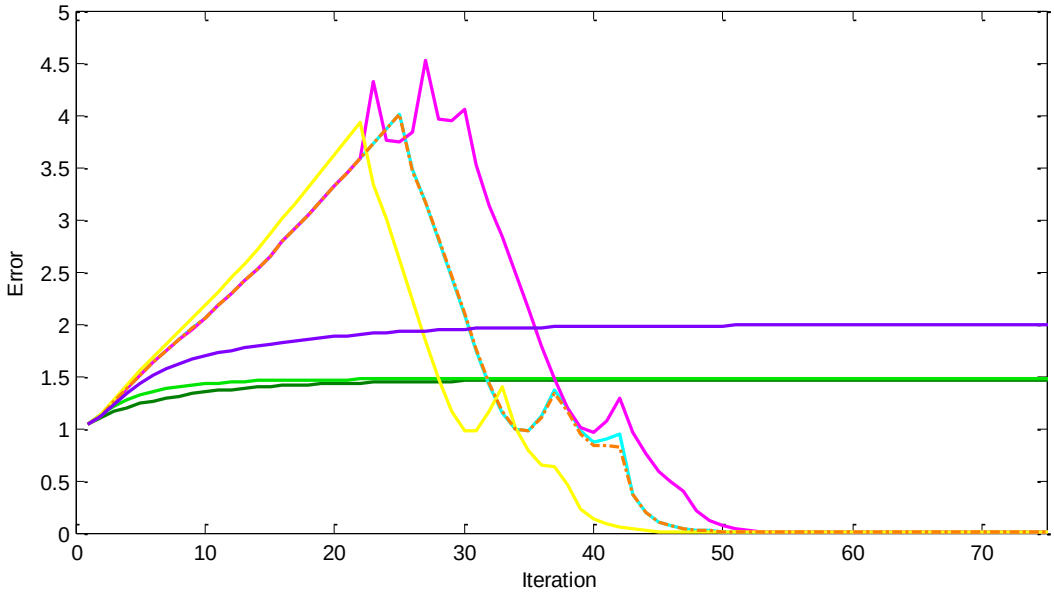


b) Distance from the global minimum.

Fig. 26. Comparison of projective algorithms seeking the intersection of a non-convex domain and an intersecting linear domain.



a) Algorithm Trajectories



b) Distance from the global minimum.

Fig. 27. Comparison of projective algorithms seeking the intersection of a non-convex and an intersecting linear domain with a positivity constraint.

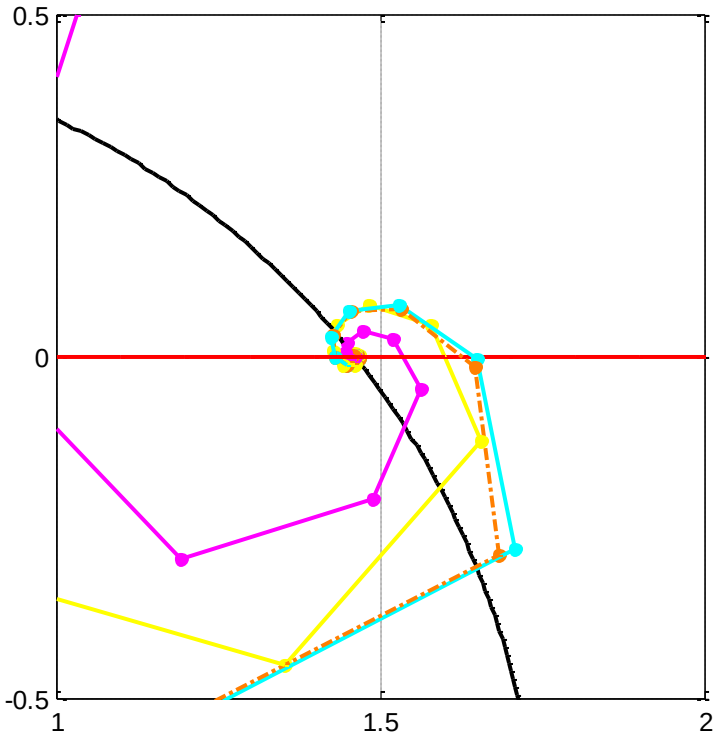


Fig. 28. Zoomed view of the intersection in Fig. 27.

3. THE CONSTRAINT RELAXATION ALGORITHM

Many phase retrieval methods have been developed with the intent of escaping local minima and finding the intersection of the Fourier modulus and image support domains. Not much attention, however, has been placed on filtering noise from the Fourier modulus data. As discussed previously, referring to equation (2.17), the noise in the Fourier modulus can be modeled as

$$\left|F\left(u,v\right)\right| = \left|F\left(-u,-v\right)\right| = \left|F_{True}\left(u,v\right)\left(1 + N_1\left[0,\sigma\right] + N_2\left[0,\sigma\right]i\right)\right|. \tag{3.1}$$

The difference between $\left|F\left(u,v\right)\right|$ and $\left|F_{True}\left(u,v\right)\right|$ is enough to make the Fourier modulus and image support domain not intersect, and it was shown in Section 2.1 that phase retrieval algorithms are sensitive to this discrepancy. A rare few phase retrieval algorithms consider the noise issue directly.

The method developed by Levi & Stark allows the modulus constraint to drift with each subsequent iteration [52]. Instead of constraining the Fourier modulus to the given $\left|F\left(u,v\right)\right|$ they use the form

$$\left|G_{k+1}\left(u,v\right)\right| = \left(1-\lambda\right)\left|G_k\left(u,v\right)\right| + \lambda\left|G_{k-1}\left(u,v\right)\right|. \tag{3.2}$$

This method however has been observed to either diverge or become trapped at local minima. Kohl developed a method where the measured Fourier modulus, $\left|F\left(u,v\right)\right|$ is mixed with the current iteration's Fourier modulus in the form

$$G_k'\left(u,v\right) = \left(1-\lambda_A\right)G_k\left(u,v\right) + \lambda_A\left|F\left(u,v\right)\right|\exp\left(i\arg\left(G_k\left(u,v\right)\right)\right) \tag{3.3}$$

where λ_A is randomly varied at every iteration [53]. This method has the effect of filtering some amount of noise; however, it was not developed with this intent. Because of the random quantity, it is difficult to analyze or give a clear rationale for its behavior. Liu developed a method where the noise is filtered by accurately knowing the statistics of the noise [43, 54]. While this method gives a unique rationale based on filtering the noise, its effectiveness is questionable because of the need to estimate the noise in the modulus quantitatively. Its performance has also not been observed to be more favorable than the original HIO method.

3.1. Effects and Filtering of Noise in Phase Retrieval

To gain insight into the effect noise has on iterative phase retrieval consider the noisy Fourier modulus

$$\tilde{G}(u,v) = F_{True}(u,v) + \sum_{(a,b)} \Delta(a,b)\,\delta(u-a, v-b). \tag{3.4}$$

This noise model places a perturbation of magnitude Δ at the location (a,b). Note that

$$\Delta(u,v) = F(u,v) - F_{True}(u,v). \tag{3.5}$$

This noise model and the following analysis are general in the sense that the statistics of Δ do not need to be specified.

Due to the linearity of the Fourier transform, during transformation the $\Delta(a,b)$ terms are independent of each other and independent of the image data F_{True}. The effect of each $\Delta(a,b)$ term can thus be analyzed separately so let $\Delta(a,b) = \Delta$. In the general iterative phase retrieval algorithm shown in Fig. 13,

$$G'(u,v) = \Delta\delta(u-a, v-b). \tag{3.6}$$

If the image has extent $x = [0, M-1]$ and $y = [0, N-1]$, the inverse Fourier transform is

$$g'(x,y) = \frac{\Delta}{MN}\exp\left(i2\pi\left(\frac{ax}{M}+\frac{by}{N}\right)\right). \tag{3.7}$$

Next, in the general algorithm an image constraint is applied. Namely, the pixels are real valued, positive, and have finite support. If the foreground of the image is a distance A from the left and right and a distance B from the top and bottom, the image has nonzero pixels for $x = [A, M-1-A]$ and $y = [B, N-1-B]$. The constrained image is thus

$$\begin{aligned} g(x,y) &= \begin{cases} |g'(x,y)| & A \le m \le M-1-A \;\&\; B \le n \le N-1 \\ 0 & \text{otherwise} \end{cases} \\ &= \frac{\Delta}{MN}\begin{cases} 1 & A \le m \le M-1-A \;\&\; B \le n \le N-1 \\ 0 & \text{otherwise.} \end{cases} \end{aligned} \tag{3.8}$$

The discrete Fourier transform of this expression is

$$\begin{aligned} G(u,v) = \frac{\Delta}{MN}\exp\left[i\pi\left(u\frac{M-2A-1}{M}+v\frac{N-2B-1}{N}\right)\right] \\ \times \frac{\sin\left(\frac{M-2A}{M}\pi u\right)}{\sin\left(\frac{\pi u}{M}\right)}\frac{\sin\left(\frac{N-2B}{N}\pi v\right)}{\sin\left(\frac{\pi v}{N}\right)}. \end{aligned} \tag{3.9}$$

This equation may seem unexpected considering it is the Fourier transform of a rectangle function which normally is a product of sinc functions; however, this is a *discrete* Fourier transform which gives the Dirichlet or "periodic sinc" function [55]. This analysis took the image data from just after constraining the Fourier modulus which contained noise to

just before applying the modulus constraint again. The goal here is to determine how to best apply the modulus constraint such that some of the contribution of Δ is removed.

Since equation (3.9) is difficult to analyze, consider its maximum magnitude

$$G(0,0) = \Delta \frac{M-2A}{M} \frac{N-2B}{N}. \tag{3.10}$$

This shows that regardless of the location of the noise in the (u,v) plane its largest contribution after one iteration is at the origin. The effect of the noise at (a,b) influences all pixels in the (u,v) plane; however, the overall influence is less than before the iteration. This is evident from the Frobenius norm of equation (3.9) which is

$$\begin{aligned} e^2 &= \sum_{u=0}^{M-1}\sum_{v=0}^{N-1} \left|G(u,v)\right|^2 \\ &= \sum_{u=0}^{M-1}\sum_{v=0}^{N-1} \left| \frac{\Delta}{MN} \exp\left[i\pi \left(u\frac{M-2A-1}{M} + v\frac{N-2B-1}{N} \right) \right] \right. \\ &\qquad \left. \times \frac{\sin\left(\frac{M-2A}{M}\pi u\right)}{\sin\left(\frac{\pi u}{M}\right)} \frac{\sin\left(\frac{N-2B}{N}\pi v\right)}{\sin\left(\frac{\pi v}{N}\right)} \right|^2 \\ &= \Delta^2 \frac{M-2A}{M} \frac{N-2B}{N}, \end{aligned} \tag{3.11}$$

whereas the Frobenius norm before the iteration was $e^2 = \Delta^2$. The amount of oversampling thus determines the amount of filtering.

To demonstrate this graphically these equations are plotted for a simple test case consisting of a single non-zero pixel in the (u,v) plane. Fig. 29a shows the initial Fourier modulus with a single pixel containing noise of magnitude $\Delta(-15) = 1$. Fig. 29b shows

the inverse Fourier transform which is the unconstrained image. The image has a uniform magnitude of 0.02 and a periodic phase factor. The constrained image in Fig. 29c has support with $A = 10$ and real, positive pixel values. The Fourier transform of the constrained image, Fig. 29d, shows the new form of the noise. The greatest magnitude is 0.6 which equals $\frac{M - 2A}{M} = \frac{50 - 2 \cdot 10}{50}$. The square root of the sum of all of the modulus values, the Frobenius norm, is 0.7746 which equals $\sqrt{\frac{M - 2A}{M}}$.

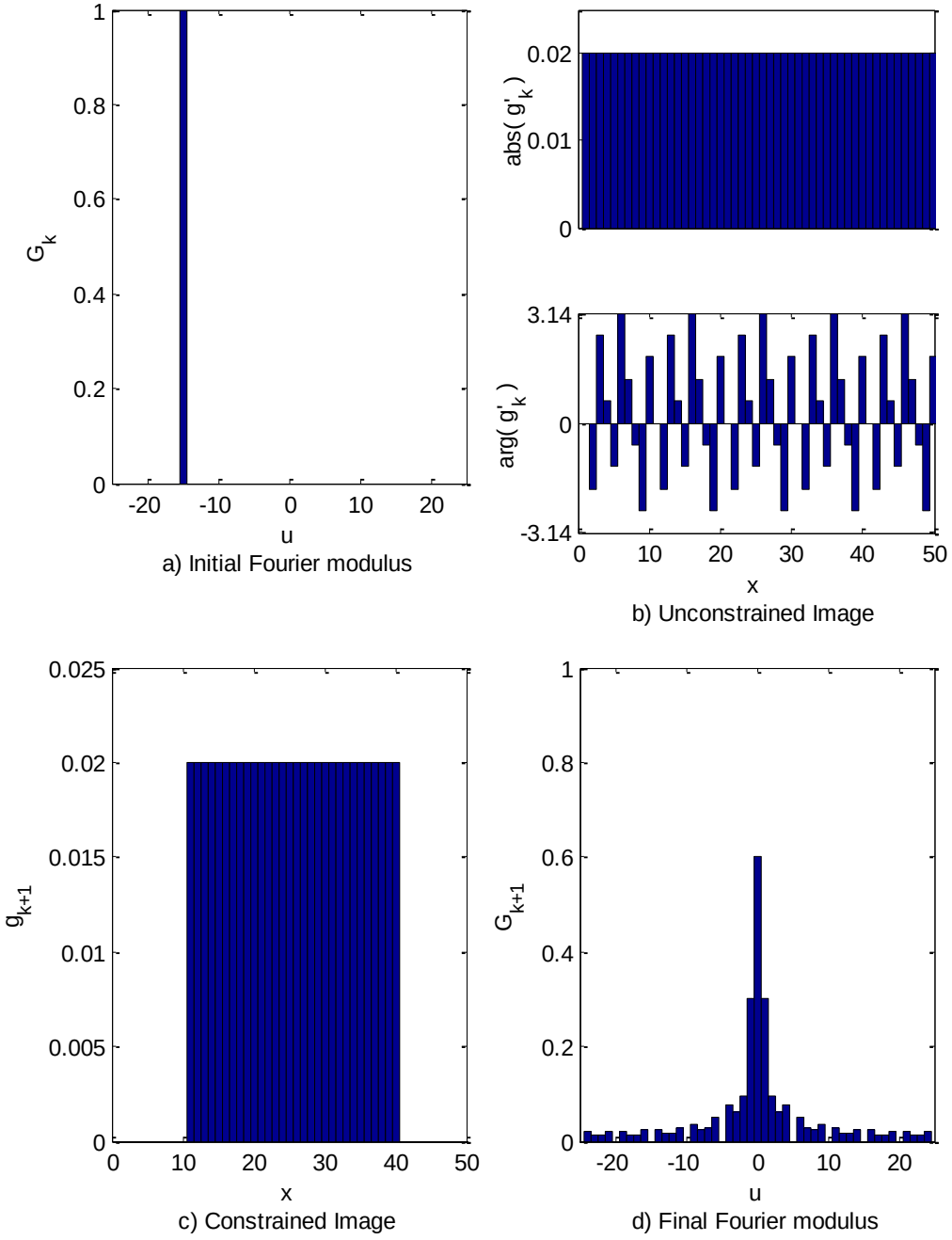


Fig. 29. Example of the Error Reduction algorithm's filtering effect through one iteration of the error-reduction algorithm.

Based on this insight, the conclusion can be drawn that the Fourier modulus after an iteration, $\left|G_{k+1}(u,v)\right|$ is a better estimate of the true Fourier modulus $F_{True}(u,v)$ than the measured noisy modulus data $F(u,v)$. This implies that the measured modulus data should be mixed with the current iteration's modulus to obtain the best estimate of the true modulus. This leads to the idea of imposing the Fourier modulus constraint in the form

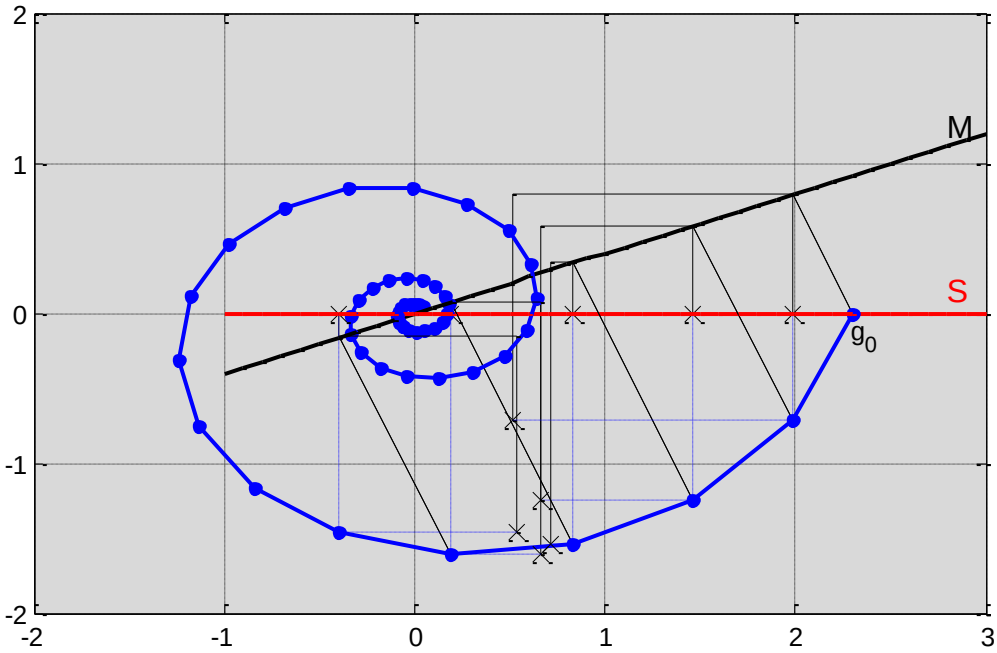
$$G_k'(u,v) = \lambda G_k(u,v) + (1-\lambda)\left|F(u,v)\right|\exp\left(i\arg\left(G_k(u,v)\right)\right) \tag{3.12}$$

which is referred to as the Constraint Relaxation (CR) method [1]. Compared to the Levi-Stark method, equation (3.2), the modulus is constrained to stay near $F(u,v)$ and is thus less likely to diverge. Compared to Liu's method, equation (3.3), the filtering effect is more stable because the relaxation parameter is not changing at every iteration. The modulus constraint in (3.12) can be written as a projection

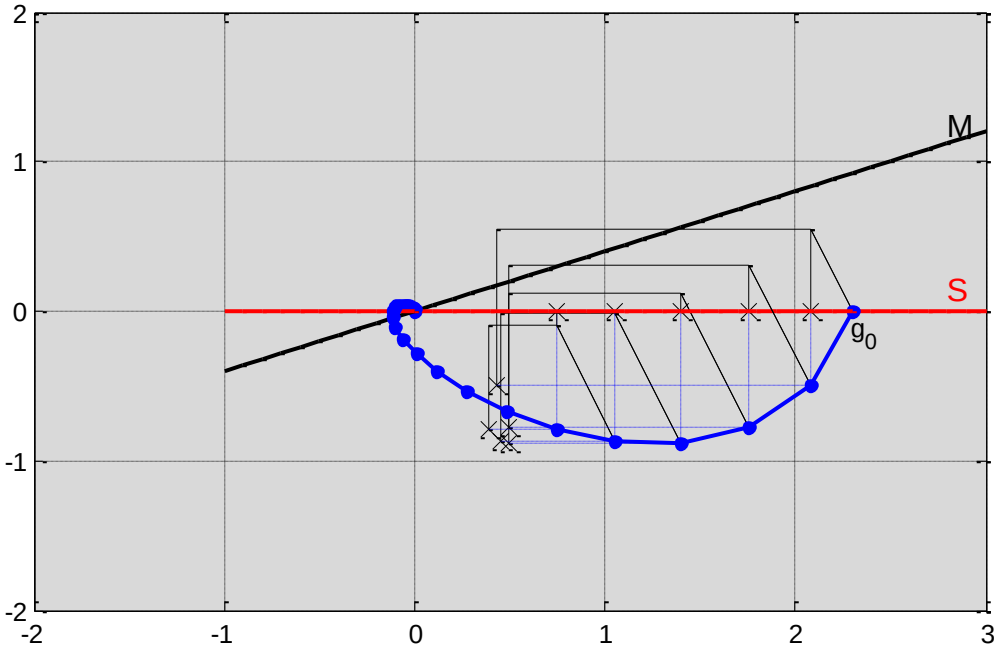
$$\begin{aligned} P_m g(x,y) = FT^{-1}\Big[\lambda FT\left[g(x,y)\right] + \\ (1-\lambda)\left|F(u,v)\right|\exp\left(i\arg\left(FT\left[g(x,y)\right]\right)\right)\Big] \end{aligned} \tag{3.13}$$

and combined with any of the phase retrieval methods discussed in section 2 to get the benefits of escaping local minima and filtering the Fourier modulus of noise. As an example the trajectory of the HIO seeking the intersection of the linear support and Fourier domains is shown in Fig. 30. The first figure shows the normal HIO where each iteration projects to the Fourier domain whereas the second figure shows the HIO with constraint relaxation where the projection does not go completely to the Fourier domain. This is apparent in the early iterations by looking at the diagonal projection paths. The result is

less spiral tendency which means less oscillation between the constraints. Convergence also occurs more quickly. This will be explored further in the next section.



a) Algorithm trajectory for HIO without constraint relaxation.

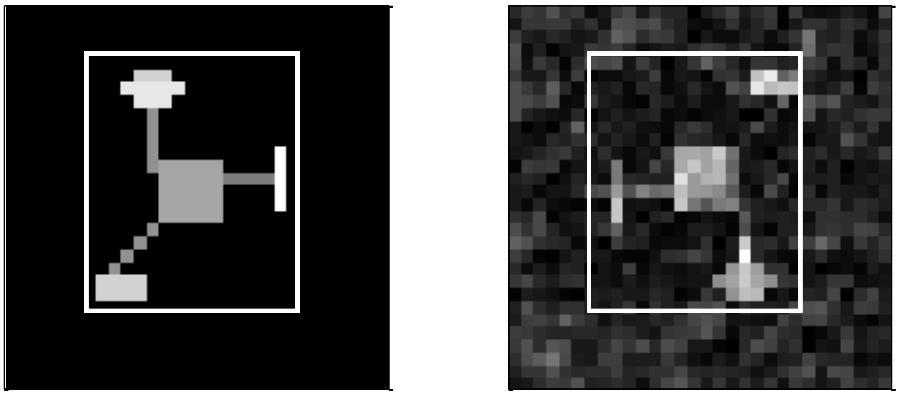


b) Algorithm trajectory for HIO with constraint relaxation.

Fig. 30. Comparison of the HIO projections with and without constraint relaxation. Intermediate projections are denoted by the thin and dashed lines.

3.2. Comparison with Existing Methods

The Hybrid Input-Output method is the typical standard for comparison since most methods of phase retrieval have their origins in the HIO. The HIO-CR method will thus be compared with the HIO first and finally to the other methods of phase retrieval developed for noise filtering. In the comparison an image of a fictitious satellite is considered which has approximately half of the pixels in the background. The foreground, the area not in γ, is assumed to be a slightly larger rectangle than the area consumed by the true foreground. The Fourier modulus is corrupted by noise with standard deviation 0.4 in the model in equation (2.17). This case has a lot of noise and typically the HIO's result is incomprehensible. The true image and the γ region used are shown in Fig. 31a. As is typical for the HIO, with the high level of noise an oscillation occurs between satisfying the modulus constraint and satisfying the image constraint. The modulus and image domain errors defined by equations (2.10) and (2.11) are shown in Fig. 32 and Fig. 33 respectively.



a) True Image b) Estimate, Iteration 500

Fig. 31. The true image of the fictitious satellite and the estimated image after 500 iterations using the HIO.

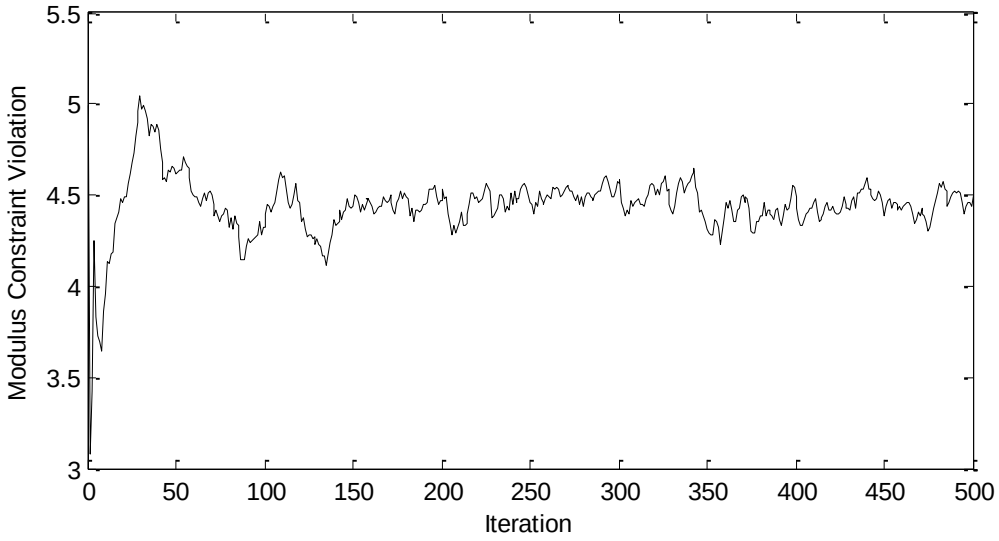


Fig. 32. Modulus constraint violations at each iteration for the HIO in the presence of noise.

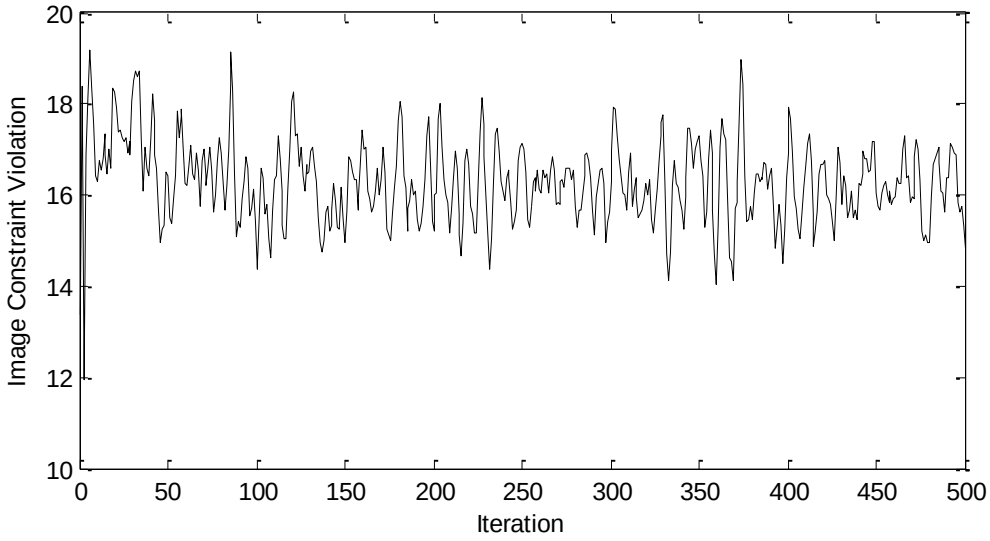
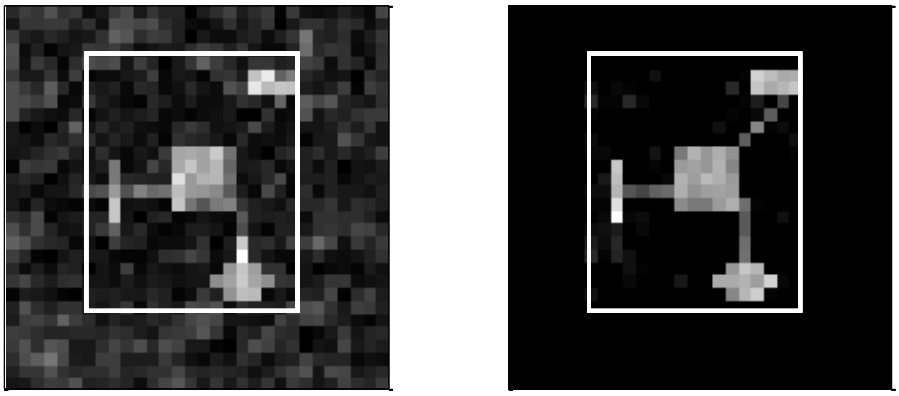


Fig. 33. Image constraint violations at each iteration for the HIO in the presence of noise.

After the 500 iterations of the standard HIO algorithm, the HIO-CR constraints were implemented for an additional 500 iterations with $\lambda = 0.9$. The image estimates before starting the CR and after the 500 iterations are shown in Fig. 34. The result has a much better resemblance to the true image. Not only is the background darker due to the image constraint, the foreground is much clearer due to filtering the noise. The reduction in the modulus constraint violations, Fig. 35, shows that approximately 75% of the noise was filtered. The image constraint violations, Fig. 36, always go to nearly zero for this method. This is because the modulus filtering both removes noise and adapts the modulus to satisfy the image constraint. The most notable effect of the CR method is its effect on the absolute Fourier modulus error defined by

$$\tilde{E}_k^2 = N^{-2} \iint \left[F_{true}(u,v) - \left| G_k'(u,v) \right| \right]^2 du dv \tag{3.14}$$

as shown in Fig. 37. This metric reveals that the initial modulus data has relative error of 2 and after the relaxation the relative error is slightly under 1. Over 50% of the noise according to this metric is filtered. A typical behavior for this error metric is a spike when the CR is initially implemented. The spike only consists of the single iteration after the relaxation starts. The spike can be eliminated by slowly increasing λ from zero to the desired value. In experience with this method so far, the spike has proven to be benign. Based on these results, the CR-HIO combination proves to be an improvement over the HIO alone.



a) Estimate, Iteration 500 b) Estimate, Iteration 1000

Fig. 34. The image of the fictitious satellite after 500 iterations using the HIO (a) and after an additional 500 iterations using the CR-HIO method (b).

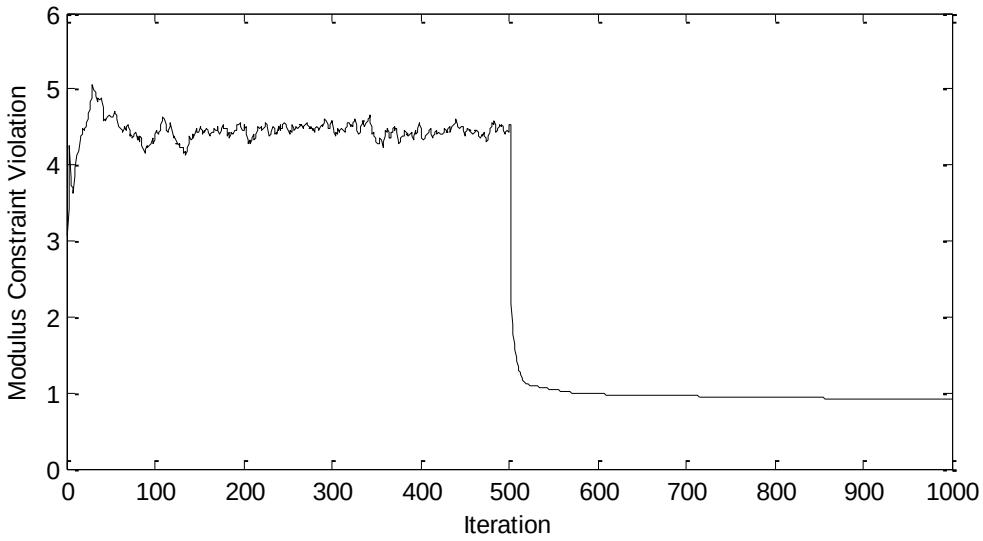


Fig. 35. The modulus constraint violations both before and after the CR is implemented.

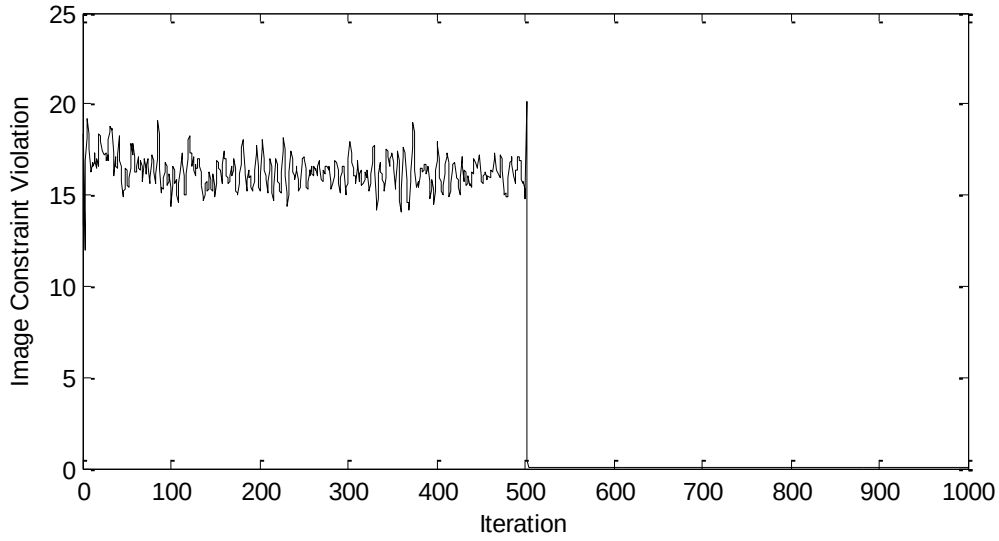


Fig. 36. The image constraint violations both before and after the CR is implemented.

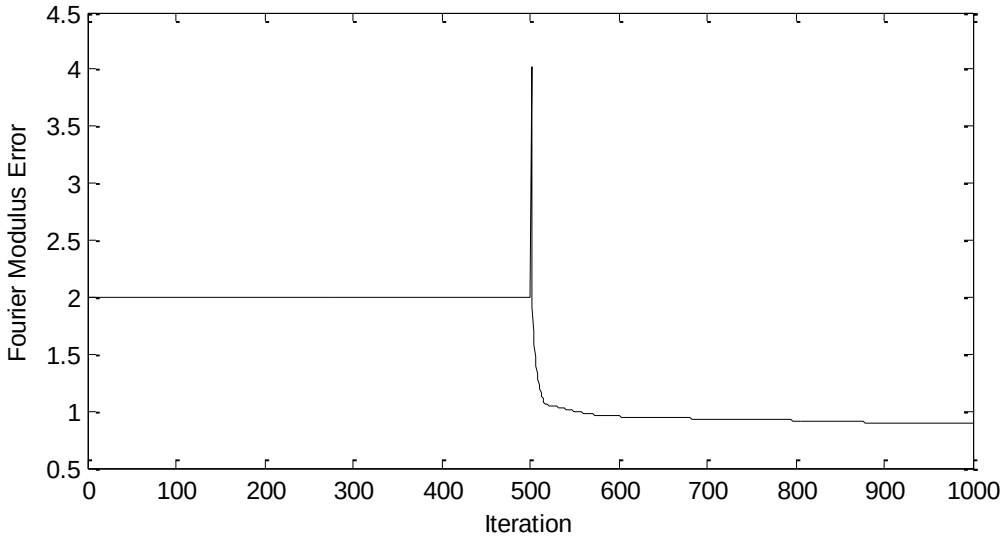
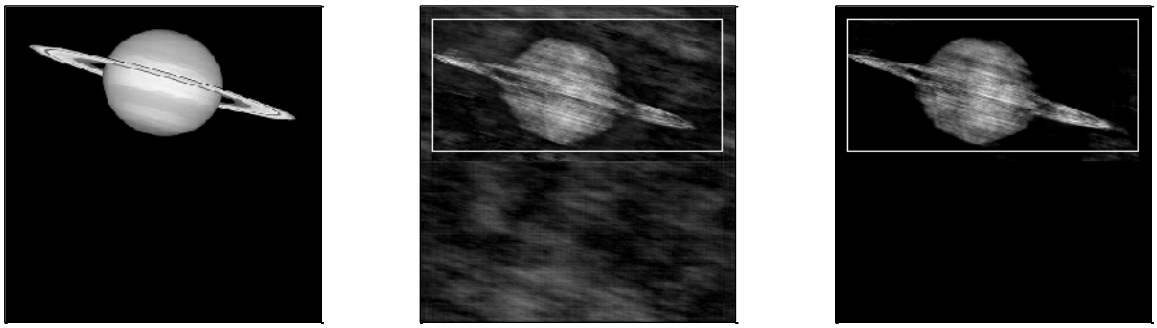


Fig. 37. The absolute Fourier modulus error both before and after the CR is implemented.

The previous example showed the image of a fictional satellite being reconstructed from its noisy Fourier modulus. To demonstrate the relaxed constraint method with a more realistic image, consider the image of Saturn in Fig. 38a. The Fourier modulus is corrupted according to equation (3.1) in the same manner as the previous example with standard deviation of 0.4. The HIO was run without the constraint relaxation for 500 iterations, and the results are shown in Fig. 38b.

The relaxation was again performed from iterations 500 to 1000 as shown in Fig. 38c. The oscillation and then reduction in the image constraint error is shown in Fig. 39. The reduction in the modulus error is shown in Fig. 40 where the total reduction is about 63%. The background region is nearly completely devoid of artifacts. The foreground is improved slightly, as shown by the planet's rings and silhouette being sharpened. The improvement of the foreground is not significant; however, the image is suffering from the convolution of the proper and flipped image. This is evident from the rings not being shown in front of the planet. Through methods such as those in [56, 57, 58, 59, 60] this can be corrected. Even with the flipped solution convolution, the Fourier modulus noise reduction was about 58%. With the addition of methods of obtaining a unique solution—which is beyond the scope of the discussions here—even more filtering is possible. It is proposed that algorithms such as those in [56, 57, 58, 59, 60], which better manage the flipping convolution and support issues, would provide far superior results if $\left|G'_{1000}(u,v)\right|$ were used as an input rather than $\left|F(u,v)\right|$, because the issue of conflicting image domain and Fourier domain constraints is nearly eliminated.



a) True Image b) Estimate, Iteration 500 c) Estimate, Iteration 1000

Fig. 38. Example result showing (a) the true image and (b) the reconstructed image after 500 iterations without constraint relaxation. The relaxation was performed from iteration 501 to 1000 with the result shown in (c). The box in (b) and (c) indicates the boundary of the background region.

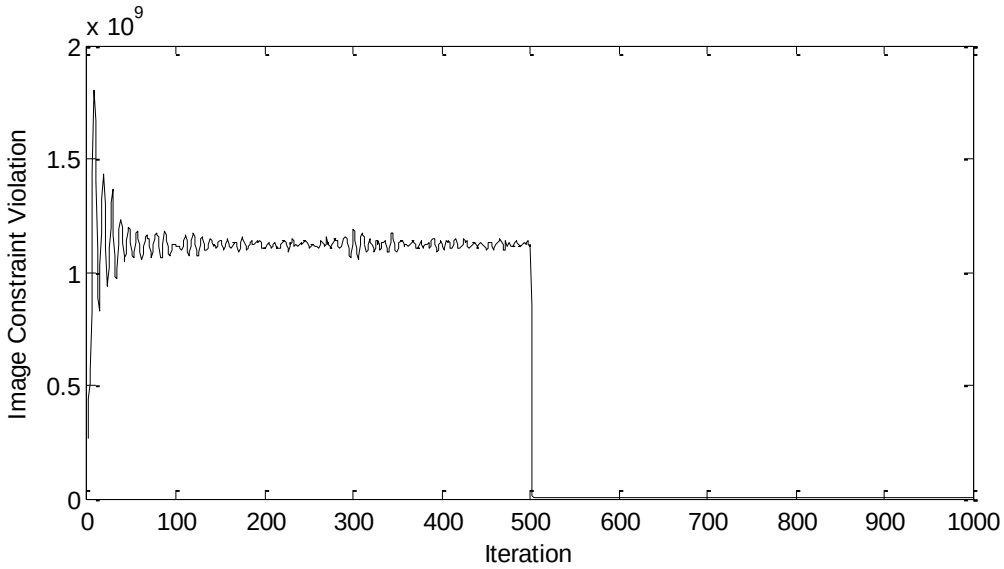


Fig. 39. The image constraint violations vs. iteration for the Saturn example.

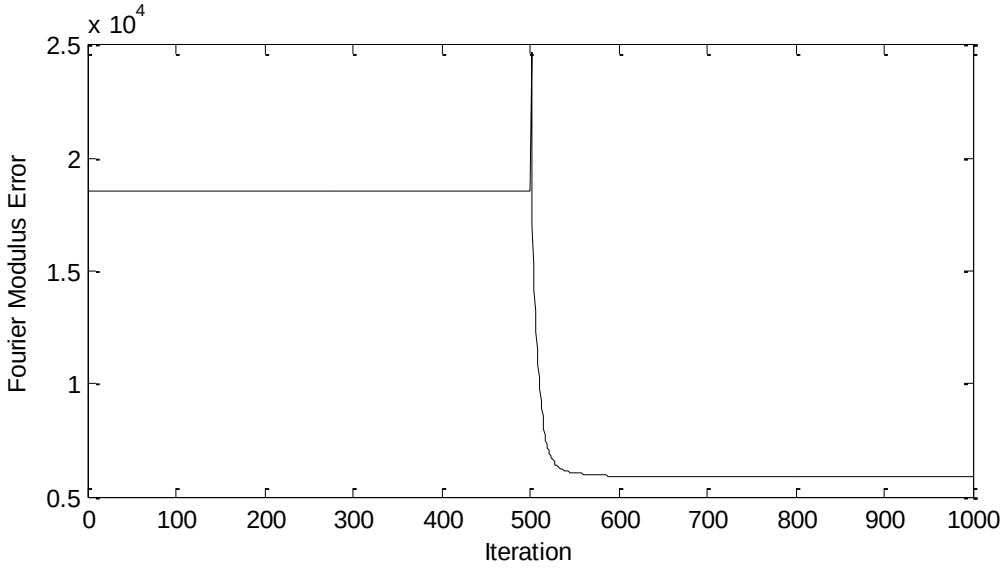


Fig. 40. The Fourier modulus error vs. iteration for the Saturn example.

To analyze the performance of the Constraint Relaxation projection in the algorithms shown in section 2, consider again the two degree-of-freedom case where the projections can be visualized. The support domain is defined as

$$S = \{(x, y) \mid x > 0, y = 0\} \tag{3.15}$$

The modulus domain is defined as a non-convex domain consisting of two semicircles. The addition of noise in the Fourier modulus data can be interpreted as a change in the modulus domain. Here for simplicity the modulus domain will be translated such that the two domains do not intersect and a gap exists [1, 43, 54]. In these examples the algorithm starts at $(x, y) = (2.5, 0)$ which is near a local minimum.

Fig. 41 through Fig. 44 show the various phase retrieval methods discussed in section 2.2. Fig. 41 has a clear intersection between the two domains. The ER, SF, and LS methods converge to the local minimum. The rest of the methods travel away from the local minimum in a direct normal to the modulus domain until the normal to the modulus domain at the intersection is crossed. After crossing the line normal to the intersection point the trajectories travel towards the intersection. In Fig. 42 the case is shown where the two domains graze each other but to not cross. In this case the RAAR converges; however, the HIO, DM, ASR, and HPR methods stagnate at an improper y value when they achieve the proper x value. Fig. 43 shows the case where the two domains do not intersect. The gap is 0.05 units at the minimum separation. Only the RAAR does not diverge or converge to the local minimum. The RAAR converges to a point near the global minimum separation. The exact stagnation distance from the global minimum is proportional to the size of the gap. Fig. 44 shows the case where the gap between the

domains is large. As shown, the gap is 0.5. In this case all of the methods diverge or converge to the local minimum.

As mentioned with the rationale for the relaxed modulus projection in equation (3.13), the relaxation parameter λ should be increased cautiously to prevent forcing the trajectory to converge to a local minimum. In the examples shown in Fig. 45, Fig. 47, and Fig. 48 the relaxation parameter is defined according to the iteration number k as

$$\lambda(k) = \begin{cases} 0, & k \le 25 \\ 0.9\,\dfrac{k-25}{25}, & 25 < k \le 50 \\ 0.9, & k > 50. \end{cases} \tag{3.16}$$

Fig. 45 shows the case where the two domains clearly intersect as also shown in Fig. 41. Comparing the region near the intersection, Fig. 46, reveals that the spiral behavior is eliminated. This means the oscillation between imposing the support constraint and the modulus constraint discussed previously has been eliminated. Fig. 47 shows the same case as shown in Fig. 42 where the two domains graze each other. Without the constraint relaxation the trajectories for several of the methods stagnate. With the relaxation implemented all of the methods either converge correctly or converge to the local minimum. With the same parameters as previously used for the case where the domains had a large separation, no method diverges as shown in Fig. 48. All of the methods except the ER, SF, and LS converge to the point on the support domain with the minimum distance from the modulus domain. If the relaxation parameter is non-zero before the trajectories move away from the local minimum, all of the trajectories converge to the local minimum as shown in Fig. 49.

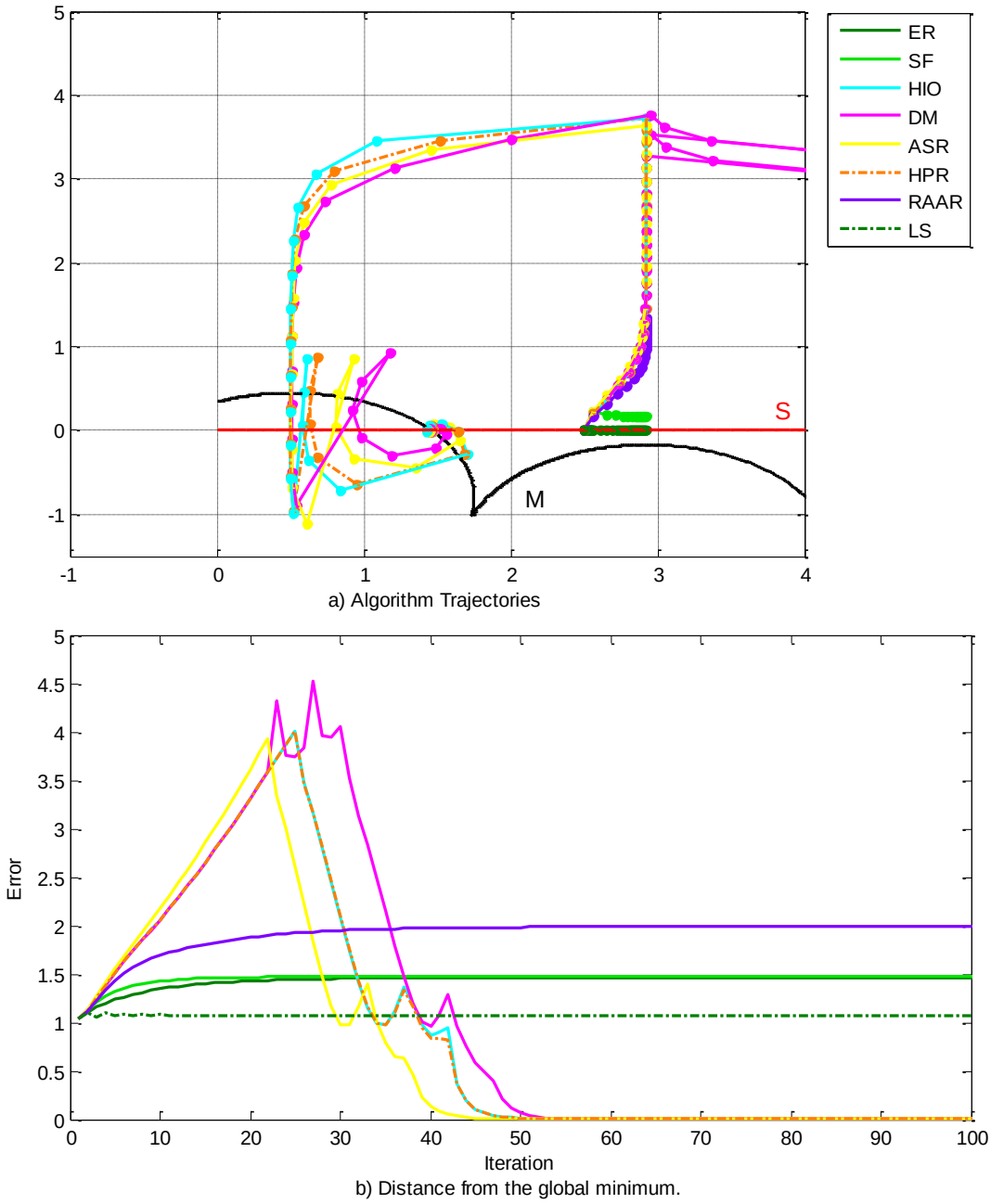


Fig. 41. Phase retrieval algorithms seeking the intersection of a non-convex domain and an intersecting linear domain.

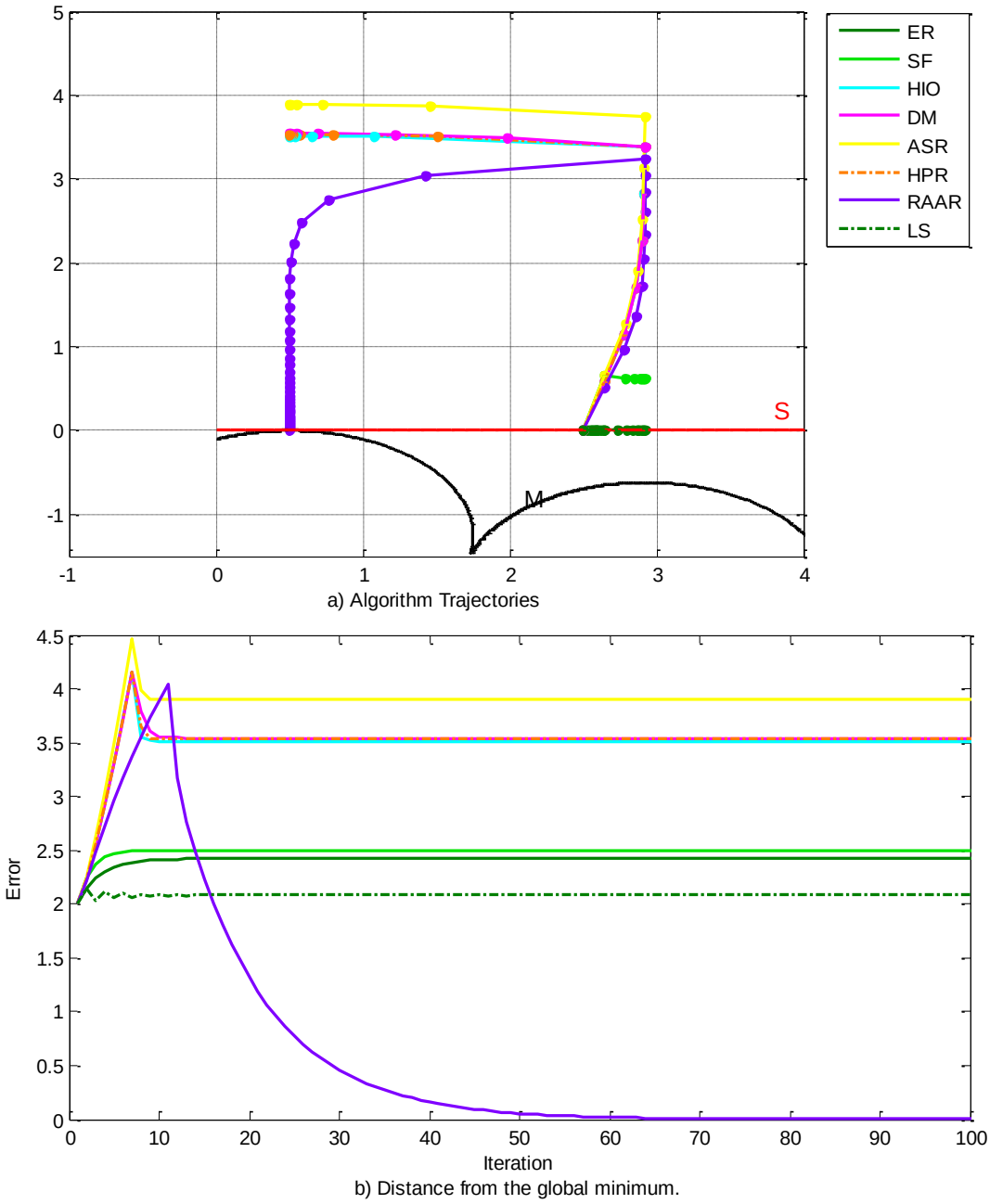


Fig. 42. Phase retrieval algorithms seeking the intersection of a non-convex domain and an intersecting linear domain. The two domains only graze each other.

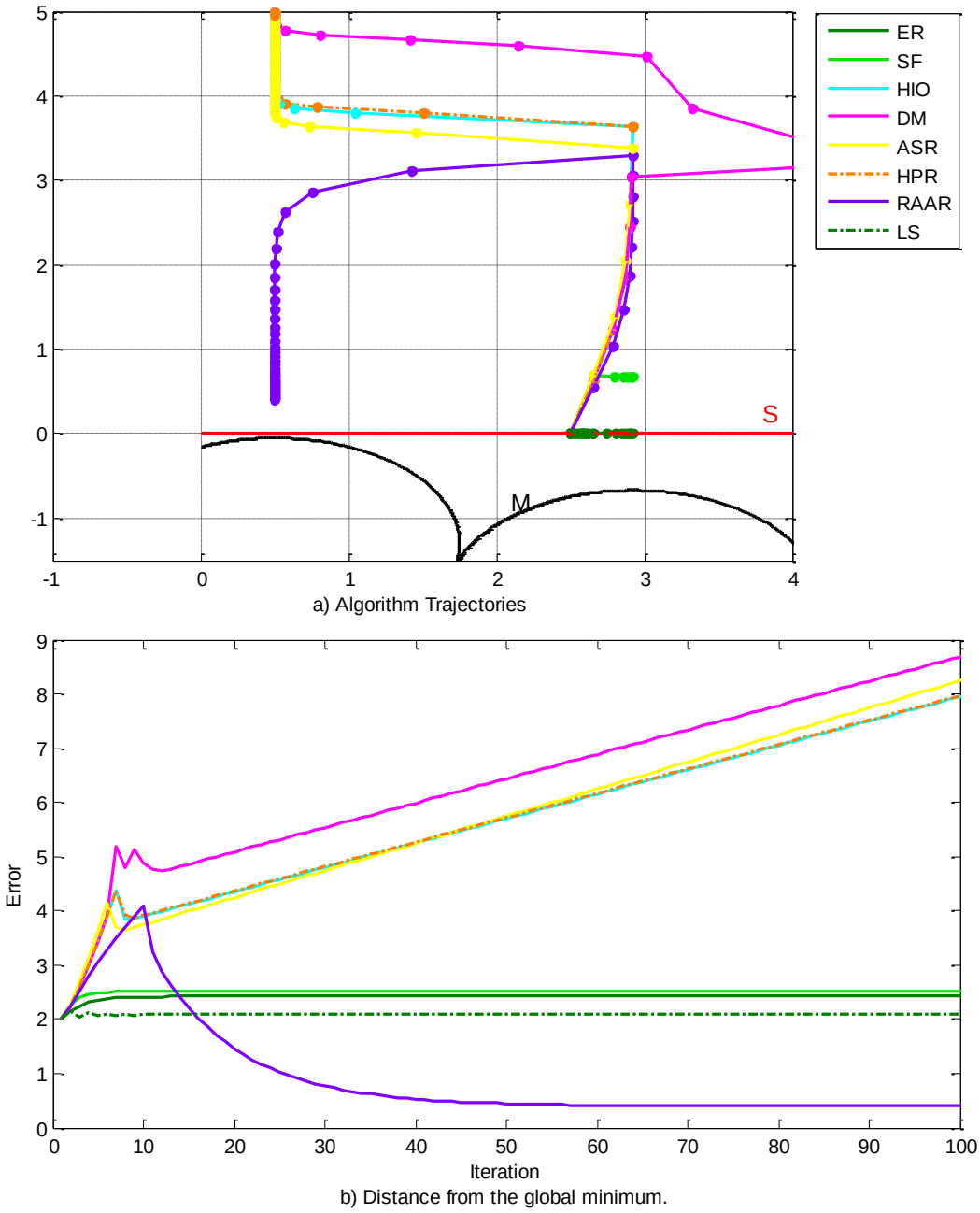


Fig. 43. Phase retrieval algorithms seeking the intersection of a non-convex domain and a non-intersecting linear domain. The minimum separation is 0.05.

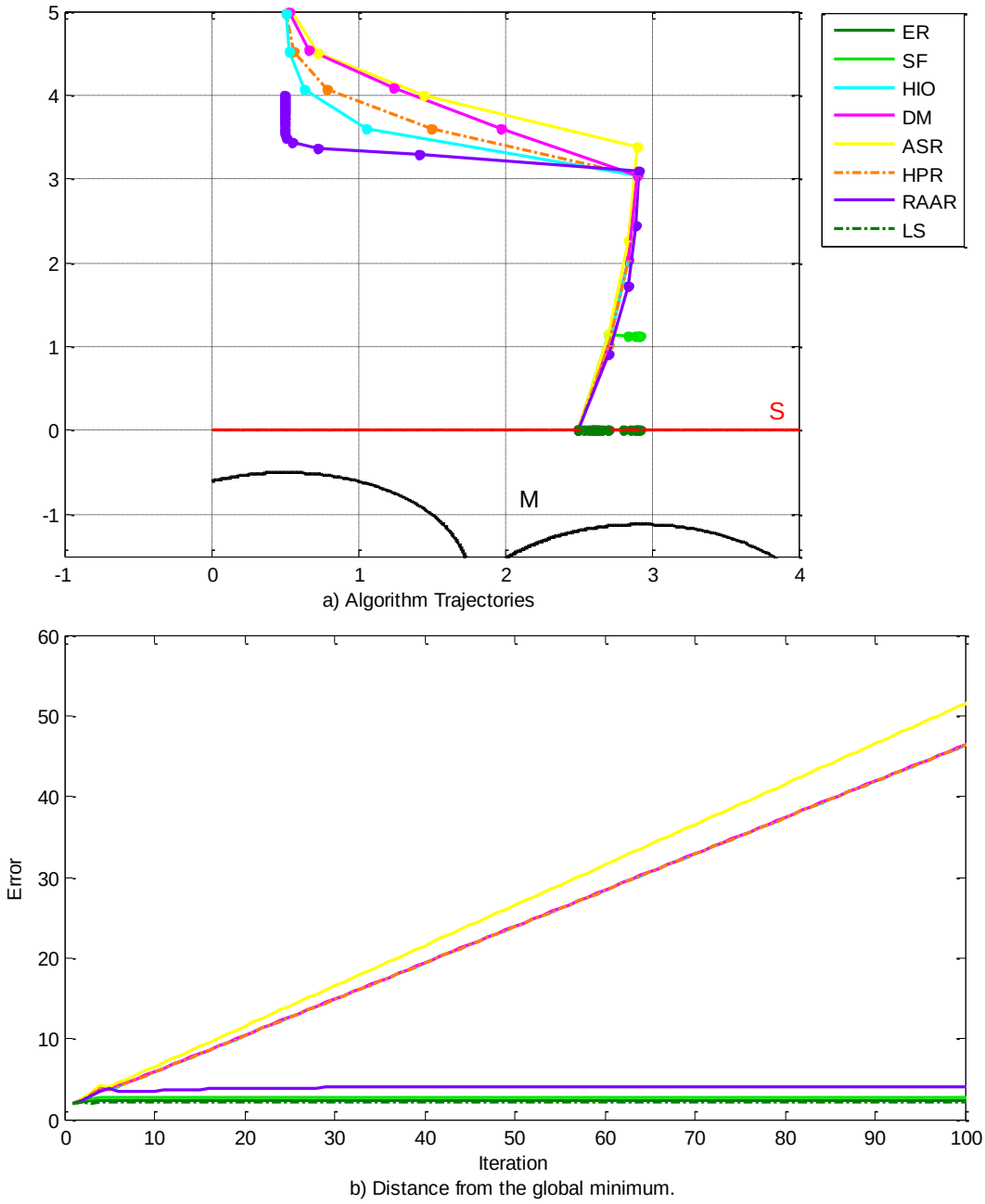


Fig. 44. Phase retrieval algorithms seeking the intersection of a non-convex domain and a non-intersecting linear domain. The minimum separation is 0.5.

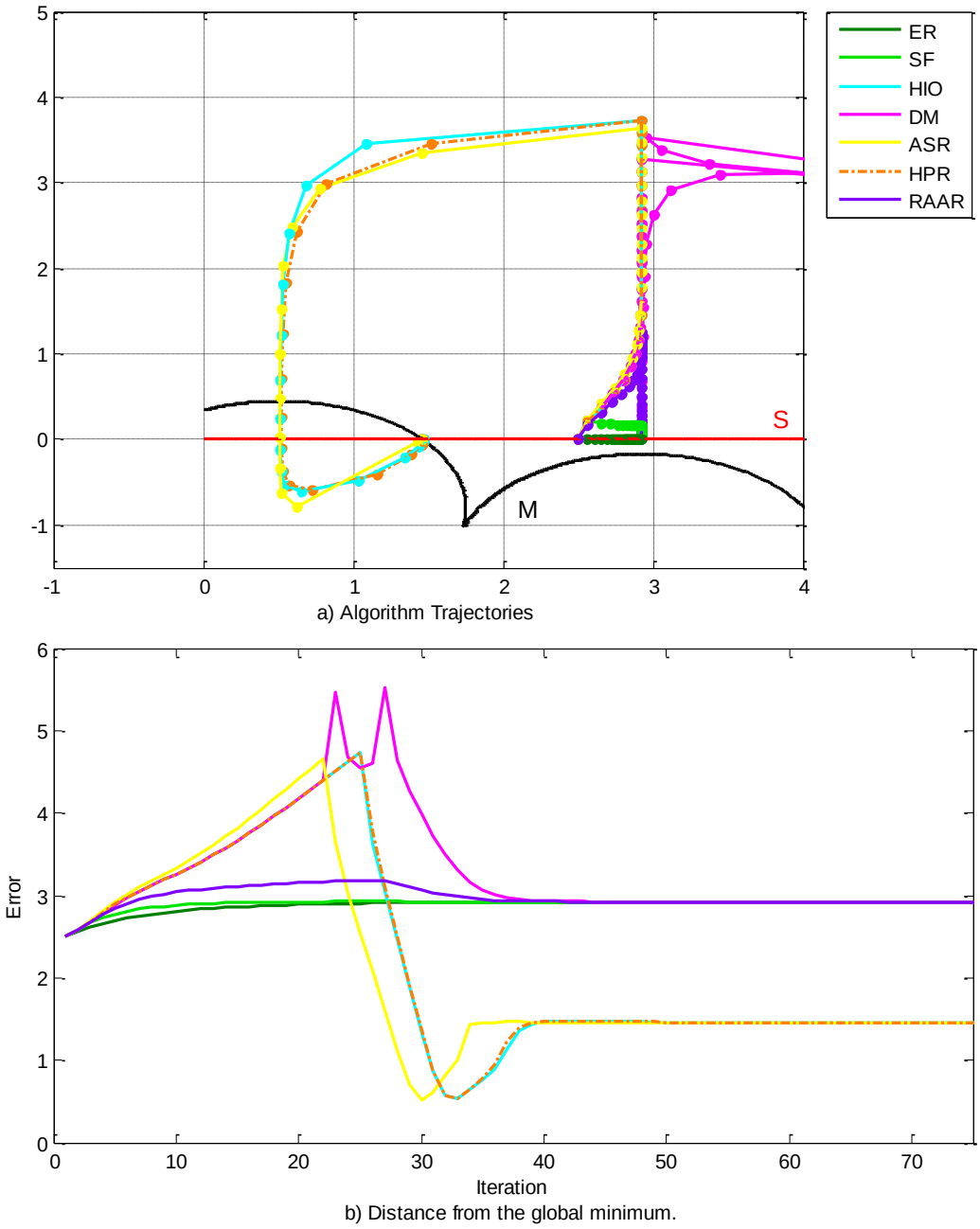


Fig. 45. Phase retrieval algorithms with the relaxed constraint seeking the intersection of a non-convex domain and an intersecting linear domain.

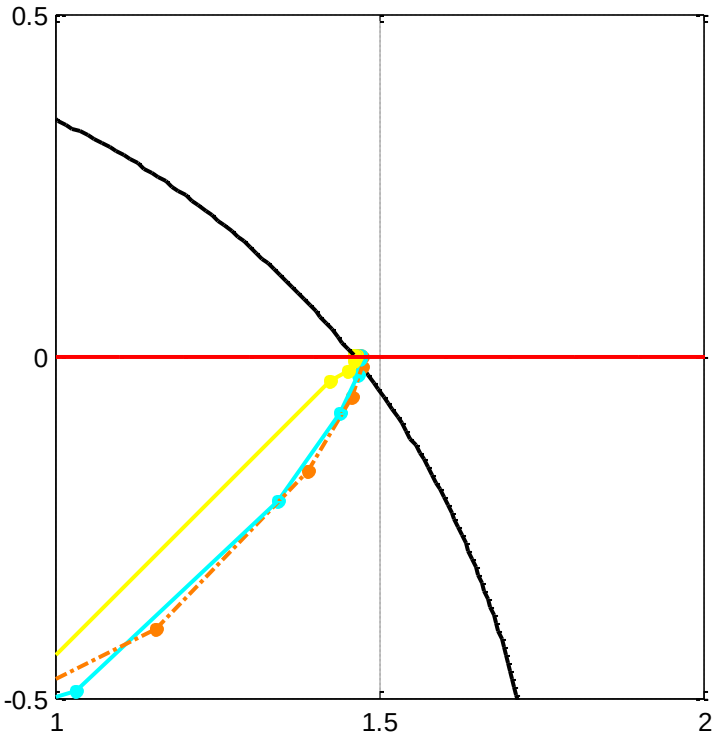


Fig. 46. Zoomed view of the intersection in Fig. 45.

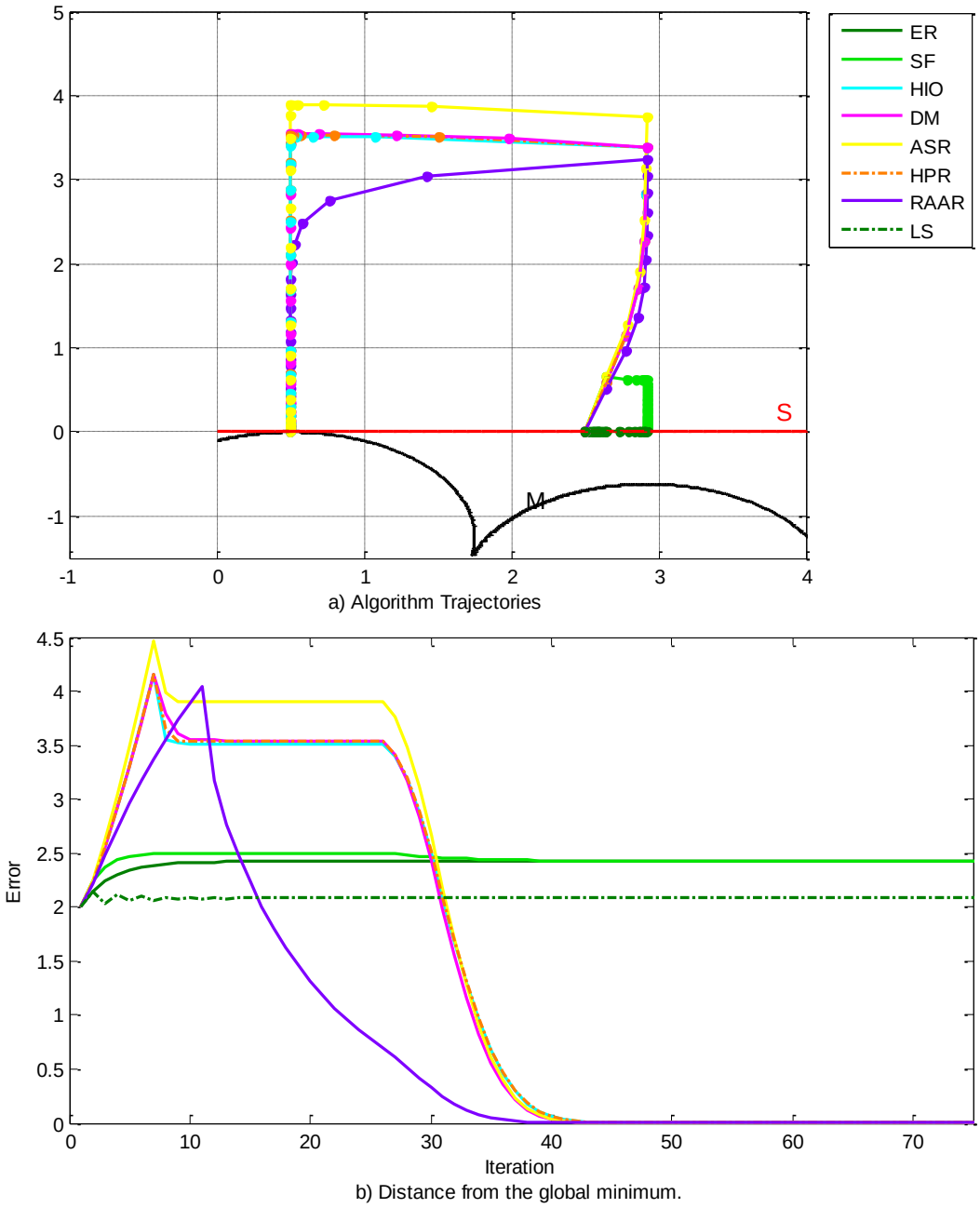


Fig. 47. Phase retrieval algorithms with the relaxed constraint seeking the intersection of a non-convex domain and a non-intersecting linear domain. The two domains only graze each other.

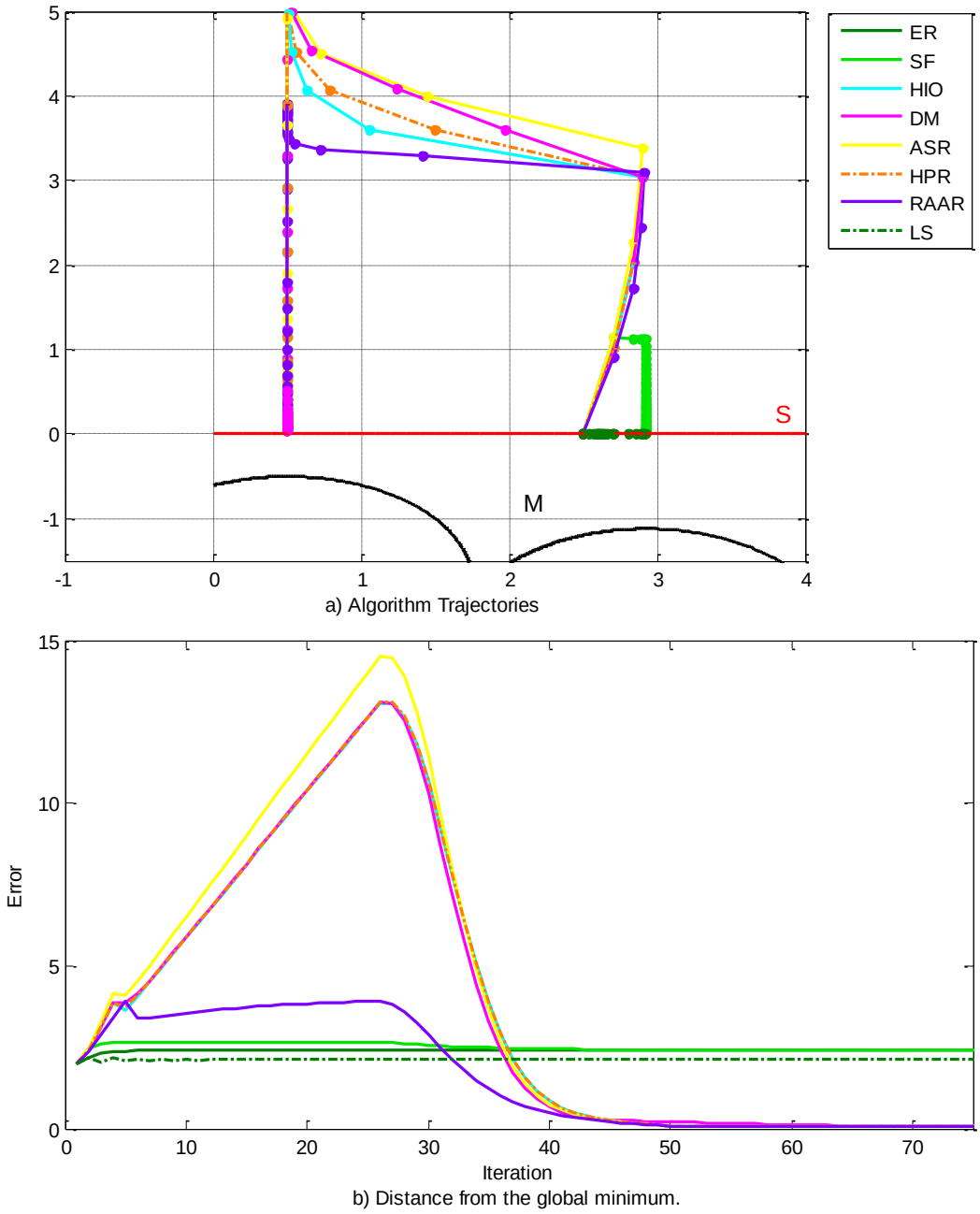


Fig. 48. Phase retrieval algorithms with the relaxed constraint seeking the intersection of a non-convex domain and a non-intersecting linear domain. The minimum separation is 0.5.

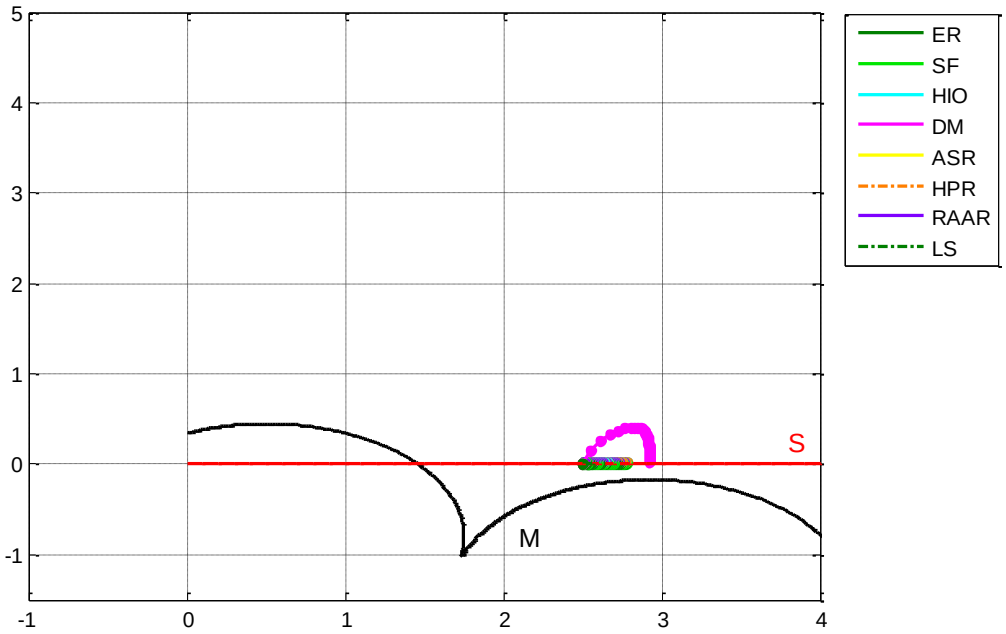


Fig. 49. Phase retrieval algorithms with the relaxed constraint seeking the intersection of a non-convex domain and a non-intersecting linear domain. The relaxation parameter was set to a non-zero value too soon.

The goal here was to devise a modification to the traditional phase retrieval methods that not only tolerates noisy modulus data but is capable of filtering the noise. Filtering the noise was set as a goal to eliminate the oscillations in the image and Fourier domain errors and help prevent stagnation. To develop a theoretical basis for the filtering scheme, the effect that noise has on an iteration of the Error Reduction method was derived. This analysis showed that noise could be filtered by carrying the effect of an iteration on the Fourier modulus over to the subsequent iteration. This result led to the development of the relaxed constraint projection operator. With the addition of the relaxation, the Frobenius norm of the error in the modulus data has been reduced by as much as 92% for noise levels high enough that the HIO alone was barely able to converge.

For comparison, the satellite example given here was run with various initial conditions and noise realizations using the method proposed in [54]. The resulting modulus error was reduced by at most 5% with an average reduction of 1%. The algorithm in [54] aides in convergence in the presence of noise but has not been shown as effective at filtering high amounts of noise in two-dimensional images. The algorithm proposed in [44] manages to filter some of the modulus noise by imposing a condition on the modulus based on the image's support. Namely, they rely on the image being largely oversampled. The algorithm presented here has no such requirement. In fact, the satellite image is under-sampled; the autocorrelation extends beyond the measured (u, v) domain. Large oversampling improved the performance here; however, it is not a requirement.

The phase retrieval methods presented in [46], [52], and [53] were shown to be similar to the algorithm presented here. These algorithms have complex projection

operators and have complex theoretical function minimization derivations and are shown to aide in eliminating local minimum induced stagnation. They, however, are not formulated to filter noise. The constraint relaxation by [52] is susceptible to instability when noise is present. In tests, it was observed to deliver comparable noise reduction to the method presented here if the algorithm did not go unstable. At high noise levels, it was rare for the algorithm to not go unstable. For the satellite example used here, the noise reduction was only 32% using [52] opposed to the 62% consistently observed using the constraint relaxation method presented here. Even when high levels of noise cancelation occurred, the filtering was still temporary. The error typically was decreased sharply when the relaxation was first implemented, and afterwards the error would steadily grow.

The relaxation method proposed in [53] performed very well with noiseless data but was never shown to exceed the noise cancelation seen by the constraint relaxation presented here. Due to the randomization of the relaxation, the noise reduction drastically varied at every iteration. The noise reduction could possibly be as high as the reduction for the algorithm presented here, but unless the iterations were stopped when the random relaxation value was optimal, the noise reduction was small or the noise was made worse. These results reinforce our claim that the algorithm presented here is intended to filter noise where others do not have this intended purpose.

Other papers such as [39], [40], and [61] discuss noise levels with respect to phase retrieval. The algorithm proposed here differs from these in that not only is the algorithm able to converge in the presence of a low SNR level—as many do—, but the noise is actually filtered.

In the examples presented here (among others we have tried), the modulus constraint relaxation provides superior results when compared to other methods of phase retrieval with noisy data. With this form of constraint relaxation combined with more complex methods of determining the image's support and handling double images, phase retrieval surpassing the current state-of-the-field is possible.

4. PHASE RETRIEVAL USING GAUSSIAN BASIS FUNCTIONS

All of the phase retrieval methods discussed previously formulated an image as a rectangular discrete grid of numbers where each number is represented by a square pixel. Within this formulation, an image with more pixels within the grid will typically reveal more detail due to the higher resolution. Several limitations of this discrete, pixelated framework can be identified [3].

A good image requires a large number of pixels; however, each iteration of the projective phase retrieval algorithms requires at least two discrete Fourier transforms. Each transform scales as $N \log N$ for a dimension of the image having N pixels. This scaling can be improved through various optimizations of the FFT method but all have unfavorable nonlinear growth. Additionally, due to the use of FFTs in the algorithms' solutions, advancement towards a closed-form solution is virtually impossible. The term closed-form is used here to refer to a solution which is exact in the sense that an additional operation or iteration is either not possible or cannot yield more accuracy. Most of the phase retrieval methods based on the ER method have parameters for the user to adjust such as feedback parameters, relaxation parameters, support bounds, etc. Convergence of these algorithms is dependent upon proper use of these parameters which add "art" to the process. In a truly blind test there is no definitive, quantitative metric to describe an output as fully converged or not. It is possible for error metrics such as equations (2.10) and (2.11) to have favorable values even though the output image is completely unlike the true image.

To overcome these difficulties with of the current methods of phase retrieval, the problem is reformulated here from the ground up. The underlying problem within all of the difficulties is the discrete representation of an image. The idea comes to mind: if a better image has a higher resolution, isn't a perfect image continuous? Work has been done in the field of super resolution but the results do not fully address the concerns here [62]. In the following discussions the phase retrieval problem is reformulated such that an arbitrary image can have a continuous representation.

In this section the formulation of a continuous image is given and compared to existing methods of forming images. Next, characteristics of an image captured using optical apertures is analyzed and used to rationalize the continuous image formulation. Finally, based on these insights a phase retrieval method is introduced which makes use of a continuous framework and has a non-iterative solution. An example is shown and compared to existing phase retrieval methods.

4.1. Pixels versus Gaussians

Since a perfect image within the traditional notion of a digital image would have infinite resolution, the perfect image should be thought of as continuous. The image pixel values can thus be a continuous function value in 2-d space. In practice the value of this continuous function is only known at discrete points in this 2-d space. If the function value is to be evaluated at any arbitrary position, the function value must be interpolated between the known points. Two of the most common methods in computer graphics for evaluating an image function value based on discrete known points are nearest-neighbor and bilinear sampling [63, 64, 65].

Nearest-neighbor sampling estimates the image function value at any arbitrary (x, y) position as

$$\hat{I}_{NN}(x, y) = I(i, j),\ i = \text{round}(x),\ j = \text{round}(y) \tag{4.1}$$

where I is the known function value at the integer coordinate $(i, j) \in \mathbb{Z}^2$. The function $\hat{I}_{NN}$ can thus be evaluated for $(x, y) \in \mathbb{R}^2$ based on limited knowledge of I [66]. An example of this sampling method is shown in Fig. 50. Nearest-neighbor sampling is the simplest of sampling methods, and its form obviously reveals the notion of visually square pixels in an image. While this form is apparently uncritically used for all current work in phase retrieval, it is far from optimal when image quality is a concern. An analytical Fourier transform of an arbitrary $\hat{I}_{NN}(x, y)$ is far too cumbersome to use. Additionally, nearest neighbor sampling is known to the computer graphics community as the worst of the sampling methods and is typically only used because of its computational simplicity [66].

Bilinear sampling performs four linear interpolations between the four nearest known function values. The estimated image function value is thus

$$\begin{aligned}\hat{I}_{BL}(x, y) = {} & I(i^-, j^-)(i^+ - x)(j^+ - y) \\ & + I(i^+, j^-)(x - i^-)(j^+ - y) \\ & + I(i^-, j^+)(i^+ - x)(y - j^-) \\ & + I(i^+, j^+)(x - i^-)(y - j^-)\end{aligned} \tag{4.2}$$

where $i^- = \text{floor}(x)$ and $i^+ = \text{ceiling}(x)$ and likewise for j and y [66]. To clarify the function $\text{floor}(x)$, it means to round x down to the nearest integer and $\text{ceiling}(x)$ rounds x up to the nearest integer. Bilinear sampling is superior to nearest-neighbor sampling in terms of the image quality to a viewer as shown in Fig. 50. The grid pattern is less evident because the function value is continuous across the grid boundaries; however, the slope is not continuous. Just as the nearest-neighbor method could be Fourier transformed, the function $\hat{I}_{BL}$ could be Fourier transformed but the result does not yield a convenient form.

A much less common image reconstruction technique is the Gaussian radial basis function (GRB) [67, 68]. This technique forms the continuous image function by superimposing two-dimensional Gaussians of various amplitudes and positions. The image function thus has the form

$$\hat{I}(x,y) = \sum_{j=1}^{N} A_j \exp\left(-\frac{1}{2\sigma_j^2}\left((x - x_j)^2 + (y - y_j)^2 \right) \right) \tag{4.3}$$

or in vector notation

$$\hat{I}(\underline{\theta}) = \sum_{j=1}^{N} A_j \exp\left(-\frac{1}{2\sigma_j^2}(\underline{\theta} - \underline{\theta}_j)^2 \right). \tag{4.4}$$

This form only requires knowledge of the position $\underline{\theta}_j = (x_j, y_j)$, amplitude A_j, and size σ_j of each Gaussian in the image. The Gaussians can be superimposed to represent an arbitrary shape as shown in Fig. 50 where six Gaussians located at $x = [0,1,2,3,4,5]$ very closely represent a sinusoid.

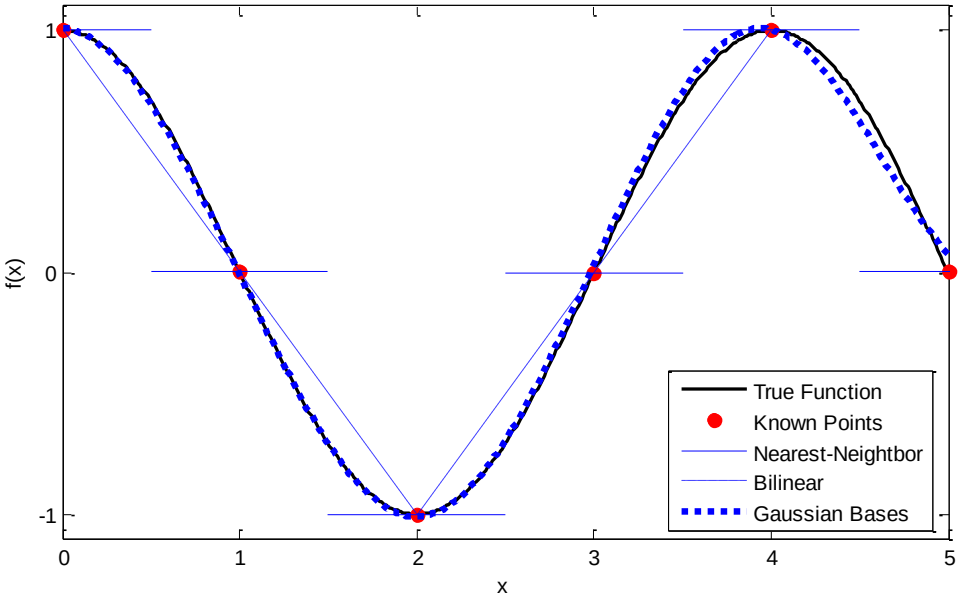


Fig. 50. Comparison of various interpolation methods used in imaging.

4.2. Gaussians in Imaging

The GRB was shown to be a viable interpolation/sampling method to convert between discrete data and a continuous function. The GRB image also lends itself well to imaging when using circular apertures such as telescopes. The point spread function for a finite circular aperture is

$$U(\omega) = C\,\frac{J_1(\omega)}{\omega} \tag{4.5}$$

where $J_1(\omega)$ is the Bessel function of the first kind and C is a constant [69, 70]. The Bessel function of the first kind is defined as

$$J_n(\omega) = \frac{1}{\pi}\int_0^{\pi} \cos\left(n\tau - x\sin(\tau)\right) d\tau. \tag{4.6}$$

The Airy pattern, which is the intensity of the point spread function, is thus

$$I(\omega) = C^2\left[\frac{J_1(\omega)}{\omega}\right]^2 = C^2\left[\operatorname{jinc}(\omega)\right]^2. \tag{4.7}$$

The jinc function is often approximated as a Gaussian because the secondary oscillations in the jinc function are small in magnitude compared to the central peak as shown in Fig. 51. By this approximation the diffraction integral leads to a Gaussian which means the optical transfer function (OTF) is also approximately a Gaussian. Since the OTF is a Gaussian, an image viewed through a circular aperture is resolved as a GRB network. The GRB is thus a better approximation of an image viewed through a circular aperture than a moderately pixelated image [3]. An example of the fictitious satellite considered

previously formed using GRBs shown in Fig. 52. This image appears blurred as if the satellite were viewed through a small circular aperture.

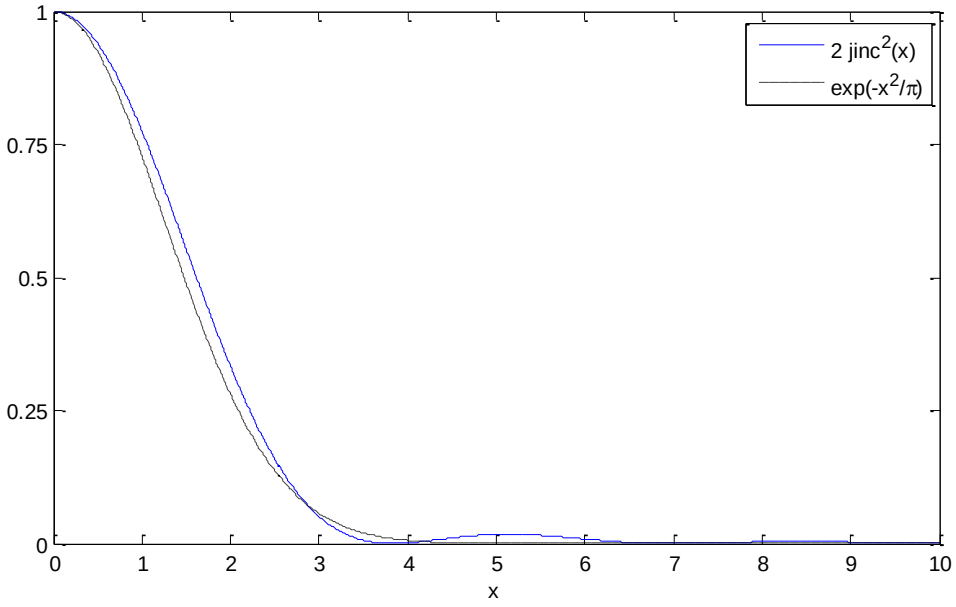


Fig. 51. Comparison of the jinc function to a Gaussian.

Fig. 52. The fictitious satellite image formed with Gaussian radial bases.

4.3. Phase Retrieval Algorithm

The primary feature of a GRB image is it can be analytically Fourier transformed easily and in a compact form for an arbitrary number of Gaussians. The end goal here is to use the known points in the diffraction pattern of the light field from a source and interpolate the data to estimate the continuous diffraction pattern. If the information in the diffraction pattern is continuous, the phase retrieval can be performed in a continuous framework rather than discrete. With this goal in mind, the transformations of the image formed with GRBs must be derived.

The image composed of N Gaussians is

$$I(\underline{\theta}) = \sum_{j=1}^{N} A_j \exp\left(-\frac{1}{2\sigma_j^2}\left(\underline{\theta} - \underline{\theta}_j\right)^2 \right) \tag{4.8}$$

where each Gaussian has the amplitude A_j, size σ_j, and location $\underline{\theta}_j$. The Fourier transform takes the form

$$J(\underline{u}) = \sqrt{2\pi} \sum_{j=1}^{N} A_j \sigma_j \exp\left(-2\pi i \underline{u} \cdot \underline{\theta}_j\right) \exp\left(-2\sigma_j^2 \pi^2 \underline{u} \cdot \underline{u}\right). \tag{4.9}$$

For clarity, the Fourier transform convention used for this derivation is

$$J(\underline{u}) = \int_{-\infty}^{\infty} I(\underline{\theta}) \exp\left(-2\pi i \underline{u} \cdot \underline{\theta}\right) d\underline{\theta}. \tag{4.10}$$

In the phase retrieval problem based on interferometry, the squared magnitude of the Fourier transform, known as the squared coherence, is measured. The squared coherence is thus

$$\left|J^2(\underline{\theta})\right| = \left| 2\pi \sum_{j=1}^{N} \sum_{k=1}^{N} A_j A_k \sigma_j \sigma_k \exp\left(-2\pi^2 \sigma_j^2 \sigma_k^2 \underline{u} \cdot \underline{u}\right) \exp\left(-2\pi i \underline{u} \cdot \left(\underline{\theta}_j - \underline{\theta}_k\right)\right) \right|. \tag{4.11}$$

This expression reveals that the relative positions of the Gaussians, $\underline{\theta}_j - \underline{\theta}_k$, appear in the squared coherence as frequency components. If these frequency components are identified, the relative position of each Gaussian can be determined. This is convenient since the frequency components within the squared coherence magnitude can be identified without making measurements over the entire (u, v) domain.

A method of identifying the frequency components in the squared coherence magnitude data is to locate maxima within the Fourier transform of the data. This Fourier transform will be referred to as the "spectrum analysis" $\hat{J}(\underline{\theta})$ to distinguish it from the Fourier domain $J(\underline{u})$. The spectrum analysis is

$$\begin{aligned} \hat{J}(\underline{\theta}) &= F\left\{\left|J^2(\underline{u})\right|\right\}(\underline{\theta}) \\ &= \sqrt{2\pi}\sum_{j=1}^{N}\sum_{k=1}^{N} A_j A_k \frac{\sigma_j \sigma_k}{\sqrt{\sigma_j^2 + \sigma_k^2}} \exp\left\{-\frac{\left(\underline{\theta} + \underline{\theta}_j - \underline{\theta}_k\right)^2}{2\left(\sigma_j^2 + \sigma_k^2\right)}\right\}. \end{aligned} \tag{4.12}$$

The spectrum contains N^2 Gaussians in the $\underline{\theta}$ space; however, the N Gaussians corresponding to $j = k$ are at the origin. There are, therefore, $N(N-1)$ Gaussians not at the origin. Looking closer, a useful property of the spectrum is revealed. Consider the value of k to be constant. The entire image is contained within $\hat{J}(\underline{\theta})$ with the $j = k$ Gaussian at the origin. The entire $\hat{J}(\underline{\theta})$ thus contains the true image N times with each copy of the image translated such that one of its Gaussians is at the origin.

Since the measured data that defines the squared coherence magnitude is typically discrete, the Fourier transform in the spectrum analysis definition can be performed as a

discrete Fourier transform. This form of the spectrum analysis would reveal maxima at the location of each Gaussian. The height of each maximum is equal to the product $A_j A_k \frac{\sigma_j \sigma_k}{\sqrt{\sigma_j^2 + \sigma_k^2}}$ and the size of the Gaussian at each maxima is equal to $\left(\sigma_j^2 + \sigma_k^2\right)$. With the identification of each maximum location $\hat{\underline{\theta}}_n$, the problem is thus to define j and k consistently such that

$$\hat{\underline{\theta}}_n = \underline{\theta}_j - \underline{\theta}_k \tag{4.13}$$

for each n. A problem can arise in identifying the maxima if two Gaussians in the spectrum overlap such that two Gaussians cannot be distinguished. In this case, rather than looking for maxima, fitting Gaussians to the spectrum with their position, size, and amplitude as variables could yield the correct result. If two Gaussians perfectly overlap, this method as documented here will fail. Therefore, based on this formulation some images cannot be reconstructed if they contain very many Gaussians. A slight modification can be added to address this issue, but it is unreliable and thus not discussed here.

Solving for the indices j and k such that equation (4.13) is satisfied for each n appears to be trivial; however, there are $N\left(N-1\right)$ possible combinations. The problem thus becomes overwhelming very quickly. For this reason the working algorithm shown here appeals to the geometric interpretation mentioned previously where the image is duplicated N times in the spectrum with each instance translated. The algorithm identifies recurring patterns and eventually identifies each of the N copies of the image.

Fig. 53 below shows the steps of the algorithm which can identify the duplicated and translated instances of the sought image within the spectrum analysis. It begins by choosing two Gaussians and designating them as part of the master image. It then finds all places within the spectrum where the increment between the two master Gaussians occur. These N occurrences are the beginnings of the N instances of the image. One Gaussian is then added to the master image at a time. If each instance of the image can find a Gaussian such that all instances match, the added Gaussian is deemed correct. If not, a different Gaussian is chosen and added to the master image. The process continues until all of the Gaussians in the spectrum have been attributed to one of the N images.

To graphically demonstrate the progression of this algorithm, consider the image shown in Fig. 54 which is simply four Gaussians. Its Fourier transform is shown in Fig. 55. Fig. 56 shows the spectrum analysis resulting from the evaluation of equation (4.12). The thickest arrows show the development of the master image. The other styled arrows denote the development of the secondary duplicate image. Note that any of the translated instances of the images shown could be considered to be the master image depending upon the algorithm's starting choice.

Once the individual Gaussians in the image are found, the image can be reconstructed using equation (4.8). The resulting image is continuous and has infinite extent. The image function can be sampled to create a grid of discrete pixel values for display.

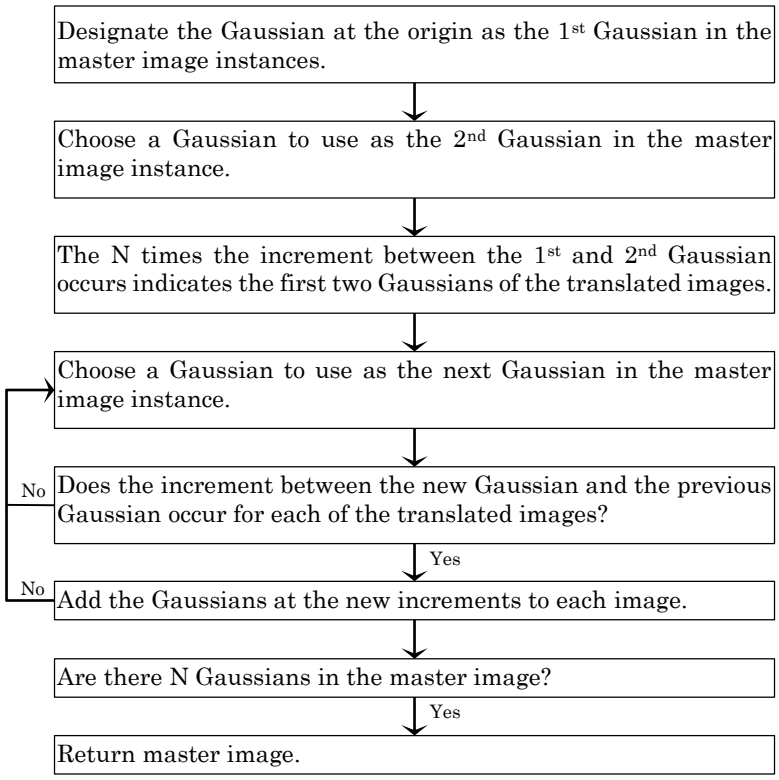


Fig. 53. Gaussian phase retrieval flow chart.

Fig. 54. Sample image formed from Gaussians used for the phase retrieval algorithm demonstration.

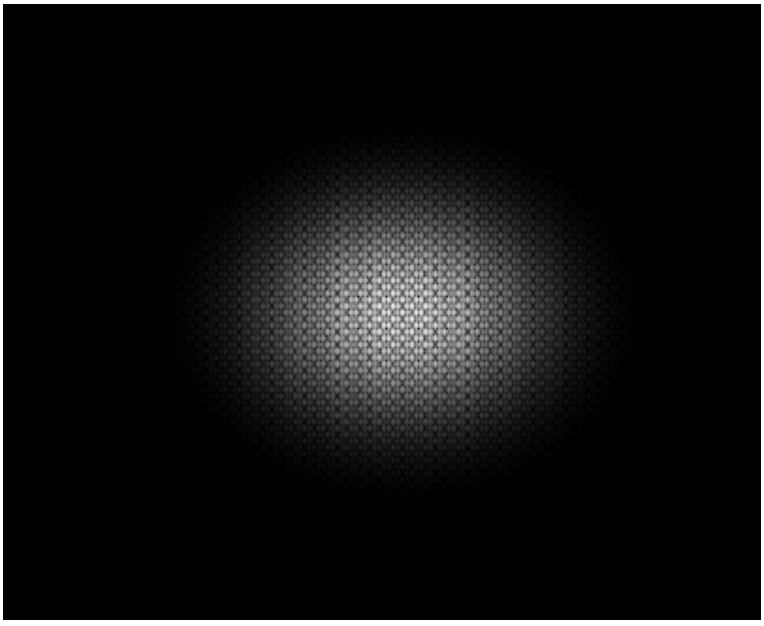


Fig. 55. The analytical Fourier transform of the image in Fig. 54

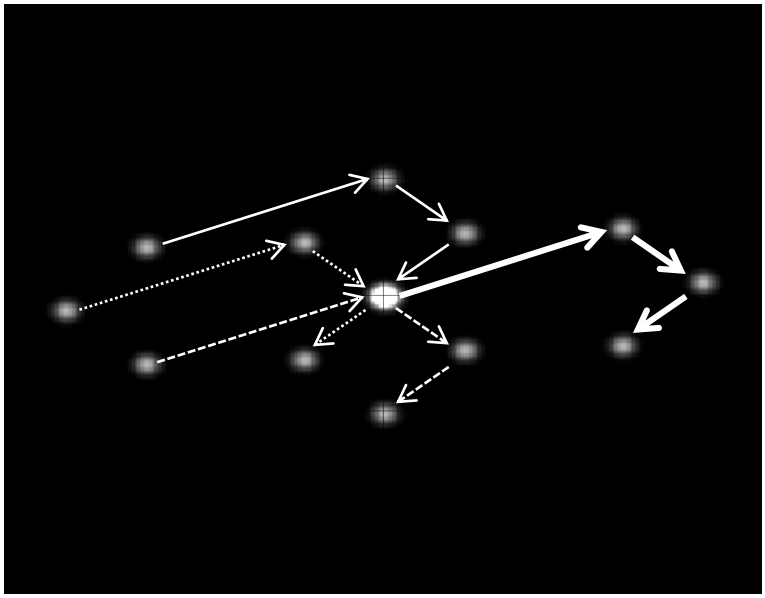


Fig. 56. Sample image spectrum analysis showing the progressive development of the four translated images. Each image has different line styles connecting the four Gaussians in the order that the algorithm identified the Gaussians.

4.4. Example

To demonstrate this technique of phase retrieval, consider the star cluster Pleiades shown in Fig. 57. For the sake of this example, the image was created such that it can be represented as a continuous function with nine Gaussians representing the nine main stars in the cluster. The measured squared coherence data in Fig. 58, sampled from the continuous Fourier transform of the image, has resolution of 1024x1024. The spectrum analysis shown in Fig. 59 is merely the discrete Fourier transform of this 1024x1024 array.

The spectrum was scanned to find all of the positions, widths, and amplitudes of the Gaussians which are then used in the phase retrieval algorithm. The resolution of the measured squared coherence data dictates the resolution of the spectrum which limits the accuracy when determining the positions of the Gaussians in the spectrum. This limited accuracy in turn limits the accuracy of the final image. The result of the Gaussian phase retrieval algorithm is shown in Fig. 60. The positions of the Gaussians in the final image are accurate to $\pm 1/2$ pixel.

It is also worth mentioning that the HIO can be used to obtain the phase estimate for this image. The HIO with feedback parameter $\beta = 0.9$ and proper rectangular background constraints requires approximately 500 iterations to converge to nearly zero error. For this image with resolution 512x512, the stars are not well defined due to the pixilation. The reproduced image in Fig. 61 looks fine; however, upon closer inspection near one of the stars there is much distortion. A comparison is shown in Fig. 62 between the topmost star in the GRB image and HIO image. In the discrete HIO image the star is not perfectly round, and there is no clear metric for describing the radius of the star. The

Gaussian bases, however, give a clear metric: the standard deviation of the continuous Gaussian. Note that the GRB allows for a discrete image to be produced at any resolution because it is a continuous image.

Fig. 57. The Pleiades star cluster used as an example of GRB phase retrieval. This image serves as both the input to the phase retrieval algorithm.

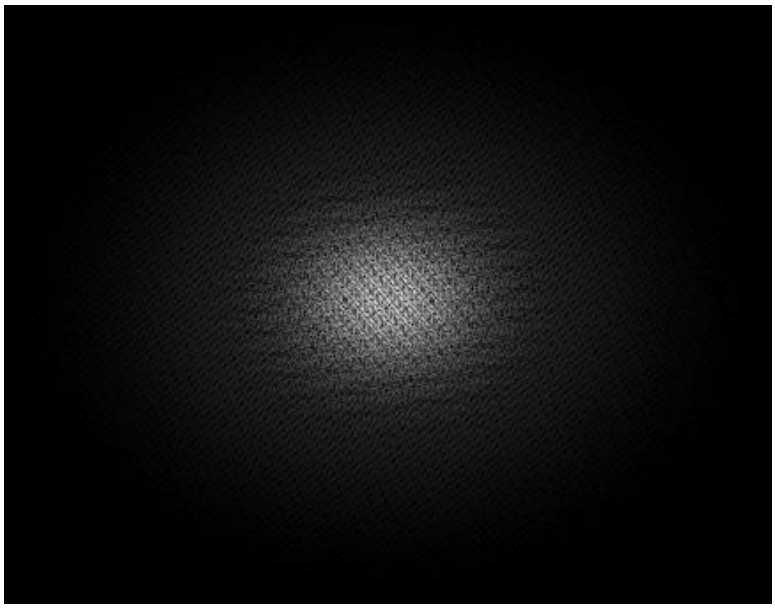


Fig. 58. The squared Fourier modulus of the Pleiades star cluster image represented as a 1024x1024 array.

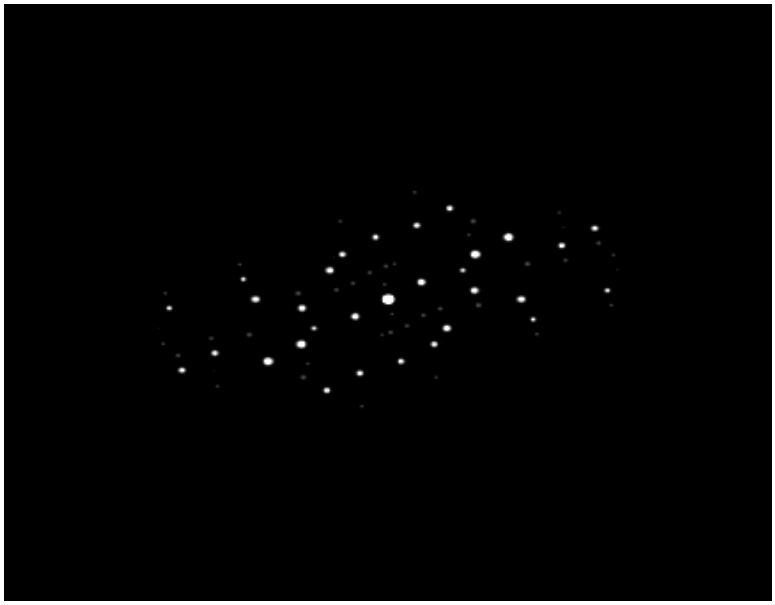


Fig. 59. The spectrum analysis of the Fourier transform of the Pleiades image.

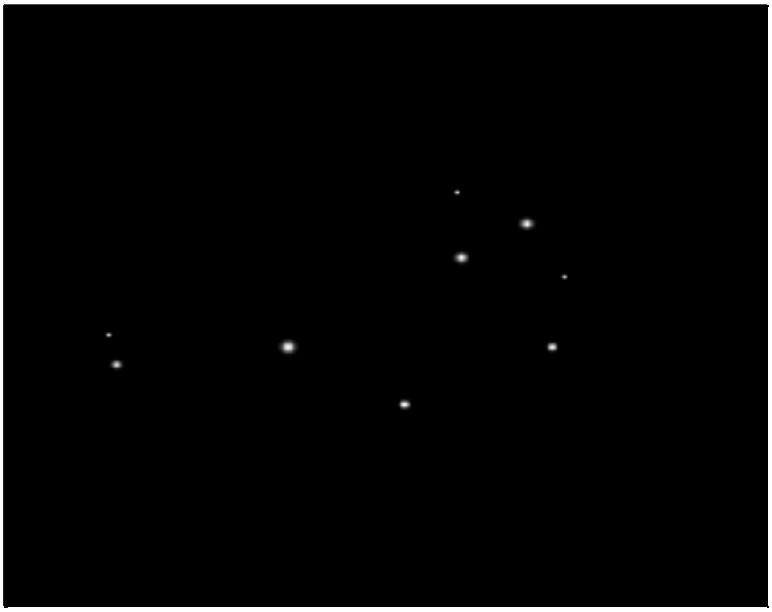


Fig. 60. The estimated image of the Pleiades resulting from the Gaussian phase retrieval algorithm applied to the spectrum shown in Fig. 59. The continuous image is point sampled for display.

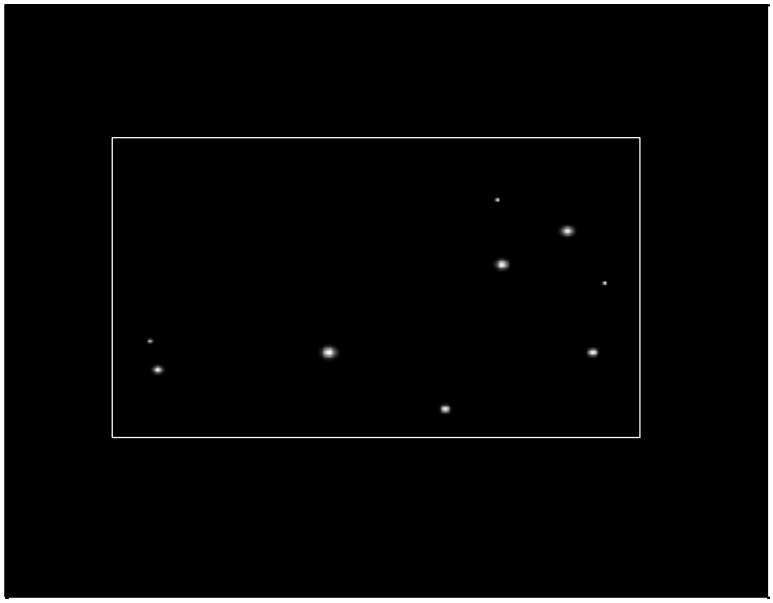
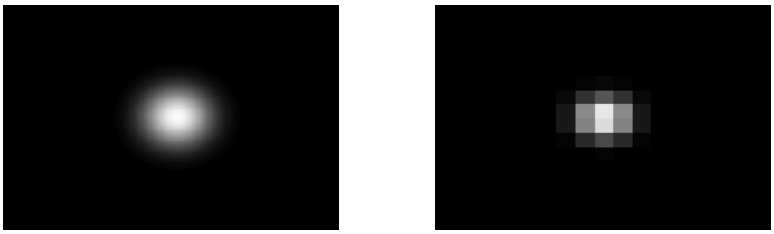


Fig. 61. 512x512 pixel result from the HIO method with a rectangular support region.



a) GRB Result b) HIO Result

Fig. 62. The topmost star in the (a) GRB and (b) HIO images as shown in Fig. 60 and Fig. 61 respectively.

4.5. Conclusion

A notable benefit of using the spectrum analysis of the squared coherence data over directly inverse Fourier transforming, as is the case in the HIO method, is that the Fourier domain must only contain enough measurements to identify its frequency components. Traditional phase retrieval methods require either complete Fourier domain coverage via measurements or some kind of estimation of the missing data. There are potentially other methods of identifying these frequency components beyond the discussion here which may lead to more accurate results and can tolerate the overlapping frequency components as discussed earlier. An additional benefit of this spectrum analysis method is the inherent continuity of the image. As shown in the HIO results, pixilation can alter the apparent content of the image. In the example shown previously, due to the pixilation a round star doesn't appear round, and the boundaries of the star are vague.

In practice the measured squared coherence data contains some amount of noise which causes a lack of smoothness in the Fourier domain. Because the Fourier transform of the coherence data is used to generate the spectrum, the smoothing effect of the Fourier transform helps to alleviate the effect of noise. Since this method of phase retrieval is in its infancy, no explicit noise analysis has been performed yet.

The method presented here builds upon the successes and shortcoming of prior research in the phase retrieval field but also takes a step back and approaches the problem from a new point of view. The properties of finite aperture optics and the continuity of an image viewed through a finite aperture lead to the Gaussian bases. The algorithm presented here employs a recursive scheme to find each Gaussian in the image based on

the pattern in the spectrum analysis. This algorithm is not iterative in the sense that there are a fixed number of recursions needed to resolve the image. This contrasts with the HIO method and other projective methods. In these other methods a subsequent iteration can always be performed, and in theory every subsequent iteration gives a better estimate of the image. Here, once the correct pattern is found via recursion, a closed form equation relates the Gaussian locations in the spectrum analysis with the bases of the image.

5. RECOVERY OF ASTEROID SILHOUETTES BY STELLAR OCCULTATION

Stellar occultation in general refers to viewing an eclipse event of a star due to an astronomical body. In particular stellar occultation typically refers to using the occultation of a star to estimate some property of the occulter such as its size. Several successful estimates have been made of the radius of moons and Kuiper belt objects which are documented in sources such as [71, 72, 73, 74, 75]. The typical method involves several observers who record the timespan in which an object occults a specific star. Each observer has a known location on the Earth's surface, thus the times of the occultation events can be used to estimate the ground track of the shadow cast by the occulter. By knowing the shape of the shadow, an estimate of the shape of the occulter can be devised. A schematic of the system is shown in Fig. 63. The major limitation of this approach is that it requires the object to cast a sharp shadow. In reality the edge of a shadow is subject to diffraction effect, thus in many cases no clear shadow exists. This is especially true for small, distant objects. Most recent research in the field of stellar occultation has focused on obtaining more precise estimates of the nominal radius of the occulter [76]. The discussions in [77] expand the circular estimate to include ellipses. The work presented here progresses beyond the estimation of simple geometric shapes. Here the general shape of an asteroid is estimated without *a priori* knowledge of the object's shape. The discussions in this section are based on the work in [4].

As shown in Fig. 64, the effect of the occulter on the light field is based upon the distance of the observer from the object, the nominal radius of the object, and the mid-

band wavelength. The Fresnel region is the region where a dark shadow zone exists. It is defined by the Fresnel number being greater than unity, where the Fresnel number is defined as

$$F = \frac{a^2}{z\lambda} \tag{5.1}$$

where a is the objects nominal radius, z is the distance of the observer from the object, and λ is the narrow-band mean wavelength of light considered [78]. A sharp shadow only exists for $F \gg 1$. Since relatively small near earth asteroids are considered here, the Fresnel number is typically less than unity which means no sharp shadow exists at the observer's location. Rather, only an interference pattern exists because the observer is within the Fraunhofer region (defined by $F < 1$). This inteference of the wave field, herein called the shadow pattern, is shown for the asteroid Itokawa at various values of z in Fig. 65b-d. It is thus suggested that characterizing an object geometrically based on the shadow is not optimal. The object should be characterized by analyzing the interference pattern it produces through an intensity mapping technique which requires a specialized method to solve the phase retrieval problem.

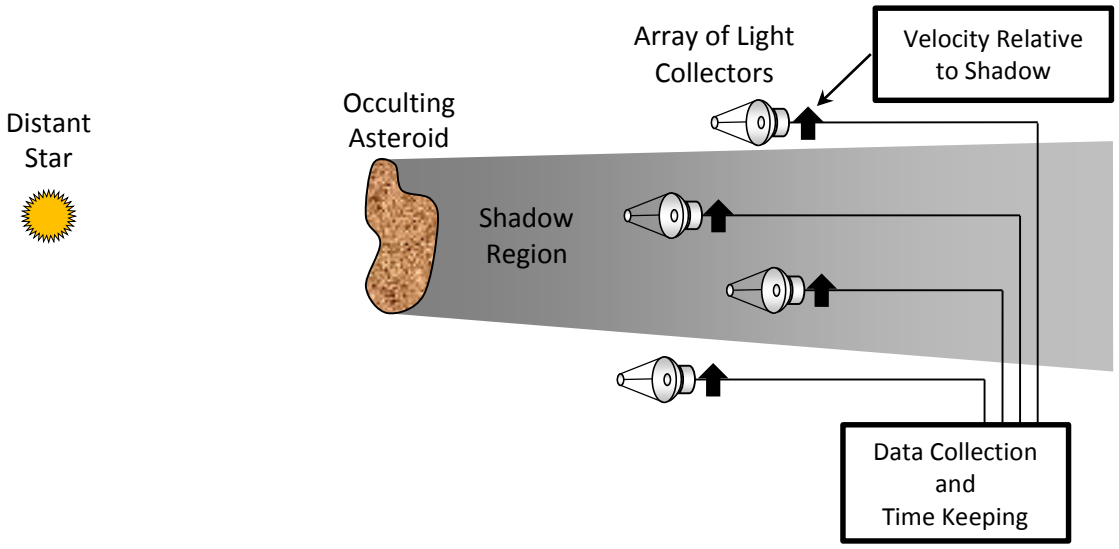


Fig. 63. Schematic of the traditional stellar occultation system which relies on a shadow with sharp edges.

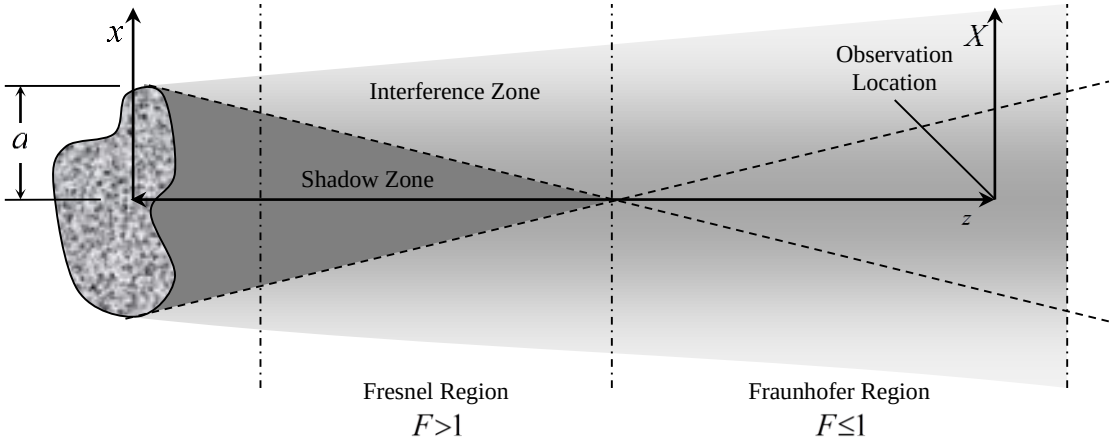


Fig. 64. Schematic of an object's shadow showing the shadow zone (darkly shaded) and interference zone (lightly shaded) and the Fresnel and Fraunhofer regions. This model assumes a point source to the left of the occulter.

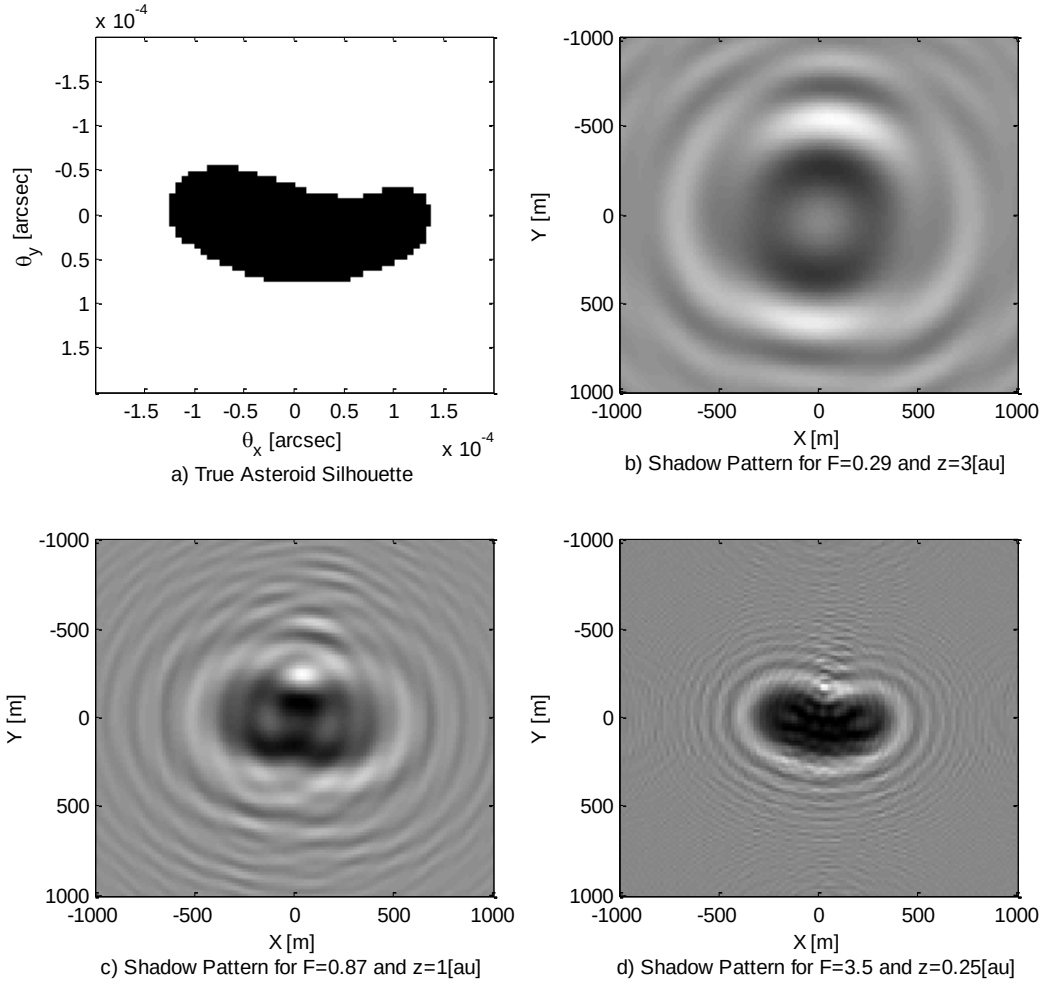


Fig. 65. Comparison of shadow patterns for the asteroid Itokawa at several Fresnel number values.

Table 3 summarizes four size classifications of asteroids and gives an estimated number of objects within the solar system in each classification. The estimated number of objects in each classification is based on the historical discovery trends. The table shows that very few of the estimated asteroids within the 40 to 140 meter range have been discovered. Based on this data, an imaging system is desired which can observe these objects. The traditional method would require the asteroid to pass close enough to have a Fresnel number of greater than 50 which would require it to pass closer to the Earth than the lunar orbit. The new method discussed here aims to function when the Fresnel number is as low as 0.5 which gives the system a range much larger than the traditional method. A new method is thus sought to characterize small asteroids which are large enough to cause significant damage upon impact before they threaten the Earth.

Table 3. Summary of asteroid size's relationship to the observation distance required for certain Fresnel numbers. Red denotes troublesome quantities and requirements. Green denotes a safe observation criterion.

Diameter (m)	**>1000**	**1000-140**	**140-40**	**40-1**
# Estimated	966	14,000	~285,000	--
# Observed	899	4,557	2,259	1,685
% Observed	93%	~33%	~1%	--
Distance (km) for F=50	>10,000,000	<10,000,000 >200,000	<200,000 >15,000	<15,000 >10
Distance (au) for F=50	>0.07	<0.07 >0.001	<0.001 >0.0001	<0.0001 >7e-8
Distance (km) for F=0.5	>1,000,000,000	<1,000,000,000 >20,000,000	<20,000,000 >1,500,000	<1,500,000 >1,000
Distance (au) for F=0.5	>7	<7 >0.1	<0.1 >0.01	<0.01 >7e-6

To formulate the problem, consider the object plane with spatial coordinates $\underline{x} = (x, y)$ and an observation plane in the Fraunhofer region with spatial coordinates $\underline{X} = (X, Y)$. Both planes are parallel and separated by the distance z, as shown in Fig. 64. The object plane can be described by the view angle from the observation location yielding $\underline{\theta} = \underline{x}/z = (\theta_x, \theta_y)$ and for convenience the observation plane coordinates can be scaled as $\underline{u} = \underline{X}/z = (u, v)$. The star far to the left in Fig. 64 is assumed to be a point source. Since the occulter is assumed to lie within the object plane, its shape defines the complex field at the image plane. The field is thus characterized by the silhouette function $\Gamma(\underline{\theta})$ which is defined as

$$\Gamma(\underline{\theta}) = \begin{cases} 0, & \text{if } \underline{\theta} \text{ is inside the asteroid profile} \\ 1, & \text{otherwise.} \end{cases} \tag{5.2}$$

Propagating the field from the point source to the object plane and then to the observation plane using the Huygens-Fresnel principle applied to shadows gives the Fresnel diffraction equation [69, 76, 78, 79, 80]

$$U(\underline{u}) = \frac{z}{\lambda} \int_{\mathbb{R}^2} \Gamma(\underline{\theta}) \exp\left(\frac{i\pi z}{\lambda} \underline{\theta} \cdot \underline{\theta} \right) \exp\left(-\frac{2\pi i}{\lambda} \underline{X} \cdot \underline{\theta} \right) d\underline{\theta}. \tag{5.3}$$

Notice that equation (5.3) contains the exponential term $\exp(i\pi z \underline{\theta} \cdot \underline{\theta}/\lambda)$ which introduces a quadratically increasing phase shift into the field. The equation thus cannot be discretized and turned into a discrete Fourier transform, as is done in the traditional phase retrieval problem, unless the discretization yields elements much smaller than the wavelengths of the term $\exp(i\pi z \underline{\theta} \cdot \underline{\theta}/\lambda)$ [76]. Since typically $z/\lambda \gg 1e10$, such a

resolution would be prohibitive computationally when considering an object plane large enough to encompass an astronomical object. Additionally, a discrete Fourier transform would not properly capture the entire image plane out to infinity as the integral in (5.3) requires. A continuous solution is thus the only apparent feasible route.

The formal problem statement here is to devise an algorithm to estimate the shape of the occulting object's silhouette function $\Gamma(\underline{\theta})$ based on knowledge of the intensity distribution (shadow pattern) $I(\underline{u}) \equiv \left|U(\underline{u})\right|^2$. The method should be able to tolerate a reasonable level of noise in the intensity data and preferably does not require spatially high resolution knowledge of the shadow pattern or *a priori* knowledge.

5.1. Data Collection

Initial designs for a stellar occultation observation system with the capability to image asteroids use a linear array of light collecting apertures spanning several kilometers. Since the asteroid and its shadow are moving relative to the apertures, each aperture has a chance to measure the wave field intensity, $I(\underline{u})$, along a line in the shadow pattern (X, Y) coordinate system [74]. A schematic of the system is shown in Fig. 66 which portrays the path of the shadow pattern moving across a linear array of observers. Preliminary designs for this system place the apertures on satellites in a "string of pearls" constellation to mitigate any atmospheric interferences.

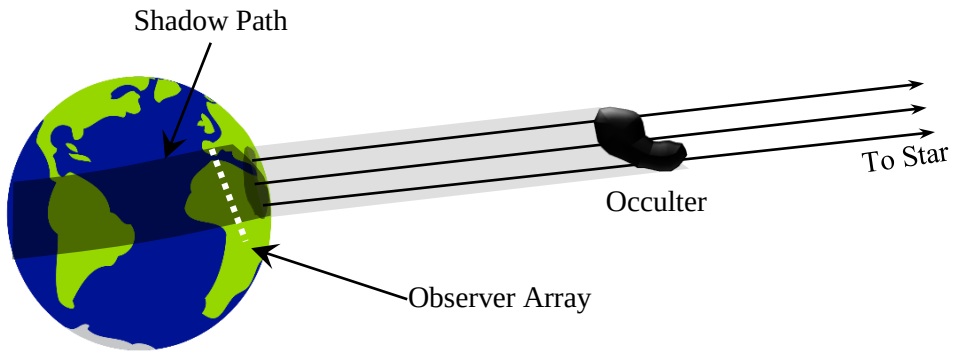


Fig. 66. Schematic of the asteroid casting a shadow which moves across observers.

5.2. Phase Retrieval Algorithm

Equation (5.3) is the standard Fresnel diffraction equation which describes the wave field some distance from the occulting object; however, its magnitude is the known quantity when measuring the changes in the wave field from the occulted star [81]. To simplify the expression it can be multiplied by the phase factor $\exp\left(i\pi z\underline{u}\cdot\underline{u}/\lambda\right)$ which has a magnitude of unity and is not a function of the variable of integration $\underline{\theta}$. This factor completes the square and gives the form

$$\begin{aligned} I\left(\underline{u}\right) &= \left|\frac{z}{\lambda}\int_{\mathbb{R}^2}\Gamma\left(\underline{\theta}\right)\exp\left(\frac{i\pi z}{\lambda}\underline{\theta}\cdot\underline{\theta}-\frac{i2\pi z}{\lambda}\underline{u}\cdot\underline{\theta}+\frac{i\pi z}{\lambda}\underline{u}\cdot\underline{u}\right)d\underline{\theta}\right|^2 \\ &= \left|\frac{z}{\lambda}\int_{\mathbb{R}^2}\Gamma\left(\underline{\theta}\right)\exp\left(\frac{i\pi z}{\lambda}\left(\underline{u}-\underline{\theta}\right)^2\right)d\underline{\theta}\right|^2. \end{aligned} \tag{5.4}$$

The phase factor is benign since only the magnitude is preserved when defining $J\left(\underline{u}\right)$. Equation (5.4) can be rewritten in the convenient form

$$\begin{aligned} I\left(\underline{u}\right) &= \left|\frac{z}{\lambda}\int_{\mathbb{R}^2}\left[1-\left(1-\Gamma\left(\underline{\theta}\right)\right)\right]\exp\left(\frac{i\pi z}{\lambda}\left(\underline{u}-\underline{\theta}\right)^2\right)d\underline{\theta}\right|^2 \\ &= \left|\frac{z}{\lambda}\int_{\mathbb{R}^2}\exp\left(\frac{i\pi z}{\lambda}\left(\underline{u}-\underline{\theta}\right)^2\right)d\underline{\theta}-\frac{z}{\lambda}\int_{\mathbb{R}^2}\left(1-\Gamma\left(\underline{\theta}\right)\right)\exp\left(\frac{i\pi z}{\lambda}\left(\underline{u}-\underline{\theta}\right)^2\right)d\underline{\theta}\right|^2 \\ &= \left|\frac{z}{\lambda}\int_{\mathbb{R}^2}\exp\left(\frac{i\pi z}{\lambda}\left(\underline{u}-\underline{\theta}\right)^2\right)d\underline{\theta}-\frac{z}{\lambda}\int_{\gamma}\exp\left(\frac{i\pi z}{\lambda}\left(\underline{u}-\underline{\theta}\right)^2\right)d\underline{\theta}\right|^2 \end{aligned} \tag{5.5}$$

where γ is the domain where $\Gamma\left(\underline{\theta}\right)=0$. The question thus becomes how to describe the region γ such that its shape can be estimated. Since it is known that the shape of the occulter may not be a standard geometric shape, an obvious choice is to describe γ using a grid of binary values. This would turn the second integral in (5.5) into a discrete

summation of continuous integrals over the elements in this grid [82]. If the (i, j) element of the grid covers the domain

$$A_{i,j} = \left\{ \left(\theta_x, \theta_y \right) \in \mathbb{R}^2 \left| 0 < \theta_x - \theta_{x,i} \le \Delta_x \wedge 0 < \theta_y - \theta_{y,j} \le \Delta_y \right. \right\} \tag{5.6}$$

and has the value $\Gamma_{i,j}$, equation (5.5) becomes

$$\begin{aligned} I\left(\underline{u}\right) = \Bigg| & \frac{z}{\lambda} \int_{\mathbb{R}^2} \exp\left(\frac{i\pi z}{\lambda} \left(\underline{u} - \underline{\theta} \right)^2 \right) d\underline{\theta} \\ & - \frac{z}{\lambda} \sum_{\forall i,j} \left(1 - \Gamma_{i,j} \right) \int_{A_{i,j}} \exp\left(\frac{i\pi z}{\lambda} \left(\underline{u} - \underline{\theta} \right)^2 \right) d\underline{\theta} \Bigg|^2 . \end{aligned} \tag{5.7}$$

Making use of the complex form of the Fresnel integral [83]

$$E\left(x\right) = \int_0^x \exp\left(\frac{i\pi t^2}{2} \right) dt, \tag{5.8}$$

the integrals in equation (5.7) expand to

$$\begin{aligned} \int_{\theta_{y,1}}^{\theta_{y,2}} \int_{\theta_{x,1}}^{\theta_{x,2}} \exp\left(\frac{i\pi z}{\lambda} \left(\underline{u} - \underline{\theta} \right)^2 \right) d\underline{\theta} = & \left[E\left(\sqrt{\frac{2z}{\lambda}} \theta_{x,2} \right) - E\left(\sqrt{\frac{2z}{\lambda}} \theta_{x,1} \right) \right] \\ & \times \left[E\left(\sqrt{\frac{2z}{\lambda}} \theta_{y,2} \right) - E\left(\sqrt{\frac{2z}{\lambda}} \theta_{y,1} \right) \right]. \end{aligned} \tag{5.9}$$

Using this expression, equation (5.7) yields [77]

$$\begin{aligned} I\left(\underline{u}\right) = \Bigg| & i - \frac{1}{2} \sum_{\forall i,j} \left(1 - \Gamma_{i,j} \right) \\ & \times \left(E\left[\sqrt{\frac{2z}{\lambda}} \left(\theta_{x,i} + \Delta x - u \right) \right] - E\left[\sqrt{\frac{2z}{\lambda}} \left(\theta_{x,i} - u \right) \right] \right) \\ & \times \left(E\left[\sqrt{\frac{2z}{\lambda}} \left(\theta_{\mathrm{y,j}} + \Delta y - v \right) \right] - E\left[\sqrt{\frac{2z}{\lambda}} \left(\theta_{\mathrm{y,j}} - v \right) \right] \right) \Bigg|^2 . \end{aligned} \tag{5.10}$$

The problem now is to estimate the (i, j) components of the grid which have $\Gamma_{i,j} = 0$. This can be done by a simple guess-and-check method where the region γ is guessed, the field's squared magnitude is computed using equation (5.10), and it is compared to the measured intensity. When a suitable estimate for γ is found the two should closely match. The only difference between the two would be the fact that the true silhouette function does not follow the grid pattern that the estimation assumed [82]. The grid should thus be dense enough to mitigate this error.

An additional convenience of this method is the relaxed requirements on the intensity data resolution. The intensity data and the silhouette grid can be different resolutions. Additionally, the intensity data can be sparse since it is only used for a comparison and not a Fourier transform as is done in the traditional phase retrieval problem. The number of intensity measurements required is discussed in section 5.5.

The simplest method to use for the guess and check is a raster scan across the grid. Each grid element $\Gamma_{i,j}$ is flipped in value and the intensity distribution error is checked for improvement. The error between the estimated and measured intensity distributions can be defined by

$$e = \sum_{\forall \underline{\hat{u}}} \left| \left| J\left(\underline{\hat{u}}_{i,j}\right) \right|^2 - \left| \hat{J}\left(\underline{\hat{u}}_{i,j}\right) \right|^2 \right| \tag{5.11}$$

where $\underline{\hat{u}}_{i,j}$ are the locations of measurements and $\hat{J}\left(\underline{\hat{u}}_{i,j}\right)$ are the measured values. If the error decreases for a change to the grid, it is kept; otherwise the value is not changed. While this appears to be a naïve approach, it often works as will be shown.

Using the error metric in Equation (5.11) has some caveats. There exist local minima in the error which can cause the algorithm to stagnate. A convenient indication is if no error reduction takes place for an entire iteration it is known that the method has stagnated. A simple method to help the algorithm along is to randomly flip a few pixel values in the image. The optimization of this algorithm to mitigate the effect of local minima is left to subsequent work. Some ideas include testing multiple pixels at one time. Within this idea the one pixel is located using the raster scan and the second pixel is chosen at random. Since such a method introduces random behaviors, only the simplest raster scan method is shown here since the performance is deterministic.

5.3. Example

To demonstrate the silhouette estimation process, consider the asteroid Itokawa as viewed from 1 astronomical unit away. The true silhouette is based on an image from the Hayabusa mission and is shown in Fig. 67. The image is 64×64 pixels and Itokawa is 40 pixels wide. Itokawa is known to be about 535m long, so each pixel in the image is 13.4m wide/tall. The angular resolution is thus $1.8 \times 10^{-5}\,\text{arcsec}$, and the field of view is $1.2 \times 10^{-3}\,\text{arcsec}$. This image is, therefore, unresolvable by even the Hubble telescope and is even not feasible for optical astronomical interferometers. If green light is considered, as is common in optical interferometry, the mean wavelength is $5.5 \times 10^{-7}\,\text{m}$. Referring to Equation (5.1) the Fresnel number is thus 0.87. Since the aphelion and perihelion of Itokawa are 1.695au and 0.953au respectively, this example represents a reasonable case of Itokawa viewed from Earth. The intensity distribution for these parameters based upon the silhouette in Fig. 67 is shown in Fig. 68. Note the bright regions within the

silhouette which correspond to a diffraction effect similar to the Arago spot which reinforces the claim that the traditional occultation method of timing the disappearance and reappearance of the occulted star is unreliable [79, 83, 84].

In practice, the entire intensity map cannot be known. If the intensity is known on a grid spanning two kilometers every twenty meters this represents a 100×100 grid as shown in Fig. 68. It is not necessary for the measurements to follow a grid pattern. This is simply done here to make the data more convenient to display in a figure. A detailed explanation of the data collection for this application is discussed in [81] and in section 5.5. It is advantageous for the measurements to span an area large enough to capture the distorted silhouette but also small enough to avoid the high frequency fluctuations that occur far away from the origin as evident in Equation (5.5). Although these fluctuations are small, proper measurements far away from the origin would require tight tolerances on spatial positioning to resolve these fluctuations. Additionally, far away from the origin the field amplitude is approximately constant according to the Fresnel integral. Information far away from the origin is thus not as useful as information near the origin.

For this example the silhouette is assumed initially to be all white. The pixels in the grid are tested by flipping their values from top to bottom, left to right in a regular pattern and looking for a decrease in the error defined in Equation (5.11). A complete scan across the grid is designated as one iteration. Fig. 69 shows the error defined by Equation (5.11) through ten iterations. The image after just one iteration is shown in Fig. 70. The general shape of the asteroid is already apparent since the most error reduction occurs in the first iteration. Although often unnecessary for convergence, *a priori* knowledge can

speed convergence. The user can manually edit the estimate before continuing to the next iteration based on intuitive knowledge that an asteroid will not typically have holes in it or any other *a priori* knowledge. This was not done for this example to prove convergence without this knowledge.

The final silhouette is shown in Fig. 71 where the input intensity matches the transformation of the estimated image, i.e. the error is nearly zero. Notice that the error drops drastically near the last iterations. This is because a single incorrect pixel accounts for the majority of the error value.

The result of the 32×32 image can be used as the initial estimate of a higher resolution estimation. This has been implemented with great success. Initially a 16×16 or similar coarse image is computed. Its estimate serves as the input to the 32×32 image recovery. This nested grid approach can reduce the number of iterations needed to recover high resolution images and helps to ensure convergence. Using this technique the exact 64×64 silhouette was estimated using the 32×32 estimate as the initial guess in only ten iterations.

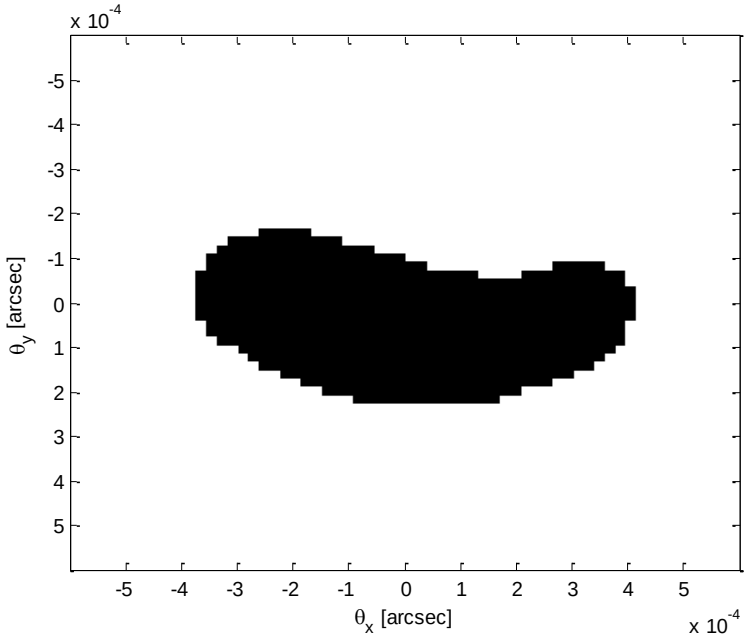


Fig. 67: The true silhouette of the asteroid Itokawa pixelated in a 64x64 grid based on images from [85] and scaled for Itokawa as viewed from 1 astronomical unit away.

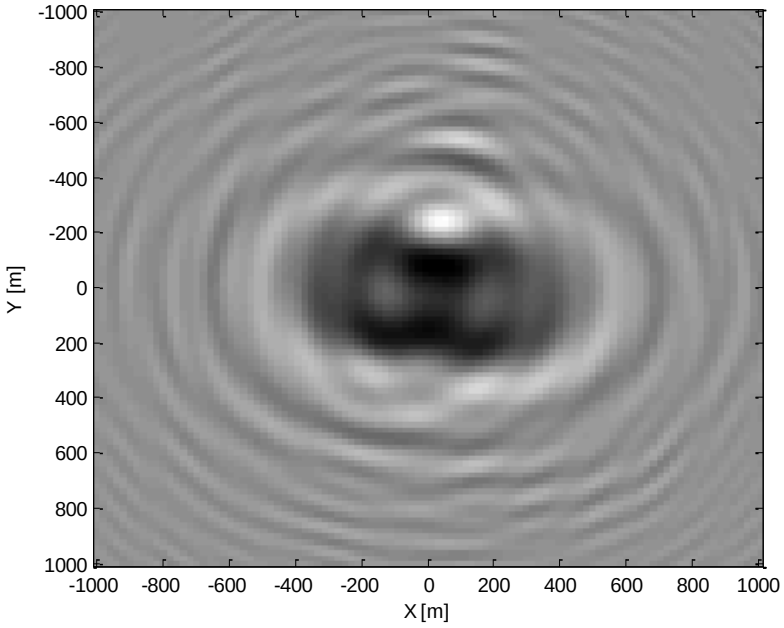


Fig. 68: The coarse grid of intensity data for the asteroid Itokawa viewed with a Fresnel number of 0.87 which serves as the input to the phase retrieval algorithm.

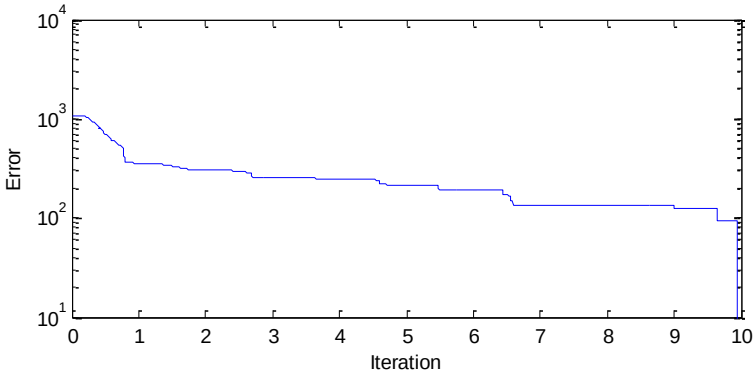
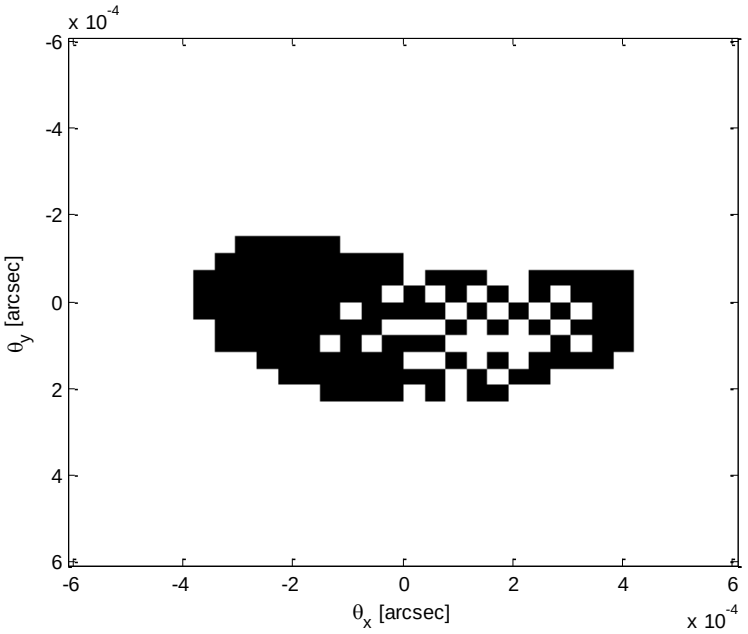


Fig. 69: The error between the intensity distribution of the estimated image and the measured intensity data.



5

Fig. 70: The estimated silhouette of Itokawa pixelated in a 32x32 grid after 1 iteration.

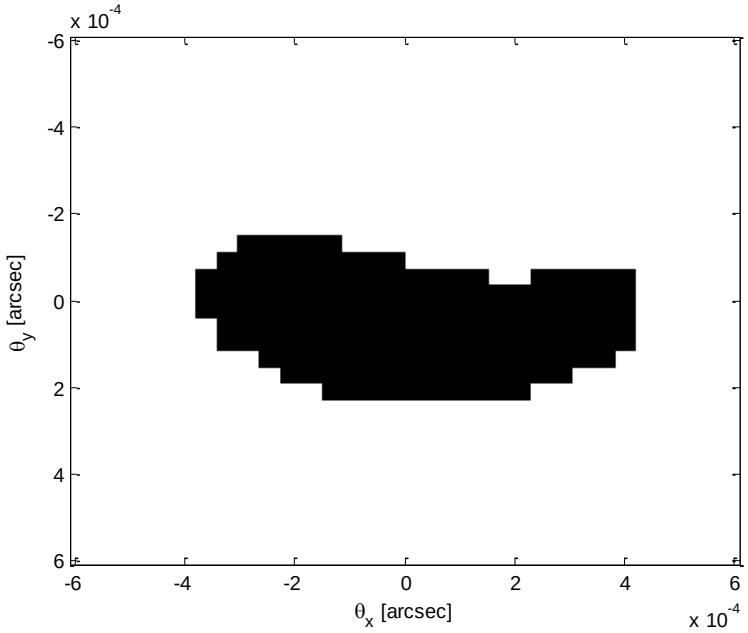


Fig. 71: The estimated silhouette of Itokawa pixelated in a 32x32 grid after 10 iterations.

5.4. Silhouette Recovery in the Presence of Noise

The intensity measurements here are independent recordings of the field intensity. Because of the quantization of light (in photons), the output of any intensity measurement device has inherent noise due to randomness of the arrival times of individual photons. Therefore, it is important that the method be capable of converging when the intensity contains random noise. The noise model used here is commonly used in Michelson interferometry and takes the form discussed previously [86, 87]. The model is

$$\hat{J}\left(\underline{\hat{u}}\right) = \left|\hat{U}\left(\underline{\hat{u}}\right)\right|^2 = \left|U\left(\underline{\hat{u}}\right)\left(1 + N_1\left(0,\sigma\right) + iN_2\left(0,\sigma\right)\right)\right|^2 \tag{12}$$

where $N_i\left(0,\sigma\right)$ is independent random Gaussian noise with a standard deviation of σ [1, 42, 88]. In an attempt to quantify the performance of this algorithm in the presence of noise, the 32×32 case for Itokawa was used in a Monte Carlo simulation. Twenty-five image estimates were constructed for different realizations of the noise at several noise levels, and no extra measures were implemented to mitigate local minima. The noise levels correspond to typical discussions of noise in phase retrieval [43, 61]. The mean errors of the twenty-five estimates at each noise level reveal an interesting trend shown in Fig. 72. As expected, the error decreases very quickly during the first iteration. For some noise levels the error later stagnates at a non-zero level while some converge to nearly zero.

The final error values are shown in Fig. 73. The trend that at near zero noise levels the algorithm typically stagnates at a higher error level than at noise levels between five and ten percent is unexpected. At near zero noise levels the standard deviations of the Monte Carlo results are much larger than at higher noise levels. The stagnated error

increases linearly above ten percent. The erratic behavior for near noiseless cases is not currently understood.

An example of the result for a noise standard deviation of 0.2 is shown in Fig. 74. The result contains some incorrect pixels but the shape of Itokawa is clearly evident. This result can be greatly improved by a simple Gaussian image filter applied to the intensity map. The result after filtering the intensity data using a Gaussian with a standard deviation of three pixels is shown in Fig. 75. Only two pixels in the final silhouette are incorrect and one is far enough away from the silhouette to be definitively attributed to noise. A comparable Monte Carlo simulation was performed implementing the Gaussian image filter and the result showed no difference in the mean error near the zero noise regime; however, for higher noise levels the mean error decreased by about 25%. The result is shown in Fig. 76.

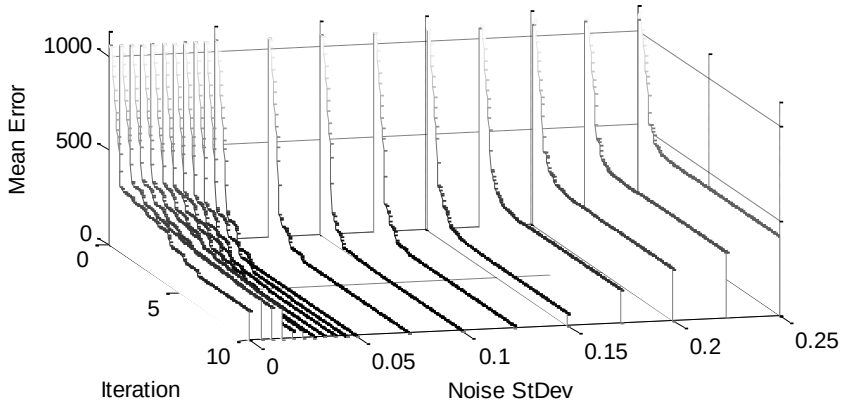


Fig. 72: Monte Carlo results showing the mean error of 25 trials at several noise levels.

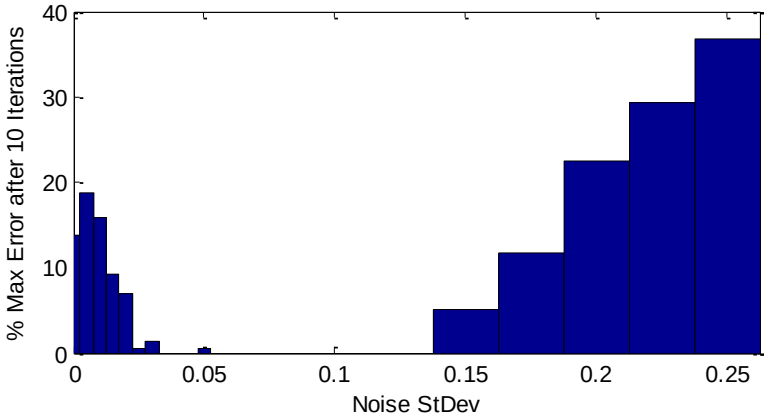


Fig. 73: Monte Carlo results showing the final mean error of 25 trials at several noise levels.

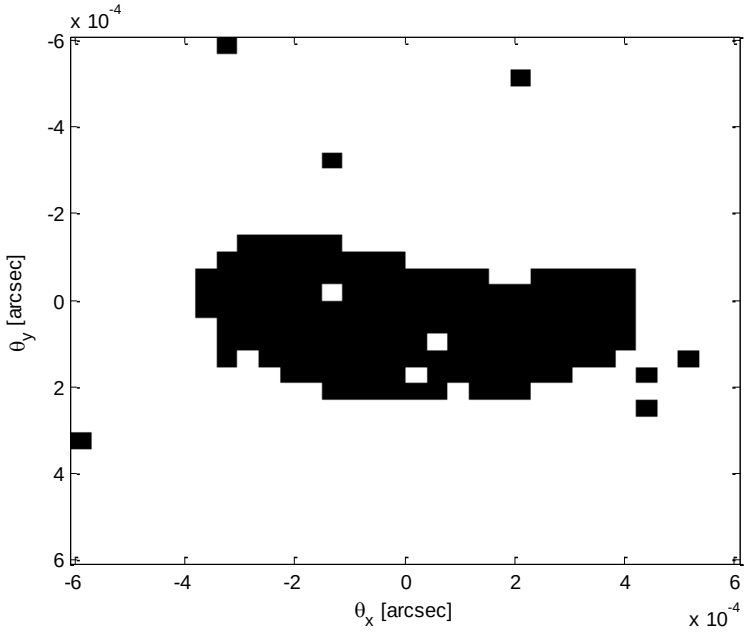


Fig. 74: Example result after 10 iterations with a noise standard deviation of 0.2.

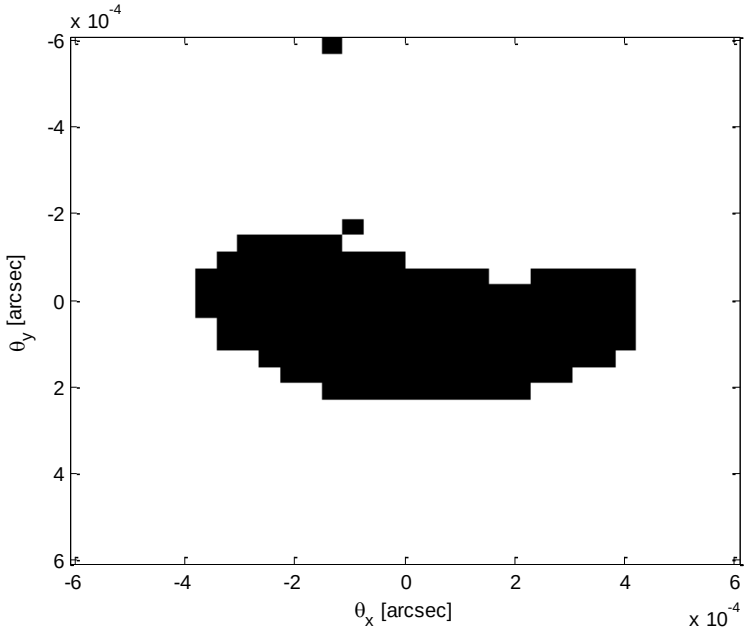


Fig. 75: Example result after 3 iterations with a noise standard deviation of 0.2 and a Gaussian filter applied to the intensity data.

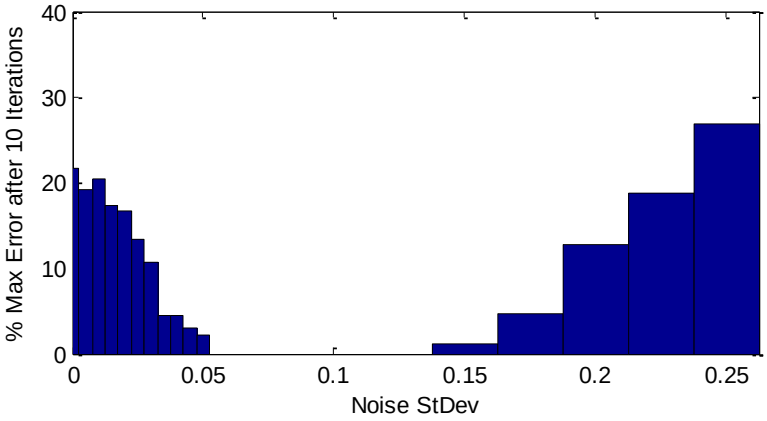


Fig. 76: Monte Carlo results showing the final mean error of 25 trials at several noise levels with a Gaussian image filter applied to the intensity data.

5.5. Data Coverage and Aperture Positioning

The previous discussions used knowledge of the shadow pattern which spanned a rectangular grid. In practice, the shadow pattern is only known at specified positions. Since the shadow pattern is moving at a high velocity relative to the Earth, each aperture can collect intensity measurements in a straight line across the shadow pattern. Shown in Fig. 77 is an example of the detected intensity distribution by 20 equally spaced apertures each 75m apart with 128 measurements along each line. To quantify the quality of the data coverage the ratio of the number of measurements to the number of pixels in the silhouette estimate is defined as

$$\rho = \frac{\#\ \text{Measurements}}{\#\ \text{Pixels in Silhouette}}. \tag{13}$$

The definition of ρ can be used to establish a theoretical minimum number of measurements. The number of constraints on the system—the number of measurements—should be greater than the number of unknowns—the number of pixels in the silhouette estimate.

In the case of the measurements shown in Fig. 77 used to recover a 32×32 pixel silhouette estimate, the ratio ρ is 2.5. Perfect silhouette recovery was achieved after 3.6 iterations as shown in Fig. 78. For half the number of apertures, the intensity data is shown in Fig. 79. In this case ρ is 1.25—slightly above the theoretical minimum. The result is shown in Fig. 80. If the number of apertures in the data collection is reduced to 8, the ratio ρ is reduced to exactly unity. In this case the recovery is theoretically possible; however,

the result shown in Fig. 82 is not clear. The threshold of data collection for practical recovery requires approximately $\rho = 2$.

The analysis method used for the previous results can be used for testing various arrangements and spacing of the apertures and can include error in the position knowledge of each aperture. A specific case shown here is the situation where the apertures are assumed to be equally spaced; however, they are actually randomly perturbed. Consider the situation shown in Fig. 83 where the 20 apertures are assumed to be equally spaced but their actual positions are independently randomly offset by a normal random distribution with a standard deviation of 5 meters. The result shown in Fig. 84 shows only a single pixel is incorrect. If the standard deviation of the position errors is increased to 25 meters, however, the image suffers greatly as shown in Fig. 85 and Fig. 86. An exact quantification of the allowable position error is difficult to obtain because of the subjective nature of the quality of the image; however, the 5 meter and 25 meter perturbation examples shown here probably show a lower and upper bound on the allowable position error.

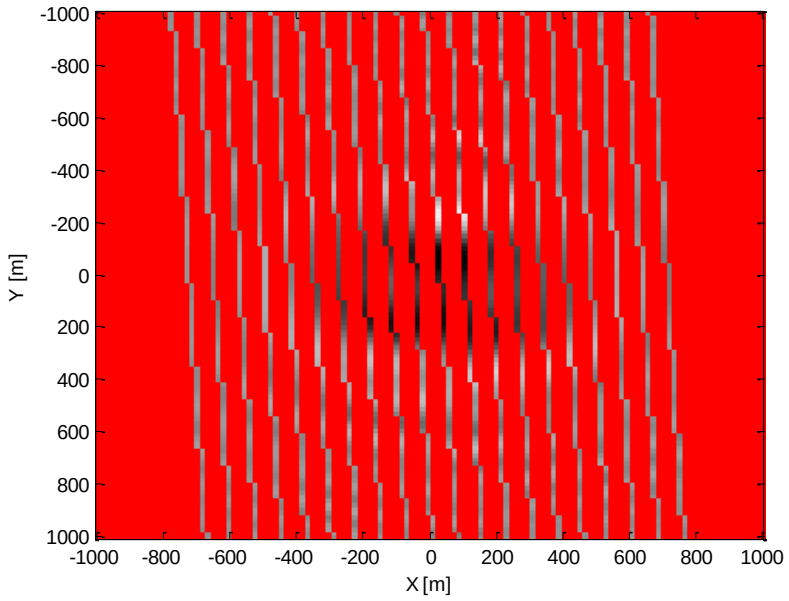


Fig. 77: Data collection pattern for 20 equally spaced apertures each 75m apart. The red regions between the lines of data denotes the absence of data. There are 128 intensity measurements along each apertures path through the shadow pattern. The full intensity distribution is identical to Fig. 68.

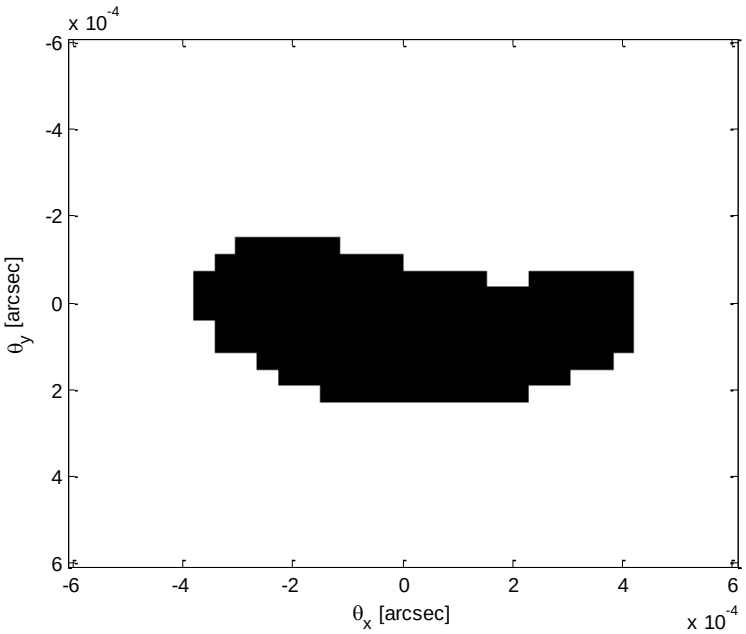


Fig. 78: Silhouette estimate for 20 apertures making 128 measurements each, i.e. ρ=2.5.

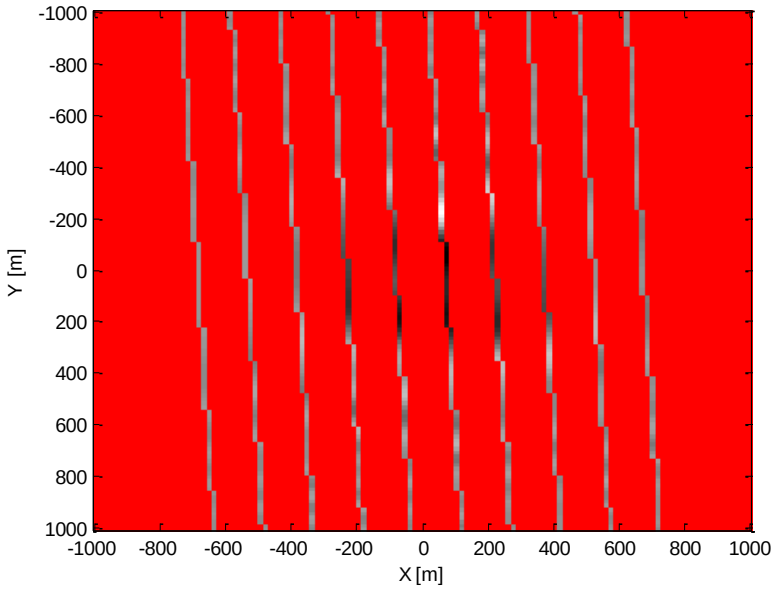


Fig. 79: Data collection pattern for 10 equally spaced apertures each 150m apart. The red regions between the lines of data denotes the absence of data. There are 128 intensity measurements along each apertures path through the shadow pattern. The full intensity distribution is identical to Fig. 68.

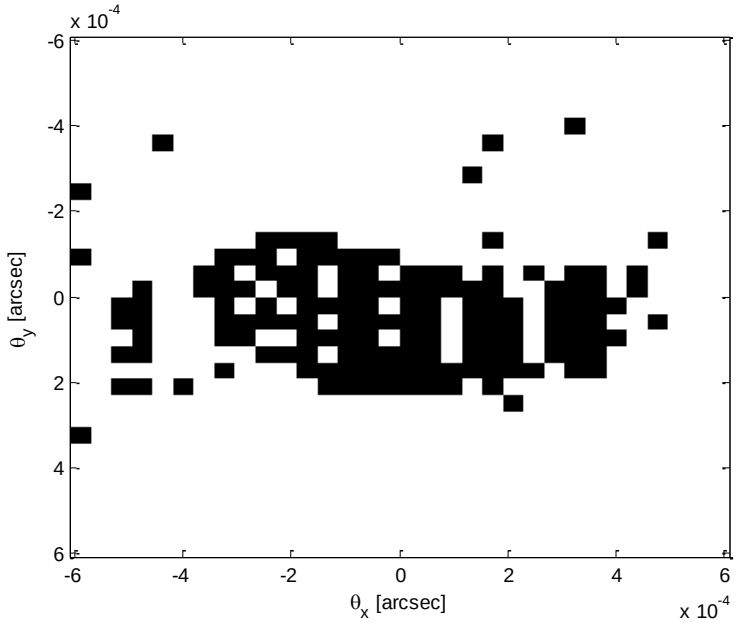


Fig. 80: Silhouette estimate for 10 apertures making 128 measurements each, i.e. ρ=1.25.

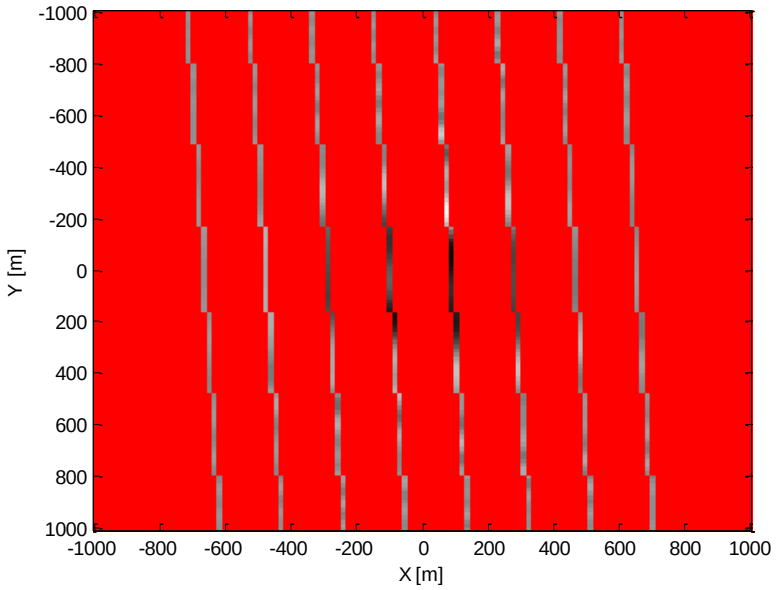


Fig. 81: Data collection pattern for 8 equally spaced apertures each 187.5m apart. The red regions between the lines of data denotes the absence of data. There are 128 intensity measurements along each apertures path through the shadow pattern. The full intensity distribution is identical to Fig. 68.

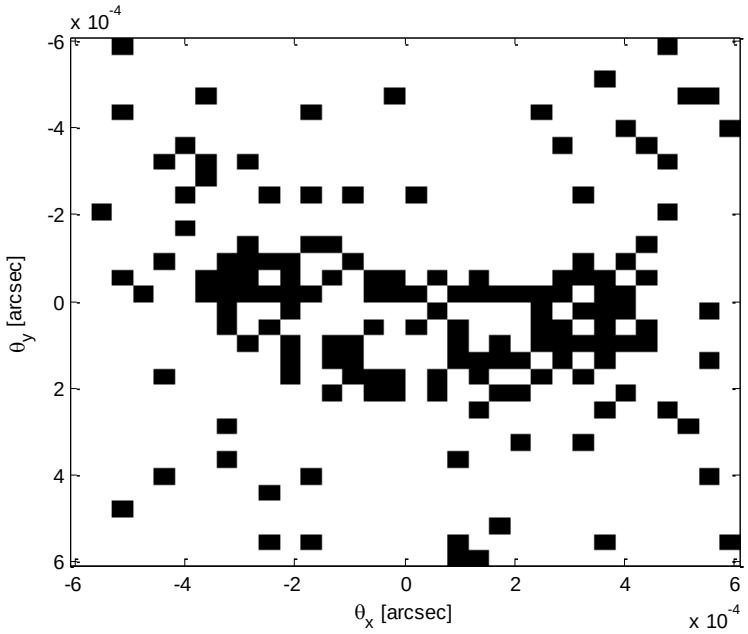
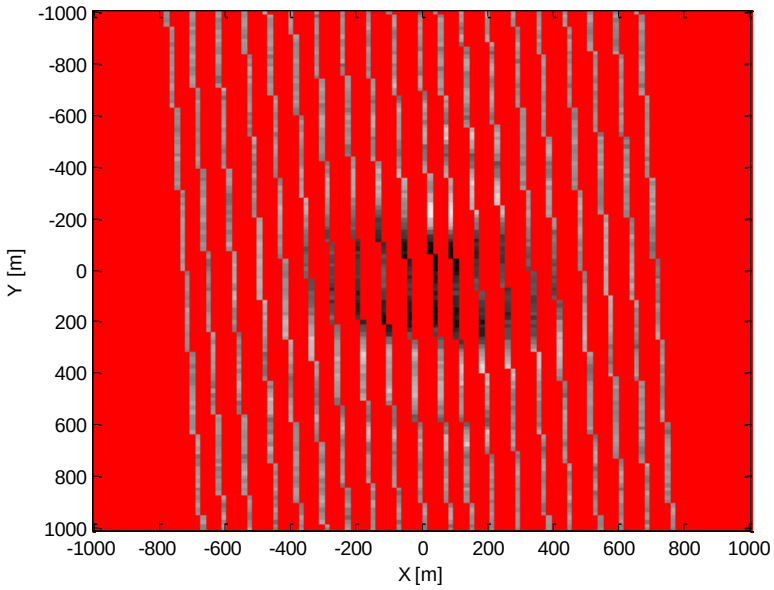
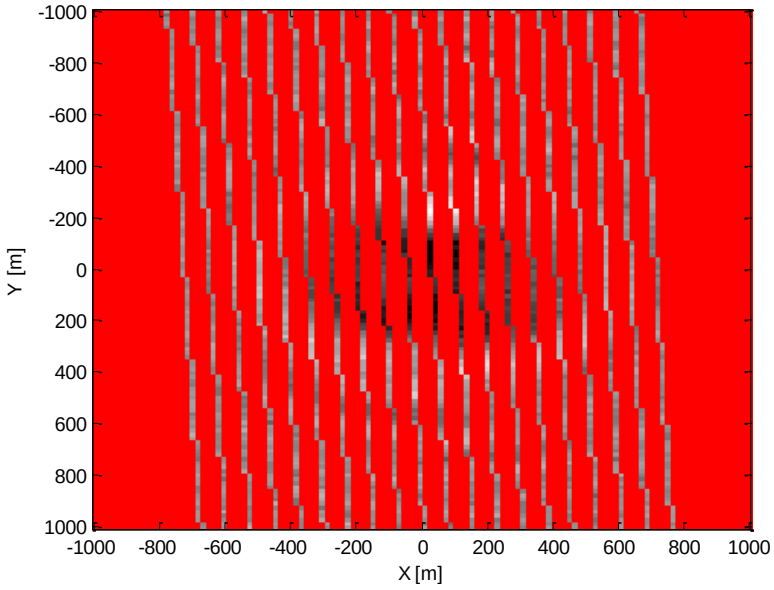


Fig. 82: Silhouette estimate for 8 apertures making 128 measurements each, i.e. ρ=1. This results demonstrates the need for more measurements than the theoretical minimum requirement.



(a)



(b)

Fig. 83: The randomly perturbed positions of the measurements with a standard deviation of 5 meters (a) and the erroneously assumed positions of the measurements (b). The difference is not easily discerned.

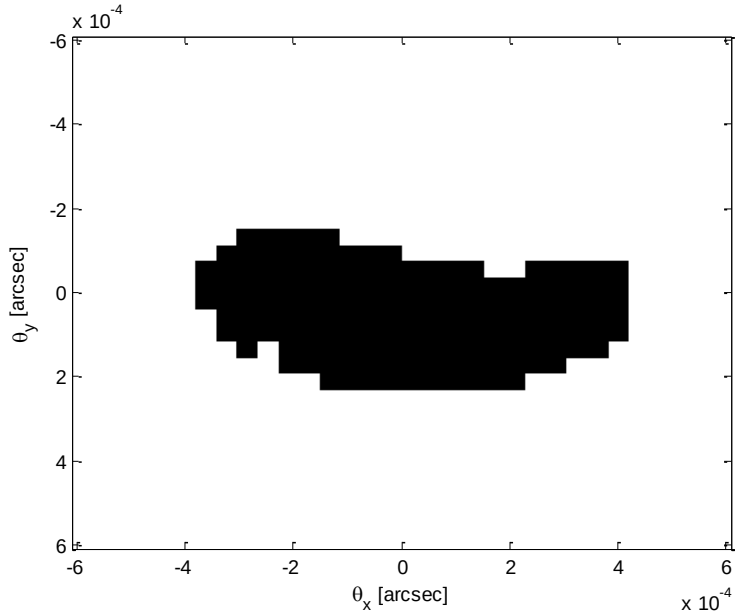
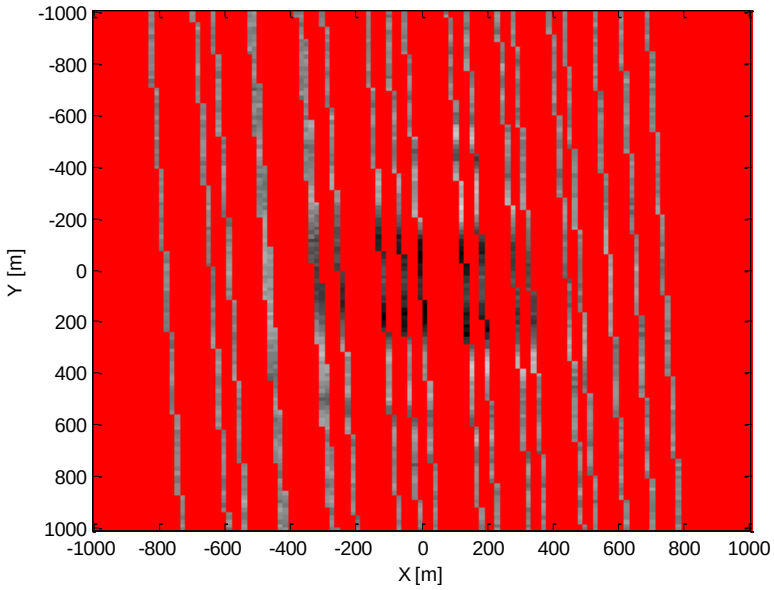
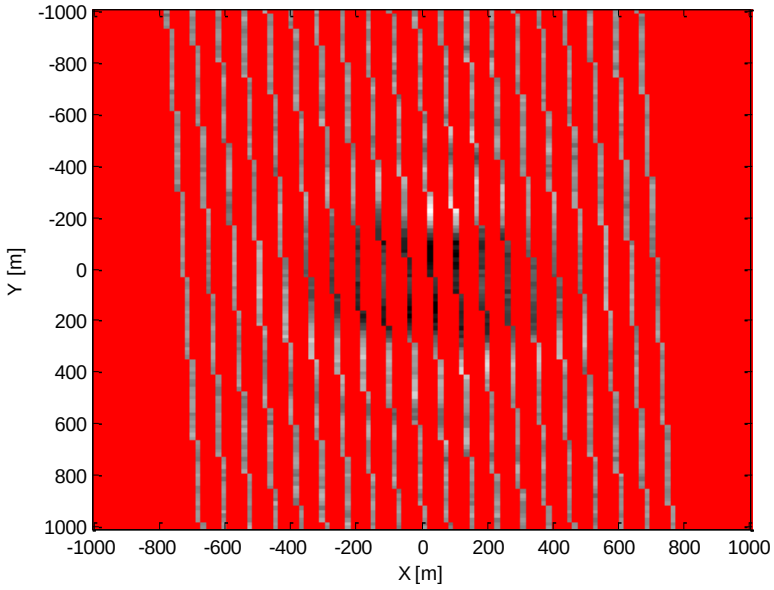


Fig. 84: The estimated silhouette recovered from 20 apertures randomly perturbed with a standard deviation of 5 meters.



(a)



(b)

Fig. 85: The randomly perturbed positions of the measurements with a standard deviation of 25 meters (a) and the erroneously assumed positions of the measurements (b).

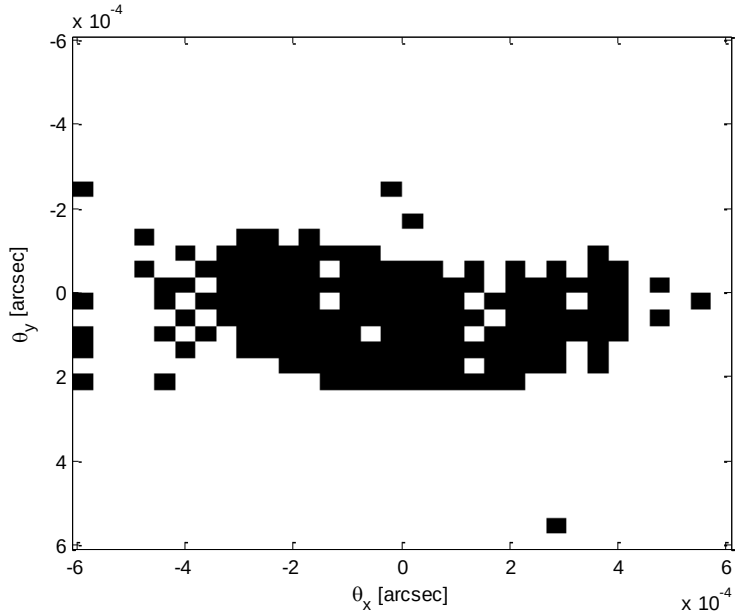


Fig. 86: The estimated silhouette recovered from 20 apertures randomly perturbed with a standard deviation of 25 meters.

6. CONCLUSIONS

In this work the phase retrieval problem was explored with the focus being the various formulations and applications of the problem. The introduction discussed the three portions of the problem. The first portion of the problem is understanding the propagation of a light wave field. The historical evolution of the light propagation model was discussed and the modern Huygens-Fresnel principle introduced. The Huygens-Fresnel principle is a near perfect approximation of the mapping from a light source to the observed wave field some distance away. It was also used here to derive the Van Cittert-Zernike theorem—a cornerstone of interferometry. Both of these derivations showed that the mapping is the Fourier transform which serves as the basis of most phase retrieval solutions.

The second portion of the phase retrieval problem is the method of measuring the wave field's intensity at the observation location. Derivations were shown for how an amplitude interferometer and an intensity correlation interferometer measure the mutual coherence of the wave field. The mutual coherence was shown to be proportional to the magnitude of the Fourier transform of the light source via the Van Cittert-Zernike theorem. This led to the heart of the classical phase retrieval problem: the estimation of the phase of the Fourier transform of the light source based on knowledge of the Fourier transform's magnitude. Similarly, the method of phase retrieval using Gaussian basis functions was shown to be viable. This method capitalized on the optical transfer function's form for a circular aperture as the data collection apparatus. Lastly, the occultation method was

shown to be a practical way of measuring the wave field resulting from an asteroid occulting a star by measuring the intensity of the field at various spatial and temporal locations.

The final portion of the phase retrieval problem is the estimation of the phase of the measured wave field. Three distinct methods of performing the estimation process were explored each within different frameworks.

The first method built upon Gerchberg and Saxton's Error-Reduction method, Fienup's Hybrid Input-Output methods, and subsequent works by various researchers. A deficiency was identified within most current phase retrieval methods in how they handle measurement noise. Since the signal-to-noise level is one of the most prominent difficulties encountered in phase retrieval applications, a method was presented here which is capable of filtering the measured data. Unlike other attempts to solve this problem, the solution here was derived specifically to filter noise and has an analytical rationale. Two types of examples were shown to demonstrate this method's abilities. An example using a typical image was shown in which the measurement noise metric was reduced by 62% and other trials of this method have sometimes (but rarely) shown over 90% noise reduction. Other phase retrieval algorithms showed worse convergence behavior given the same noisy input data and comparable parameter values. Examples were also shown using a two-dimensional case which clearly shows how each algorithm performs projections to converge to the intersection of the Fourier modulus and image support domains. Within these examples, the relaxed Fourier modulus projection showed favorable performance when compared to the classical Fourier modulus projection.

The second method of solving the phase problem was formulated using an image formed with Gaussian bases. A recursive method was derived capable of exactly solving the phase problem and recovering the geometry of the light source. It was only shown to work here for a star cluster, because in its current form the algorithm only works if the true image is formed from less than a dozen Gaussians. A notable benefit of a phase retrieval solution using Gaussian bases is the existence of a definitive error metric. That is, it is clearly known whether or not the algorithm has converged and whether it has converged to the true solution. While the applications of this method are limited because of the restriction on the number of Gaussian bases in the image, it is a step towards a universal analytical solution.

The final method of phase retrieval presented here was a specialized formulation applied to recovering the silhouette of an asteroid occulting a star. Since this method required the use of the Fresnel diffraction equation instead of the Van Cittert-Zernike theorem to describe the light's propagation, the Fourier transform could not be used as a mapping between the source and observation planes. This led to an entirely new method being developed. Where the traditional phase retrieval method has a closed loop, this new method is a guess and check method because the Fresnel diffraction equation cannot be inverted analytically. This method was shown to be able to theoretically recover the silhouette of an asteroid with resolution far surpassing any current optical imaging system. The solution traded efficiency and simplicity found in the classical phase retrieval method for computation expense and computer memory usage. Where the classical phase solution is able to affect every pixel in an iteration by performing Fast Fourier Transforms, this

solution method uses pre-computed solutions to the Fresnel integral and summations of large arrays of numbers. It is thus much more computationally and memory intensive. Since these computations are performed offline, it is suggested that these expenses are not prohibitive. The performance of this algorithm was explored in the presence of three adverse effects. The theoretical and practical minimum number of measurements was derived. The performance was shown to be robust in the presence of noise. Finally, the uncertainty in the location of the apertures was shown to have a small effect. Additionally, it was shown that a low resolution silhouette estimate can be used as the initial guess for a higher resolution estimate. This nested grid approach can greatly reduce the computation time required for high resolution estimates. Additionally, the motivation for such an imaging system was justified based on the statistics of 40 to 140 meter asteroid discovery rates.

The research presented here thoroughly explored the various parts of the phase retrieval problem. Contribution have been made and published which both progress the existing state-of-the-field phase retrieval methods and explore new formulations and applications of the phase retrieval problem.

REFERENCES

[1] R. Trahan and D. Hyland, "Mitigating the effect of noise in the hybrid input–output method of phase retrieval," *Applied Optics,* vol. 52, no. 13, pp. 3031-3037, 2013.

[2] R. Trahan and D. Hyland, "Mitigating the Effect of Noise in Iterative Projection Phase Retrieval.," in *Proceedings of the 2014 Imaging and Applied Optics: Optics and Photonics Conference,* Seattle, 2014.

[3] R. Trahan and D. Hyland, "Phase retrieval of images using Gaussian radial bases," *Applied Optics,* vol. 52, no. 36, pp. 8627-8633, 2013.

[4] R. Trahan and D. Hyland, "Phase Retrieval Applied to Stellar Occultation for Asteroid Characterization," *Applied Optics,* vol. 53, no. 15, pp. 3540-3547, 2014.

[5] S. Singh, Fundamentals of Optical Engineering, Darya Ganji: Discovery Publishing House, 2009.

[6] M. D. Fayer, Absolutely Small: How Quantum Theory Explains Our Everyday World, New York: American Management Association, 2010.

[7] R. Hooke, Micrographia: or, Some Physiological Descriptions of Minute Bodies Made by Magnifying Glasses, London: J. Martyn and J. Allestry, 1665.

[8] P. Hariharan, Basics of Interferometry, 2nd ed., San diego, Ca: Elsevier, 2007.

[9] T. Young, "On the Theory of Light and Colours," *Philosophical Transactions of the Royal Society of London,* vol. 92, pp. 12-48, 1802.

[10] A. Ghatak, Optics, 4th ed., West Patel Nagar: McGraw-Hill, 2009.

[11] G. J. Gbur, Mathematical Methods for Optical Physics and Engineering, Cambridge, GBR: Cambridge University Press, 2010.

[12] Physclips, "Diffraction from a single slit. Young's experiment with finite slits.," [Online]. Available: http://www.animations.physics.unsw.edu.au/jw/light/single-slit-diffraction.html. [Accessed 17 1 2014].

[13] "Young's Double Slit Experiment," [Online]. Available: http://cnx.org/content/m42508/latest/?collection=col11406/latest. [Accessed 04 06 2014].

[14] W. H. Steel, Interferometry, 2nd ed., New York, NY: Cambridge University Press, 1983.

[15] P. H. van Cittert, "Die Wahrscheinliche Schwingung Verteilung in Einer von Einer Lichtquelle Direkt Oder Mittels Einer Linse Beleuchteten Ebene," *Physica,* vol. 1, p. 201, 1934.

[16] C. Foellmi, "Intensity interferometry and the second-order correlation function g^2 in astrophysics," *Astonomy & Astrophysics,* vol. 507, no. 3, pp. 1719-1727, 2009.

[17] E. H. Linfoot, Fourier Methods in Optical Image Evaluation, London: The Focal Press, 1964.

[18] M. Bass, C. DeCusatis, J. Enoch, V. Lakshminarayanan, G. Li, C. MacDonald, V. Mahajan and E. Van Stryland, Handbook of Optics, 3 ed., vol. 2, New York: McGraw-Hill Professional, 2009.

[19] J. A. Roberts, Indirect Imaging: Measurement and Processing for Indirect Imaging, Cambridge: Cambridge University Press, 1984.

[20] G. E. Hale, "Wikipedia," 1922. [Online]. Available: http://en.wikipedia.org/wiki/File:Hooker_interferometer.jpg. [Accessed 30 5 2014].

[21] "Navy Prototype OpticalInterferometer (NPOI)," [Online]. Available: http://usic.wikispaces.com/Navy+Prototype+OpticalInterferometer+(NPOI). [Accessed 30 5 2014].

[22] R. Hanbury Brown and R. Q. Twiss, "A New Type of Iterferometer for Use in Radio Astronomy," *Philosophical Magazine,* vol. 45, no. 366, 1954.

[23] R. Hanbury Brown and R. Q. Twiss, "Interferometry of the Intensity Fluctuations in Light," *Proceedings of the Royal Society of London,* vol. 242, no. 1230, pp. 300-324, 5 Nov 1957.

[24] R. Hanbury Brown, "Stellar Interferometer at Narrabri Observatory," *Nature,* vol. 218, pp. 637-641, 1968.

[25] R. Hanbury Brown and R. Q. Twiss, "A Test of a New Type of Stellar Interferometer on Sirius," *Nature,* vol. 178, no. 4541, pp. 1046-1048, 1956.

[26] R. H. Brown, "Stellar Interferometer at Narrabri Observatory," *Nature,* vol. 218, pp. 637-641, 1968.

[27] J. M. Zuo, I. Vartanyants, M. Gao, R. Zhang and L. A. Nagahara, "Atomic Resolution Imaging of a Carbon Nanotube from Diffraction Intensities," *Science,* vol. 300, p. 1419, 2003.

[28] I. A. Vartanyants, I. K. Robinson, J. D. Onken, M. A. Pfeifer, G. J. Williams, F. Pfeiffer, H. Metzger, Z. Zhong and G. Bauer, "Coherent x-ray diffraction from Quantum dots," *Physical Review B,* vol. 71, p. 245302, 2005.

[29] M. Pfeifer, G. K. Williams, I. A. Vartanyants, R. Harder and I. K. Robinson, "Three-dimensional mapping of a deformation field inside a nanocrystal," *Nature Netters,* vol. 442, pp. 63-66, 2006.

[30] H. H. Rose, "Optics of High-Performance Electron Microscopes," *Science and Technology of Advanced Materials,* vol. 9, no. 014107, pp. 1-30, 2008.

[31] R. G. Lyon, "HST phase retrieval: a parameter estimation," in *Applications of Digital Image Processing XIV,* San Diego, 1991.

[32] J. R. Fienup, J. C. Marron, T. J. Schulz and J. H. Seldin, "Hubble Space Telescope characterized by using," *Applied Optics,* vol. 32, no. 10, pp. 1747-1767, 1 4 1993.

[33] H. A. Arsenault and K. Chalasinska-Macukow, "The Solution to the Phase Retrieval Problem Using the Sampling Theorem," *Optics Communications,* vol. 47, no. 6, pp. 380-386, Oct 1983.

[34] R. W. Gerchbert and W. O. Saxton, "A Practical Algorithm for the Determination of the Phase from Image and Diffraction Plane Pictures," *Optik,* vol. 35, p. 237, 1972.

[35] J. R. Fienup, "Reconstruction of an Object from the Modulus of Its Fourier Transform," *Optics Letters,* vol. 3, no. 1, pp. 27-29, July 1978.

[36] N. C. Gallagher and B. Liu, "Convergence of a Spectrum Shaping Algorithm," *Applied Optics,* vol. 13, no. 11, pp. 2470-2471, 1974.

[37] J. R. Fienup, T. R. Crimmins and W. Holsztynski, "Reconstruction of the support of an object from the support of its autocorrelation," *JOSA,* vol. 72, no. 5, pp. 610-624, 1982.

[38] J. R. Fienup, "Phase Retrieval Algorithms: A Comparison," *Applied Optics,* vol. 21, no. 15, pp. 2758-2769, 1 Aug 1982.

[39] S. Marchesini, H. He, H. N. Chapman, S. P. Hai-Riege, A. Noy, M. R. Howells, U. Weierstall and J. C. Spence, "X-ray image reconstruction from a diffraction pattern alone," *Physical Review B,* vol. 68, no. 140101, pp. 1-4, 2003.

[40] J. Miao, D. Sayre and H. N. Chapman, "Phase retrieval from the magnitude of the Fourier transforms of nonperiodic objects," *J. Opt. Soc. Am. A,* vol. 15, no. 6, pp. 1662-1669, 1998.

[41] D. Dravins, S. LeBohec, H. Jensen and P. Nunez, "Optical intensity interferometry witht eh Cherenkov Telescope Array," *Astroparticle Physics,* vol. 43, pp. 331-347, March 2013.

[42] R. H. Brown and R. Q. Twiss, "The question of correlation between photons in coherent light rays," *Nature,* vol. 178, no. 4548, pp. 1447-1448, 1956.

[43] G. Liu, "Object reconstruction from noisy holograms: multiplicative noise model," *Optics Communications,* vol. 79, no. 6, pp. 402-406, 1990.

[44] R. Bates and D. Mnyama, Advances in Electronics and Electron Physics, vol. 67, P. W. Hawkes, Ed., Toulous: Academic Press, 1987.

[45] C. M. Caves, "Quantum-mechanical noise in an interferometer," *Phy. Rev. D,* vol. 23, pp. 1693-1708, 1981.

[46] S. Marchesini, "A unified evaluation of iterative projection algorithms for phase retrieval," *Review of Scientific Instruments,* vol. 78, no. 011301, pp. 1-11, 2007.

[47] J. P. Abrahams and A. G. W. Leslie, "Methods used in the structure determination of bovine mitochondrial F1 ATPase," *Acta Cryst.*, vol. D, no. 52, pp. 30-42, 1996.

[48] J. von Neumann, Functional Operators, vol. 2, Princeton: Princeton University Press, 1950.

[49] H. H. Bauschke and J. M. Borwein, "On the convergence of von Neumann's alternaing projection algorithm for two sets," *Set-Valued Analysis,* vol. 1, no. 2, pp. 185-202, 1993.

[50] Elser, Viet, "Phase retrieval by iterated projections," *J. Opt. Soc. Am. A,* vol. 20, no. 1, pp. 40-55, 2003.

[51] D. R. Luke, "Relaxed averaged alternating relfections for diffraction imaging," *Inverse Problems*, vol. 21, pp. 37-50, 2005.

[52] A. Levi and H. Stark, "Image restoration by the method of generalized projections with application to restoration from magnitude," *J. Opt. Soc. Am. A,* vol. 1, pp. 932-943, 1984.

[53] M. Kohl, A. A. Minkevich and T. Baumback, "Improved success rate and stability for phase retrieval by including randomized overrelaxation in the hybrid input-output algorithm," *Opt. Express*, vol. 20, pp. 17093-17106, 2012.

[54] G. Liu, "Fourier phase retrieval algorithm with noise constraints," *Signal Processing*, vol. 21, no. 4, pp. 339-347, 1990.

[55] A. V. Oppenhein and R. W. Schafer, Discrete-Time Signal Processing, 3rd ed., Upper Saddle River: Prentice-Hall, 1999.

[56] J. R. Fienup and C. C. Wackerman, "Phase-retrieval stagnation problems and solutions," *J. Opt. Soc. Am. A*, vol. 3, no. 11, pp. 1897-1907, 1986.

[57] S. Marchesini, H. He, H. N. Chapman, S. P. Hau-Riege, A. Noy, M. R. Howells, U. Weierstall and J. Spence, "X-ray image reconstruction from a diffraction pattern alone," *Physical Review B,* vol. 68, no. 140101, 2003.

[58] I. Kodama, M. Yamaguchi, N. Ohyama, T. Honda, K. Shinohara, A. Ito, T. Matsumura, K. Kinoshita and K. Yada, "Image reconstruction from an in-line X-ray hologram," *Optics communications,* vol. 125, pp. 36-42, 1996.

[59] J. Zhao, D. Wang, F. Zhang and Y. Wang, "Hybrid phase retrieval approach for reconstruction of in-line digital holograms without twin image," *Opt. Eng.,* vol. 50, no. 9, 2011.

[60] J. S. Wu, U. Weierstall and J. Spence, "Iterative phase retrieval without support," *Optics Letters,* vol. 29, no. 23, pp. 2737-2739, 2004.

[61] J. R. Fienup and C. C. Wackerman, "Phase-retrieval stagnation problems and solutions," *Journal of the Optical Society of America,* vol. 3, pp. 1897-1907, 1986.

[62] E. Gur, V. Sarafis, I. Falat, F. Vacha, M. Vacha and Z. Zalevsky, "“Super-resolution via iterative phase retrieval for blurred and saturated biological images," *Opt. Express,* vol. 16, pp. 7894-7903, 2008.

[63] R. Fernando, GPU Gems: Programming Techniques, Tips and Tricks for Real-Time Graphics, Boston: Addison-Wesley Professional, 2004.

[64] M. Pharr and R. Fernando, GPU Gems 2: PRogramming Techniques for High-Performance Graphics and General-Purpose Computation, Boston: Addison-Wesley, 2005.

[65] P. S. Heckbert, "Survey of texture mapping," *IEEE Comp. Graph. Appl.,* vol. 6, pp. 56-67, 1986.

[66] D. Shreiner, G. Sellers, J. M. Kessinish and B. M. Licea-Kane, OpenGL Programming Guide, Boston: Addison=Wesley, 2013.

[67] A. R. Smith, "A pixel is not a little square," in *Microsoft Technical Memo 6*, 1995.

[68] E. Hartman and J. Keeler, "Layered neural networks with gaussian hidden units as universal approximations," *Neural Comput.*, vol. 2, pp. 210-215, 1990.

[69] M. Born and E. Wolf, Principles of Optics, 6th ed., Cambridge: Cambridge University Press, 1997.

[70] N. M. Temme, Special functions: An introduction to the classical functions of mathematical physics, New York: Wiley-Interscience, 1996.

[71] J. L. Elliot, Person, Zuluaga, Bosh, Adams, Brothers, Gulbis, Levine, Lockhart, Zangari, Babcock, Dupre, Pasachoff, Souza, Rosing, Secrest, Bright, Dunham, Sheppard, Kakkala, Tilleman, Berger, Briggs, Jacobson, Valleli, Volz, Rapoport, Hart, Brucker, Michel, MAttingly, Zambrano-Marin, Meyer, Wolf, Ryan, Ryan, Morzinsky, Grigsby, Brimacombe, Ragozzine, Montano and Gilmore, "Size and albedo of Kuiper belt object 55636 from a stellar occultation," *Nature*, vol. 465, p. 897–900, 2010.

[72] B. Sicardy, Bellucci, Gendron, Lacombe, LAcour, Lecacheux, Lellouch, Renner, Pau, Roques, Widemann, Colas, Vachier, Vieira Martins, Ageorges, Jainaut, Marco, Beisker, Hummel, Feinstein, Levato, Maury, Frappa, Gaillard, Lavayssiere, Di Sora, Mallia, Masi, Behrend, Carrier, Mousis, Rousselot, Alvarez-Candal, Lazzaro, Veiga, Andrei, Assafin, da Silva Neto, Jacques,

Pimentel, Weaver, Lecampion, Doncel, Momiyama and Tancredi, "Charon's size and an upper limit on its atmosphere from a stellar occultation," *Nature 439,* vol. 439, pp. 52-54, 2006.

[73] B. Sicardy, Brahic, Ferrari, Gautier, Lecacheux, Lellouch, Roques, Arlot, colas, Thuillot, Sevre, Vidal, Blanco, Cristaldi, Buil, Klotz and Thouvenot, "Probing Titan's atmosphere by stellar occultation," *Nature,* vol. 343, pp. 350-353, 1990.

[74] H. E. Schlichting, E. O. Ofek, M. Wenz, R. Sari, A. Gal-Yam, M. Livio, E. Nelan and S. Zucker, "A single sub-kilometre Kuiper belt object from a stellar occultation in archival data," *Nature,* vol. 462, pp. 895-897, 2009.

[75] F. Rogues, Doressoundiram, Dhillon, Marsh, Bickerton, Kavelaars, Moncuquet, Auvergne, Belskaya, Chevreton, Colas, Fernandez, Fitzsimmons, Lecacheux, Mousis, Pau, Peixinho and Tozzi, "Exploration of the Kuiper Belt by High-Precision Photometric Stellar Occultations: First Results," *The Astronomical Journal,* vol. 132, no. 2, 2006.

[76] E. F. Young, "A Fourier optics method for calculating stellar occultation light curves by objects with thin atmospheres," *The Astronomical Journal,* vol. 144, no. 2, pp. 1-13, 2012.

[77] F. Roques, M. Moncuquet and B. Sicardy, "Stellar occultations by small bodies: diffraction effects," *The Astronomical Journal,* vol. 93, no. 6, pp. 1549-1558, 1987.

[78] O. K. Ersoy, Diffraction, Fourier Optics and Imaging, Hoboken: Wiley-Interscience, 2007.

[79] R. E. English and N. George, "Diffraction patterns in the shadows of disks and obstacles," *Applied optics,* vol. 27, no. 8, pp. 1581-1587, 1988.

[80] D. Paganin, Coherent X-Ray Optics, Oxford: Oxford University Press, 2006.

[81] H. Altwaijry and D. Hyland, "Detection and characterization of near Earth asteroids using stellar occultation," in *AAS/AIAA Astrodynamics Specialist Conference,* Hilton Head, South Carolina, 2013.

[82] V. Laude, "Diffraction analysis of pixelated incoherent shadow casting," *Optics Communications,* vol. 138, pp. 394-402, 1997.

[83] M. Abramowitz and I. A. Stegun, Handbook of Mathematical Functions, New York: Dover, 1972, pp. 300-302.

[84] F. L. Pedrotti, L. M. Pedrotti and L. S. Pedrotti, Introduction to Optics, 3rd ed., Harlow: Pearson, 2014.

[85] "JAXA," [Online]. Available: http://www.isas.jaxa.jp/j/snews/2005/1101_hayabusa.shtml. [Accessed 2014].

[86] R. H. Brown and R. Q. Twiss, "A new type of interferometer for use in radio astronomy," *Nature,* vol. 178, no. 4541, pp. 1046-1048, 1956.

[87] R. H. Brown, "Stellar Interferometer at Narrabri Observatory," *Nature,* vol. 218, pp. 637-641, 1968.

[88] D. Dravins, S. LeBohec, H. Jensen and P. Nunez, "Optical intensity interferometry with the Cherenkov Telescope Array," *Astroparticle Physics*, vol. 43, pp. 331-347, March 2013.

APPENDIX I – 2D PROJECTION MATLAB CODE

The following Matlab (2013) functions generate a two-dimensional visualization of several phase retrieval methods. It includes the code to produce the plots shown previously.

ProjectionPhaseRetrieval.m

```
clear
global mode;
global a b;
global C1 C2 R;

%%
% The type of case to run
% Mode 1 is the intersection of two lines
% Mode 2 has a line S domain and non-convex modulus domain
% Mode 3 has a line S domain > 0 and a non-convex modulus domain
mode = 1;

% The number of iterations to perform
iterations = 75;

% The maximum relaxation value reached
MaxLambda = 0.9;
% The iteration to start increasing the relaxation parameter
RelaxationStartIteration = 20;
% The iteration to finish increasing the relaxation parameter
RelaxationFinishIteration = 50;

% The parameter for methods such as the HIO
beta = 0.9;

% Slope of the modulus line for mode 1
a = 0.4;
b = 0;

% Position of the center of the modulus constraint circles for mode 2 and 3
C1 = [0.5, -0.8];
%C1 = [0.5, -1.25];
C1 = [0.5, -1.35];
C2 = [C1(1)+cos(0.25)*R*2, C1(2)-sin(0.25)*R*2, 0.5];

% Radius of the modulus constraint circles for mode 2 and 3
R = 1.25;

% The initial image position
g0 = [2.5, 0.0];
%g0 = [1.0, 0.0];

% Whether to show the error subplot
ShowError = 1;
```

```
% The width of the lines on the plot
LineWidth = 2;

%% Define the image projection operators
if mode == 3
    Ps = @(p) [p(:,1), p(:,2)*0] .* [heaviside(p(:,1)) 1];
    Rs = @(p) (2*Ps(p) - p)  .* [heaviside(p(:,1)) 1];
else
    Ps = @(p) [p(:,1), p(:,2)*0];
    Rs = @(p) (2*Ps(p) - p);
end

%% Find the intersection point

if mode == 1
   Intersection = [0 0];
elseif mode == 2 || mode == 3
    if C1(2) + R > 0
        Angle = asin(-C1(2)/R)
        Intersection(1) = C1(1) + R*cos(Angle);
    else
        Intersection(1) = C1(1);
    end
    Intersection(2) = 0;
end

%% Allocate memory and parameters for plotting
dist = [];
p = [];
colors = [0 0.5 0; 0 0.9 0; 0 1 1; 1 0 1; 0.75 1 0; 1 0.5 0; 0.5 0 1; ];
colors = [colors; colors];
algs = {'ER', 'SF', 'HIO', 'DM', 'ASR', 'HPR', 'RAAR'};

%% Plot domains

figure(1)
if ShowError
    subaxis(2,1,1, 'Spacing', 0.03, 'PaddingTop', 0.01, 'PaddingLeft', 0.075,
'PaddingRight', 0.075, 'PaddingBottom', 0.025, 'Margin', 0);
else
    subaxis(1,1,1, 'Spacing', 0.03, 'PaddingTop', 0.01, 'PaddingLeft', 0.075,
'PaddingRight', 0.075, 'PaddingBottom', 0.025, 'Margin', 0);
end
cla
hold on

if mode == 1
    plot(-2:0.1:3, a*(-2:0.1:3)+b, 'k-' ,'LineWidth', 2)
    text(2.8,1.3,'M','FontSize',12,'Color','k')

    plot(-(2*(mode~=3)):0.1:3, 0*(-(2*(mode~=3)):0.1:3), 'r-' ,'LineWidth', 2)
    text(2.8,0.15,'S','FontSize',12,'Color','r')
    grid on
else
    theta = -0.185:0.01:1.98;
    plot(cos(theta)*R+C1(1), sin(theta)*R+C1(2), 'k-' ,'LineWidth', 2)
    theta = 0.45:0.01:2.8;
    plot(cos(theta)*R+C2(1), sin(theta)*R+C2(2), 'k-' ,'LineWidth', 2)
    text(2.1,-0.8,'M','FontSize',12,'Color','k')

    plot(-(1*(mode~=3)):0.1:4, 0*(-(1*(mode~=3)):0.1:4), 'r-' ,'LineWidth', 2)
    text(3.8,0.25,'S','FontSize',12,'Color','r')
    grid on
end

zlabel('Iteration')
box on
```

```
set(gcf,'Color',[1 1 1])

text(g0(1)-0.01,g0(2)-0.15,['g_0'],'FontSize',10)

if mode == 1
    set(gca, 'XTick', -20:30);
    set(gca, 'YTick', -20:20);
    axis([-2 3 -2 2])
    set(gcf,'Position',[0 0 700 400])
else
    set(gca, 'XTick', -10:10);
    set(gca, 'YTick', -10:10);
    axis([-1 4 -1 5])
    set(gcf,'Position',[0 0 700 800])
end
xlabel('a) Algorithm Trajectories')

%% Compute trajectories

for j=1:length(algs)
    alg = algs(j);
    g = g0;

    for i=1:iterations
        g2 = g(end,:);

        % Compute the current relaxation parameter
        if i > RelaxationFinishIteration
           lambda = MaxLambda;
        elseif i > RelaxationStartIteration
            lambda = MaxLambda * (i-RelaxationStartIteration) /
(RelaxationFinishIteration - RelaxationStartIteration);
        else
            lambda = 0;
        end

        % Perform the projections for the current algorithm
        if strcmp(alg, 'ER')
            g(end+1,:) = Ps(Pm(g2, lambda));
        elseif strcmp(alg, 'SF')
            g(end+1,:) = Rs(Pm(g2, lambda));
        elseif strcmp(alg, 'HIO')
            temp1 = [g2; Pm(g2, lambda); Ps(Pm(g2, lambda)) ];
            temp2 = [g2; Pm(g2, lambda)];
            temp3 = g2 - beta*Pm(g2, lambda);

            temp2(end+1,:) = [temp3(1) temp2(end,2)];
            temp2(end+1,:) = [temp3(1) temp3(2)];
            g(end+1,:) = [temp1(end,1) temp2(end,2)];

%             Uncomment to show the intermediate steps of the HIO
%             if i < 6
%                 plot(temp1(:,1), temp1(:,2),'k-','LineWidth', 1)
%                 plot(temp2(:,1), temp2(:,2),'k-','LineWidth', 1)
%
%                 plot(temp1(end,1), temp1(end,2),'k x','MarkerSize', 10)
%                 plot(temp2(end,1), temp2(end,2),'k x','MarkerSize', 10)
%
%                 plot([temp1(end,1) temp1(end,1)], [temp1(end,2)
temp2(end,2)],'LineStyle',':')
%                 plot([temp2(end,1) temp1(end,1)], [temp2(end,2)
temp2(end,2)],'LineStyle',':')
%             end

        elseif strcmp(alg, 'DM')
            lambda = min(lambda, 0.2 + i/100);
            gs = beta^-1;
```

```
            gm = -beta^-1;
            g(end+1,:) = g2 + beta*Ps( (1+gs)*Pm(g2, lambda)-gs*g2 ) - beta*Pm(
(1+gm)*Ps(g2)-gm*g2, lambda );
        elseif strcmp(alg, 'ASR')
            g(end+1,:) = 1/2* (Rs(Rm(g2, lambda))+g2);
        elseif strcmp(alg, 'HPR')
            g(end+1,:) = (  Rs(Rm(g2, lambda) + (beta-1)*Pm(g2, lambda)) + g(end,:) + (1-
beta)*Pm(g2, lambda)  )/2;
        elseif strcmp(alg, 'RAAR')
            g(end+1,:) = 1/2*beta*(Rs(Rm(g2, lambda))+g2) + (1-beta)*Pm(g2, lambda);
        elseif strcmp(alg, 'LS')

            lambda2 = 0.9;
            lambda2 = min(0.9, 0.2 + i/100);
            if i==1
                g(end+1,:) = Ps(Pm(g2, 0));
            else
                Pm_relaxed = (1-lambda2)*g(end-1,:) + lambda2*g(end,:);

                g(end+1,:) = Ps(Pm_relaxed);
            end
        end
    end

    % Plot the path of the algorithm with the correct color and line style
    if strcmp(algs{j}, 'HPR') || j > 7
        p(end+1) = plot3(g(:,1),g(:,2),0:iterations,'k-.','LineWidth',
LineWidth,'Color',colors(j,:));
    else
        p(end+1) = plot3(g(:,1),g(:,2),0:iterations,'k-','LineWidth',
LineWidth,'Color',colors(j,:));
    end
    plot3(g(1:1:end,1),g(1:1:end,2),0:iterations,'k .','MarkerSize',
15,'Color',colors(j,:))

    % Compute distance to the global intersection or minimum
    dist(end+1,:) = sqrt((g(:,1) - Intersection(1)).^2 + g(:,2).^2);
end

legend('boxoff')
legend(p, algs{1}, algs{2}, algs{3}, algs{4}, algs{5}, algs{6}, algs{7},
'Location','NorthEastOutside')

%% Plot error for each algorithm
if ShowError
    subaxis(2,1,2, 'Spacing', 0.03, 'PaddingTop', 0.01, 'PaddingLeft', 0.075,
'PaddingRight', 0.075, 'PaddingBottom', 0.075, 'Margin', 0);
    cla
    hold on

    for j=1:length(algs)
        if strcmp(algs{j}, 'HPR') || j > 7
            semilogy(dist(j,:)','k-.','LineWidth', 2, 'Color', colors(j,:))
        else
            semilogy(dist(j,:)','LineWidth', 2, 'Color', colors(j,:))
        end
    end
    xlim([0 iterations])
    xlabel(sprintf('Iteration\nb) Distance from the global minimum.'))
    ylabel('Error')
    set(gcf,'Position',[0 0 700 800])
end
```

Pm.m

```
function [ P ] = Pm( p, lambda )

global mode;
global a b;
global C1 C2 R;

if mode == 1
    P = [(p(:,1)+(p(:,2)-b)*a)/(1+a^2), ((p(:,1)+(p(:,2)-b)*a)/(1+a^2))*a + b];
elseif mode == 2 || mode == 3
    dist1 = sqrt( (p(1)-C1(1))^2 + (p(2)-C1(2))^2 );
    dist2 = sqrt( (p(1)-C2(1))^2 + (p(2)-C2(2))^2 );

    if dist1 <= dist2
        angle = atan2(p(2)-C1(2), p(1)-C1(1));

        P = [cos(angle)*R+C1(1), sin(angle)*R+C1(2)];
    else
        angle = atan2(p(2)-C2(2), p(1)-C2(1));

        P = [cos(angle)*R+C2(1), sin(angle)*R+C2(2)];
    end
end

P = (1-lambda)*P + lambda*p;

end
```

Rm.m

```
function [ P ] = Rm( p, lambda )

P = 2*Pm(p, lambda) - p;

end
```

subaxis.m

```
function h=subaxis(varargin)
%SUBAXIS Create axes in tiled positions. (just like subplot)
%   Usage:
%      h=subaxis(rows,cols,cellno[,settings])
%      h=subaxis(rows,cols,cellx,celly[,settings])
%      h=subaxis(rows,cols,cellx,celly,spanx,spany[,settings])
%
% SETTINGS: Spacing,SpacingHoriz,SpacingVert
%           Padding,PaddingRight,PaddingLeft,PaddingTop,PaddingBottom
%           Margin,MarginRight,MarginLeft,MarginTop,MarginBottom
%           Holdaxis
%
%           all units are relative (e.g from 0 to 1)
%
%           Abbreviations of parameters can be used.. (Eg MR instead of MarginRight)
%           (holdaxis means that it wont delete any axes below.)
%
%
% Example:
%
%   >> subaxis(2,1,1,'SpacingVert',0,'MR',0);
%   >> imagesc(magic(3))
%   >> subaxis(2,'p',.02);
%   >> imagesc(magic(4))
```

```
%
% 2001 / Aslak Grinsted  (Feel free to modify this code.)
f=gcf;

Args=[];
UserDataArgsOK=0;
Args=get(f,'UserData');
if isstruct(Args)

UserDataArgsOK=isfield(Args,'SpacingHorizontal')&isfield(Args,'Holdaxis')&isfield(Args,'r
ows')&isfield(Args,'cols');
end
OKToStoreArgs=isempty(Args)|UserDataArgsOK;

if isempty(Args)&(~UserDataArgsOK)
    Args=struct('Holdaxis',0, ...
        'SpacingVertical',0.05,'SpacingHorizontal',0.05, ...
        'PaddingLeft',0,'PaddingRight',0,'PaddingTop',0,'PaddingBottom',0, ...
        'MarginLeft',.1,'MarginRight',.1,'MarginTop',.1,'MarginBottom',.1, ...
        'rows',[],'cols',[]);
end
Args=parseArgs(varargin,Args,{'Holdaxis'},{'Spacing' {'sh','sv'}; 'Padding'
{'pl','pr','pt','pb'}; 'Margin' {'ml','mr','mt','mb'}});

if (length(Args.NumericArguments)>1)
    Args.rows=Args.NumericArguments{1};
    Args.cols=Args.NumericArguments{2};
%remove these 2 numerical arguments
    Args.NumericArguments={Args.NumericArguments{3:end}};
end

if OKToStoreArgs
    set(f,'UserData',Args);
end

switch length(Args.NumericArguments)
   case 0
       return % no arguments but rows/cols....
   case 1
      x1=mod((Args.NumericArguments{1}-1),Args.cols)+1; x2=x1;
      y1=floor((Args.NumericArguments{1}-1)/Args.cols)+1; y2=y1;
   case 2
      x1=Args.NumericArguments{1};x2=x1;
      y1=Args.NumericArguments{2};y2=y1;
   case 4
      x1=Args.NumericArguments{1};x2=x1+Args.NumericArguments{3}-1;
      y1=Args.NumericArguments{2};y2=y1+Args.NumericArguments{4}-1;
   otherwise
      error('subaxis argument error')
end

cellwidth=((1-Args.MarginLeft-Args.MarginRight)-(Args.cols-
1)*Args.SpacingHorizontal)/Args.cols;
cellheight=((1-Args.MarginTop-Args.MarginBottom)-(Args.rows-
1)*Args.SpacingVertical)/Args.rows;
xpos1=Args.MarginLeft+Args.PaddingLeft+cellwidth*(x1-1)+Args.SpacingHorizontal*(x1-1);
xpos2=Args.MarginLeft-Args.PaddingRight+cellwidth*x2+Args.SpacingHorizontal*(x2-1);
ypos1=Args.MarginTop+Args.PaddingTop+cellheight*(y1-1)+Args.SpacingVertical*(y1-1);
ypos2=Args.MarginTop-Args.PaddingBottom+cellheight*y2+Args.SpacingVertical*(y2-1);

if Args.Holdaxis
    h=axes('position',[xpos1 1-ypos2 xpos2-xpos1 ypos2-ypos1]);
else
    h=subplot('position',[xpos1 1-ypos2 xpos2-xpos1 ypos2-ypos1]);
end

set(h,'box','on');
```

```
set(h,'units',get(gcf,'defaultaxesunits'));
set(h,'tag','subaxis');

if (nargout==0) clear h; end;
```

parseArgs.m

```
function ArgStruct=parseArgs(args,ArgStruct,varargin)
% Helper function for parsing varargin.
%
% ArgStruct=parseArgs(varargin,ArgStruct[,FlagtypeParams[,Aliases]])
%
% * ArgStruct is the structure full of named arguments with default values.
% * Flagtype params is params that don't require a value. (the value will be set to 1 if
it is present)
% * Aliases can be used to map one argument-name to several argstruct fields
%
% example usage:
% --------------
% function parseargtest(varargin)
%
% %define the acceptable named arguments and assign default values
% Args=struct('Holdaxis',0, ...
%        'SpacingVertical',0.05,'SpacingHorizontal',0.05, ...
%        'PaddingLeft',0,'PaddingRight',0,'PaddingTop',0,'PaddingBottom',0, ...
%        'MarginLeft',.1,'MarginRight',.1,'MarginTop',.1,'MarginBottom',.1, ...
%        'rows',[],'cols',[]);
%
% %The capital letters define abrreviations.
% %  Eg. parseargtest('spacingvertical',0) is equivalent to  parseargtest('sv',0)
%
% Args=parseArgs(varargin,Args, ... % fill the arg-struct with values entered by the user
%           {'Holdaxis'}, ... %this argument has no value (flag-type)
%           {'Spacing' {'sh','sv'}; 'Padding' {'pl','pr','pt','pb'}; 'Margin'
{'ml','mr','mt','mb'}});
%
% disp(Args)
%
% % Aslak Grinsted 2003

Aliases={};
FlagTypeParams='';

if (length(varargin)>0)
    FlagTypeParams=strvcat(varargin{1});
    if length(varargin)>1
        Aliases=varargin{2};
    end
end

%---------------Get "numeric" arguments
NumArgCount=1;
while (NumArgCount<=size(args,2))&(~ischar(args{NumArgCount}))
    NumArgCount=NumArgCount+1;
end
NumArgCount=NumArgCount-1;
if (NumArgCount>0)
    ArgStruct.NumericArguments={args{1:NumArgCount}};
else
    ArgStruct.NumericArguments={};
end

%--------------Make an accepted fieldname matrix (case insensitive)
Fnames=fieldnames(ArgStruct);
for i=1:length(Fnames)
```

```
    name=lower(Fnames{i,1});
    Fnames{i,2}=name; %col2=lower
    AbbrevIdx=find(Fnames{i,1}~=name);
    Fnames{i,3}=[name(AbbrevIdx) ' ']; %col3=abreviation letters (those that are
uppercase in the ArgStruct) e.g. SpacingHoriz->sh
    %the space prevents strvcat from removing empty lines
    Fnames{i,4}=isempty(strmatch(Fnames{i,2},FlagTypeParams)); %Does this parameter have
a value? (e.g. not flagtype)
end
FnamesFull=strvcat(Fnames{:,2});
FnamesAbbr=strvcat(Fnames{:,3});

if length(Aliases)>0
    for i=1:length(Aliases)
        name=lower(Aliases{i,1});
        FieldIdx=strmatch(name,FnamesAbbr,'exact'); %try abbreviations (must be exact)
        if isempty(FieldIdx)
            FieldIdx=strmatch(name,FnamesFull); %&??????? exact or not?
        end
        Aliases{i,2}=FieldIdx;
        AbbrevIdx=find(Aliases{i,1}~=name);
        Aliases{i,3}=[name(AbbrevIdx) ' ']; %the space prevents strvcat from removing
empty lines
        Aliases{i,1}=name; %dont need the name in uppercase anymore for aliases
    end
    %Append aliases to the end of FnamesFull and FnamesAbbr
    FnamesFull=strvcat(FnamesFull,strvcat(Aliases{:,1}));
    FnamesAbbr=strvcat(FnamesAbbr,strvcat(Aliases{:,3}));
end

%--------------get parameters--------------------
l=NumArgCount+1;
while (l<=length(args))
    a=args{l};
    if ischar(a)
        paramHasValue=1; % assume that the parameter has is of type 'param',value
        a=lower(a);
        FieldIdx=strmatch(a,FnamesAbbr,'exact'); %try abbreviations (must be exact)
        if isempty(FieldIdx)
            FieldIdx=strmatch(a,FnamesFull);
        end
        if (length(FieldIdx)>1) %shortest fieldname should win
            [mx,mxi]=max(sum(FnamesFull(FieldIdx,:)==' ',2));
            FieldIdx=FieldIdx(mxi);
        end
        if FieldIdx>length(Fnames) %then it's an alias type.
            FieldIdx=Aliases{FieldIdx-length(Fnames),2};
        end

        if isempty(FieldIdx)
            error(['Unknown named parameter: ' a])
        end
        for curField=FieldIdx' %if it is an alias it could be more than one.
            if (Fnames{curField,4})
                val=args{l+1};
            else
                val=1; %parameter is of flag type and is set (1=true)....
            end
            ArgStruct.(Fnames{curField,1})=val;
        end
        l=l+1+Fnames{FieldIdx(1),4}; %if a wildcard matches more than one
    else
        error(['Expected a named parameter: ' num2str(a)])
    end
end
```

APPENDIX II – GAUSSIAN PHASE RETRIEVAL

The following Matlab (2013) code generates a Gaussian basis image, computes its Fourier transform, and performs the phase retrieval algorithm on the Fourier transform data.

GaussianPhaseRetrieval.m

```
clear
close all

%% Parameters

DoRetrieval = 1;
ImageRes = 512;
FTRes = 1024;
FTRes_Low = 256;
FTExtent = 0.5;

%% True Image

% Pieades Example
Input = [...
    2  ,105, 290;...
    3  ,220, 275;...
    2  ,298, 324;...
    2  ,397, 275;...
    2.5,336, 199;...
    2.5,380, 170;...
    1  ,405, 215;...
    1  ,333, 143;...
    1  ,100, 265;...
    ];

% Diamond Example
Input = [...
    2  ,300, 200;...
    2  ,175, 250;...
    2  ,350, 250;...
    2  ,300, 300];

Input(:,2) = Input(:,2) - mean(Input(:,2)) + ImageRes/2;
Input(:,3) = Input(:,3) - mean(Input(:,3)) + ImageRes/2;

N_Thetas = size(Input,1);
Input = Input + [zeros(N_Thetas,1) ones(N_Thetas,1)*50 ones(N_Thetas,1)*50];

%% Generate sampled data from continuous image

% Generate a sampled image from the true image
TrueImage = MakeImageFromGaussians(Input, ImageRes);
% Generate a sampled Fourier transform from the true Fourier transform
TrueFT = MakeFTofGaussians(Input, FTRes, FTExtent);

%% Find local maxima of frequency spectrum
```

```
disp('Starting finding Delta Thetas...')

% Generate the spectrum of the sampled Fourier transform
TrueMagSpect = abs(fftshift(fft2(abs(TrueFT).^2)));

% Find maxima in spectrum and store the locations and heights
RawEst_DeltaTheta2 = [];
blocksize = 5;
tic
for i = 1+blocksize:(FTRes-blocksize)
    for j=1+blocksize:(FTRes-blocksize)
        if max(max(TrueMagSpect(j-blocksize:j+blocksize,i-blocksize:i+blocksize))) ==
TrueMagSpect(j,i)
            RawEst_DeltaTheta2(end+1, :) = [i j TrueMagSpect(j,i)];
        end
    end
end

% Sort maxima by their amplitude
RawEst_DeltaTheta2 = sortrows(RawEst_DeltaTheta2, -3);
% Remove trivial Gaussian at origin
RawEst_DeltaTheta2(1,:) = [];
% Only keep the n*(n-1) largest Gaussians
RawEst_DeltaTheta2 = RawEst_DeltaTheta2(1:N_Thetas*(N_Thetas-1),1:3);
% Translate into center of field of view
RawEst_DeltaTheta2(:,1) = RawEst_DeltaTheta2(:,1) - (FTRes)/2-1;
RawEst_DeltaTheta2(:,2) = -RawEst_DeltaTheta2(:,2) + (FTRes)/2+1;
% Copy positions only
RawEst_DeltaTheta = RawEst_DeltaTheta2(1:2:end,1:2);

Est_N_Thetas = N_Thetas;

disp('Finished finding Delta Thetas')

%% Determine thetas from delta thetas

if DoRetrieval
    disp('Starting Solver...')

    tic

    % Solve for the estmated Gaussian locations
    [Est_Theta, Est_DeltaTheta, Est_Error] = FindPairs( RawEst_DeltaTheta, Est_N_Thetas )

    % Translate image to center
    Est_Theta(:,1) = Est_Theta(:,1) - mean(Est_Theta(:,1)) + ImageRes/2;
    Est_Theta(:,2) = Est_Theta(:,2) - mean(Est_Theta(:,2)) + ImageRes/2;

    % Create sampled versions of the estimated image and FT
    EstImage = MakeImageFromGaussians([ones(Est_N_Thetas,1)*3 Est_Theta], ImageRes);
    EstImageFT = single(MakeFTofGaussians([ones(Est_N_Thetas,1) Est_Theta], FTRes,
FTExtent));

    toc

    % Plot estimated image
    EstImage = fliplr(EstImage);
    figure(4)
    image(1:ImageRes, 1:ImageRes, EstImage / max(max(abs(EstImage))) * 64)
    title('Estimated Image')
    colormap gray

    disp('Solver Finished')
end

%% Output
```

```
figure(1)
image(TrueImage / max(max(abs(TrueImage))) * 64)
title('True Image')
colormap gray

figure(2)
image(1:ImageRes, 1:ImageRes, (abs(TrueFT) / max(max(abs(TrueFT)))).^0.5 * 64)
title('True Image FT')
colormap gray

figure(3)
image((TrueMagSpect/max(max(abs(TrueMagSpect)))*64).^1.5*4)
title('FT Mag Spectrum')
colormap gray
```

MakeFTofGaussians.m

```
function [ FT ] = MakeFTofGaussians( Gaussians, FTRes, FTExtent )

N_Gaussians = size(Gaussians,1);

FT = zeros(FTRes, FTRes);

[U, V] = meshgrid(linspace(-FTExtent,FTExtent,FTRes),linspace(-FTExtent,FTExtent,FTRes));

for g=1:N_Gaussians
    FT = FT + exp(-2*pi*1i*(Gaussians(g,2)*U + Gaussians(g,3)*V)) .*
2*Gaussians(g,1)^2*pi .* exp(-2*Gaussians(g,1)^2*pi^2*(U.^2+V.^2));
end

end
```

MakeImageFromGaussians.m

```
function [ Image ] = MakeImageFromGaussians( Gaussians, ImageRes )

N_Gaussians = size(Gaussians,1);

Image = zeros(ImageRes,ImageRes);

[x, y] = meshgrid(linspace(1,ImageRes,ImageRes),linspace(1,ImageRes,ImageRes));

for g=1:N_Gaussians
    Image = Image + exp(-((x - Gaussians(g,2)).^2 + (y -
Gaussians(g,3)).^2)./(2*Gaussians(g,1)^2));
end

end
```

FindPairs.m

```
function [Est_Theta, Est_DeltaTheta, Est_Error] = FindPairs( RawEst_DeltaTheta,
Est_N_Thetas )

RawEst_DeltaTheta = [0,0; RawEst_DeltaTheta; -RawEst_DeltaTheta];
N_DeltaThetas = length(RawEst_DeltaTheta);

MainImageIndices = 3:N_DeltaThetas;

%% Find initial increments
Increment = RawEst_DeltaTheta(2,:) - RawEst_DeltaTheta(1,:);

Indices = [];
```

```
for i=1:N_DeltaThetas
    for j=1:N_DeltaThetas

        TestIncrement = RawEst_DeltaTheta(j,:) - RawEst_DeltaTheta(i,:);
        %abs(TestIncrement - Increment)
        if(max(abs(TestIncrement - Increment)) <= 5)
            Indices(end+1,:) = [i j];
        end

    end
end

%% Recursively loop through next increments

[Success, Indices] = NextIndex(MainImageIndices, RawEst_DeltaTheta, Indices,
N_DeltaThetas, Est_N_Thetas);

%% Output
Est_Theta = RawEst_DeltaTheta(Indices(1,:),:);
Est_DeltaTheta = 0;
Est_Error = 0;

end


function [Success, Indices] = NextIndex(MainImageIndices, RawEst_DeltaTheta, Indices,
N_DeltaThetas, Est_N_Thetas)

Indices = [Indices zeros(size(Indices,1), 1)];

Success = 0;
for AttempedIndex = MainImageIndices
    Increment = RawEst_DeltaTheta(AttempedIndex,:) - RawEst_DeltaTheta(Indices(1,end-
1),:);

    Indices(:,end) = 0;

    for i=1:size(Indices,1)
        for j=1:N_DeltaThetas
            TestIncrement = RawEst_DeltaTheta(j,:) - RawEst_DeltaTheta(Indices(i,end-
1),:);

            if(TestIncrement(1) == Increment(1) && TestIncrement(2) == Increment(2))
                Indices(i,end) = j;
            end
        end
    end

    if sum(Indices(:,end)>0) >= Est_N_Thetas
        Success = 1;

        if size(Indices,2) < Est_N_Thetas
            [Success2, Indices2] = NextIndex(MainImageIndices(MainImageIndices ~=
AttempedIndex), RawEst_DeltaTheta, Indices(Indices(:,end) ~= 0, :), N_DeltaThetas,
Est_N_Thetas);

            if Success2
                Success = 1;
                Indices = Indices2;
            else
                Success = 0;
            end
        else

        end

        if Success
```

```
            break;
        else
            % Subsequent iteration failed, this maxima was incorrect
        end
    end
end

end
```

APPENDIX III – OCCULTATION PHASE RETRIEVAL

The following Matlab (2013) functions use a black and white bitmap image as the silhouette for an asteroid and produces a shadow pattern based on the specified parameters. The solver then attempts to recover the silhouette from the shadow pattern. The code includes the functionality to produce plots of each step of the process. The code here uses the helper functions in Appendix I for plotting. Included below are functions to compute Fresnel integrals. The function generateTables.m must be called before using the Fresnel integral functions. Included after the code are three bitmaps of the silhouette of Itokawa at 32x32, 64x64 and 128x128 resolutions.

OccultationPhaseRetrieval.m

```
%% Occultation Phase Retrieval
clear
clc
% Start a matlab pool to use multipls cores and parfor loops
if(matlabpool('size') < 2)
    matlabpool
end

%% Paramters

% true or false if should perform retrieval otherwise just shows the shadow pattern
DoRetrieval = 1;
% The maximum number of retrieval iterations
Iterations = 3;
% Whether to randomly pick a second pixel to test at each iteration
RandomlyTestSecondPixel = 0;

% Plot options
ShowTrueShadowPattern = 1;
ShowNoisyShadowPattern = 1;
ShowSensorData = 0;
ShowSensorPattern = 0;
ShowErrorHistory = 0;
ShowTrueErrorHistory = 0;
ShowCurrentIteration = 1;

% The number of apertures, use inf for unrestricted data
NSensors = inf;
% The angle from vertical of the sensor paths
SensorPatternAngle = 0.05; % [rad]
```

```
% The standard deviation of the error in the position of the sensors
SensorPositionErrorStdDev = 0; % [meters]

% Resolution of the image of the silhouette
ImgRes = 32; % [pixels]
% Resolution of the shadow pattern
IntRes = 128; % [pixels]
% SNR of the intensity measurements
IntensitySNR = 10;

% Light wavelength
lambda = 5.5e-7; % [meters]

% The filename of the image used to create the shadow pattern
ImageFilename = ['Itokawa' num2str(ImgRes) '.bmp'];

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Enter range to the asteroid
z = 1.00 * 149597871000; % [meters] (### * 1[au in meters])
F = (535/2)^2/z/lambda;
%%%%%%%%%%%%%%%   or   %%%%%%%%%%%%%%
% Enter Fresnel Number
%F = 0.87;
%z = (535/2)^2/lambda/F;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

zl = z/lambda;

%% Compute dimensional data and allocate memory
% Compute the standard deviation of the noise based on the SNR
if IntensitySNR == inf
    noise_stddev = 0;
else
    noise_stddev = fzero(@(sigma) IntensitySNR - 1/(2*sigma*sqrt(1+2*sigma^2)),0.1)
end
ViewSize = 535/(80/128); % [m]
PixSize = 535/(80/128*ImgRes); % [m]
AngRes = PixSize / z; % [rad]
AngView = AngRes * ImgRes; % [rad]

ImgExtent = ViewSize / 2 / z; % [rad]
IntExtent = 1000 / z; % [m/z]

AngRes_AS = AngRes * 180/pi*60*60; % [arcsec]
AngView_AS = AngView * 180/pi*60*60; % [arcsec]

E = @(x)(fresnelS(x)+1i*fresnelC(x));
ImgPixelSize = ImgExtent*2/(ImgRes-1);
IntPixelSize = IntExtent*2/(IntRes-1);
[X,Y] = meshgrid(-ImgExtent:ImgPixelSize:ImgExtent,-ImgExtent:ImgPixelSize:ImgExtent);
[U,V] = meshgrid(-IntExtent:IntPixelSize:IntExtent,-IntExtent:IntPixelSize:IntExtent);

Js = zeros(size(U,2), size(U,1), ImgRes, ImgRes);

%% Generate Image
Img = mean(imread(ImageFilename),3)/255;

%% Generate diffraction pattern for every pixel as if they're all lit.
tic
disp('Starting generating pixel contributions to shadow pattern...')
parfor i = 1:ImgRes
    for j = 1:ImgRes
        Js(:,:,i,j) = 1/2* ( ...
            E(sqrt(2*zl)*(ImgPixelSize + X(i,j) - U)) -...
            E(sqrt(2*zl)*(             + X(i,j) - U)) ...
            ) .* ( ...
            E(sqrt(2*zl)*(ImgPixelSize + Y(i,j) - V)) -...
```

```
            E(sqrt(2*zl)*(             + Y(i,j) - V)) ...
            );
    end
end
disp('Finished generating pixel contributions to shadow pattern.')
toc

%% Generate true diffraction pattern for image using pixel contributions.
J = 1i - squeeze(sum(Js(:,:,~logical(Img)),3));
Psi = abs(J).^2;

Psi2 = abs(Psi + (randn(size(Psi)) + 1i*randn(size(Psi)))*noise_stddev);

% Plots
figure(1)
colormap gray
image(X(1,:)*180/pi*60*60,Y(:,1)*180/pi*60*60,Img*64)
title('True Image: \Gamma(x_a,y_a)')
xlabel('\theta_x [arcsec]')
ylabel('\theta_y [arcsec]')

if ShowTrueShadowPattern
    if 0
        figure(2)
        subaxis(1,1,1, 'Spacing', 0.02, 'PaddingTop', 0.08, 'PaddingLeft', 0.13,
'PaddingRight', 0.3, 'PaddingBottom', 0.12, 'Margin', 0);
        colormap gray
        image(U(1,:)*z,V(:,1)*z,Psi / max(max(Psi))*64)
        title(sprintf('z = %.2f [au], F = %.2f',z/149597871000,F),'FontSize',15)
        xlabel('X [m]','FontSize',15)
        ylabel('Y [m]','FontSize',15)

        subaxis(1,1,1, 'Spacing', 0.02, 'PaddingTop', 0.08, 'PaddingLeft', 0.82,
'PaddingRight', 0.15, 'PaddingBottom', 0.12, 'Margin', 0);
        plot([0 0], [eps z/149597871000],'b-','LineWidth',10,'Color',[60/255 0 0])
        ylim([0.008 50])
        ylabel('z [au]','FontSize',15)
        set(gca, 'YScale', 'log')
        set(gca, 'XTick', []);
        set(gca, 'YTick', [0.01 0.02 0.05 0.1 0.2 0.5 1 2 5 10 20 50]);
        set(gca, 'YMinorTick', 'off');

        subaxis(1,1,1, 'Spacing', 0.02, 'PaddingTop', 0.08, 'PaddingLeft', 0.96,
'PaddingRight', 0.01, 'PaddingBottom', 0.12, 'Margin', 0);
        plot([0 0], [eps F],'b-','LineWidth',10,'Color',[60/255 0 0])
        ylim([0.008 100])
        ylabel('F','FontSize',15)
        set(gca, 'YScale', 'log')
        set(gca, 'XTick', []);
        set(gca, 'YTick', [0.01 0.02 0.05 0.1 0.2 0.5 1 2 5 10 20 50 100]);
        set(gca, 'YMinorTick', 'off');


        temp = get(gcf,'PaperPosition')
        set(gcf,'PaperPosition', [0 0 10.66666666666 6])
        print(gcf, '-dpng', ['F' sprintf('%07.3f',F) '.png']);
    else
        figure(2)
        colormap gray
        image(U(1,:)*z,V(:,1)*z,Psi / max(max(Psi))*64)
        xlabel('X [m]')
        ylabel('Y [m]')
        title('True Shadow Pattern')
    end
end

if ShowNoisyShadowPattern
```

```
    figure(3)
    colormap gray
    image(U(1,:)*z,V(:,1)*z,Psi2 / max(max(Psi2))*64)
    title(['Noisy Shadow Pattern'])
    xlabel('X [m]')
    ylabel('Y [m]')
end

if NSensors == inf
    SensorPattern = ones(IntRes, IntRes);
    SensorPatternPix = [];

    for x = 1:IntRes
        for y = 1:IntRes
            SensorPatternPix(end+1,:) = [((x-1)*IntRes) + y, ((x-1)*IntRes) + y];
        end
    end
else
    SensorPattern = zeros(IntRes, IntRes);
    SensorPatternPix = [];

    for i=0:NSensors-1

        MidPos = IntRes/2 + (i-NSensors/2+0.5)/NSensors*2*IntRes/2 * 0.75;
        MidPosEr = IntRes/2 + (i-NSensors/2+0.5)/NSensors*2*IntRes/2 * 0.75 + randn(1) *
IntRes / (IntExtent*2*z) * SensorPositionErrorStdDev;

        for y=1:IntRes

            x = round(MidPos + sin(SensorPatternAngle) * (y-IntRes/2));
            xEr = round(MidPosEr + sin(SensorPatternAngle) * (y-IntRes/2));

            if x <= IntRes && x > 0 && y <= IntRes && y > 0 && xEr <= IntRes && xEr > 0
                SensorPatternPix(end+1,:) = [((x-1)*IntRes) + y, ((xEr-1)*IntRes) + y];
                SensorPattern(y,xEr) = 1;
            end
        end
    end
end

if ShowSensorPattern
    figure(7)
    colormap gray
    image(U(1,:)*z,V(:,1)*z,SensorPattern*64)
    xlabel('X [m]')
    ylabel('Y [m]')
    title('Sensor Pattern')
end
if ShowSensorData
    figure(8)
    colormap([[linspace(0,1,64) 1]', [linspace(0,1,64) 0]', [linspace(0,1,64) 0]']);
    image(U(1,:)*z,V(:,1)*z, (SensorPattern.*Psi2 / max(max(Psi2))*64 + (1-
SensorPattern)*65) )
    xlabel('X [m]')
    ylabel('Y [m]')
    title('Sensor Data')
end

%% Performs the phase retrieval
if DoRetrieval
    %% Initialize plots
    if ShowErrorHistory
        figure(5)
        temp = get(gcf,'Position');
        set(gcf,'Position',[temp(1) temp(2) temp(3) 250])
        %set(gcf,'PaperPosition', [temp(1) temp(2) temp(3) 250])
    end
```

```
    if ShowTrueErrorHistory
        figure(6)
        temp = get(gcf,'Position');
        set(gcf,'Position',[temp(1) temp(2) temp(3) 250])
    end

    %% Allocate Memory
    Error = inf;
    ErrorHistory = [];
    TrueErrorHistory = [];
    PsiEst = zeros(size(U));
    Trials = 0;

    %% Initial Estimate

    % Blank image
    ImgEst = ones(size(X));

    % if(exists('ImgEst_Save'))
    %    ImgEst = ImgEst_Save;
    % end

    % Read in 32x32 image and resize it to 64x64 to use as the input to the 64x64
estimation
    % Img32 = mean(imread('Itokawa32.bmp'),3)/255;
    % ImgEst = round(imresize(Img32,[64 64],'bilinear'));

    %% Solve
    i = 0;
    j = 1;
    tic
    for Iteration = 1:Iterations
        for Trial = 1:ImgRes^2

            i = i + 1;
            if i > ImgRes
                i = 1;
                j = j + 1;
                if j > ImgRes
                    j = 1;
                end
            end

            TrialImgEst = ImgEst;
            TrialImgEst(i,j) = ~ImgEst(i,j);

            if RandomlyTestSecondPixel
                ir = 0;
                jr = 0;
                if rand(1) > 0.9 && Iteration == 1
                    ir = round(rand(1)*(ImgRes-1)+1);
                    jr = round(rand(1)*(ImgRes-1)+1);
                    TrialImgEst(ir,jr) = ~ImgEst(ir,jr);
                end
            end

            TrialPsiEst = abs(1i - squeeze(sum(Js(:,:,~logical(TrialImgEst)),3))).^2;
            NewError = sum(sum(abs(Psi2(SensorPatternPix(:,1)) -
TrialPsiEst(SensorPatternPix(:,2)))));

            if Trials == 0
                Error = NewError;
            end

            if NewError < Error
                ImgEst = TrialImgEst;
```

```
                PsiEst = TrialPsiEst;
                Error = NewError;
            end

            % Adding eps makes the value never exactly zero so it will plot on log scale
correctly
            TrueErrorHistory(end+1) = sum(sum(abs(Psi - PsiEst))) + eps;
            ErrorHistory(end+1) = Error + eps;

            if rem(Trial,ImgRes) == 0 || Trial == 1 || 1
                if ShowErrorHistory
                    figure(5)
                    semilogy((1:length(ErrorHistory))/ImgRes/ImgRes,
ErrorHistory,'LineWidth', 3)
                    xlim([0 Iterations])
                    ylim([max([10 10^floor(log10(min(ErrorHistory)))])
10^ceil(log10(max(ErrorHistory)))])
                    xlabel('Iteration')
                    ylabel('Error')
                end
                if ShowCurrentIteration
                    figure(100)
                    clf
                    axes('Position',[0.25 0.45 0.5 0.5])
                    temp = ImgEst;
                    temp(i,j) = 0.5;
                    if RandomlyTestSecondPixel
                        if(ir > 0 && jr > 0)
                            temp(ir,jr) = 0.5;
                        end
                    end
                    image(X(1,:)*180/pi*60*60,Y(:,1)*180/pi*60*60,temp*4);
                    colormap([0 0 0; 1 0 0; 1 1 1])
                    xlabel('\theta_x [arcsec]','FontSize',18)
                    ylabel('\theta_y [arcsec]','FontSize',18)

                    axes('Position',[0.25 0.1 0.5 0.25])
                    semilogy((1:length(ErrorHistory))/ImgRes/ImgRes,
ErrorHistory,'LineWidth', 3)
                    xlim([0 Iterations])
                    ylim([max([10 10^floor(log10(min(ErrorHistory)))])
10^ceil(log10(max(ErrorHistory)))])
                    xlabel('Iteration','FontSize',18)
                    ylabel('Error','FontSize',18)
                    temp = get(gcf,'Position');
                    set(gcf,'Position',[temp(1) temp(2) temp(3) 700])
                    set(gcf,'PaperPosition', [0 0 576 432])
                    %print(gcf, '-dpng', ['Iter' num2str(sprintf('%5.5d',Trials))
'.png']);
                end
                if ShowTrueErrorHistory
                    figure(6)
                    semilogy((1:length(TrueErrorHistory))/ImgRes/ImgRes,
TrueErrorHistory,'LineWidth', 3)
                    xlim([0 Iterations])
                    ylim([max([10 10^floor(log10(min(TrueErrorHistory)))])
10^ceil(log10(max(TrueErrorHistory)))])
                    xlabel('Iteration')
                    ylabel('True Error')
                end

                pause(eps);
                if TrueErrorHistory(end) < 10*eps
                    break
                end
            end
```

```
            Trials = Trials + 1;
        end

        if TrueErrorHistory(end) < 10*eps
            break
        end
    end
    toc
end
```

fresnelS.m

```
function FSint = fresnelS(X,fresnelType)
% fresnelS - Fresnel sine integrals, S(X), S1(X), or S2(X)
% usage: FSint = fresnelS(X,fresnelType)
%
% Fresnel sine integrals fall into three classes, simple
% transformations of each other. All three types described
% by Abramowitz & Stegun are supported.
%
% The maximum error of this code has been shown to be less
% than (approximately) 1.5e-14 for any value of X.
%
% arguments: (input)
%  X - Any real, numeric value, vector, or array thereof.
%      X is the upper limit of the Fresnel sine integral.
%
%  fresnelType - scalar numeric flag, from the set {0,1,2}.
%
%      The type 0 Fresnel sine integral (A&S 7.3.1)
%        S(x) = \int_0^x sin(pi*t^2/2) dt,
%
%      Type 1 (A&S  7.3.3a)
%        S_1(x) = \sqrt(2/pi) \int_0^x sin(t^2) dt
%
%      Type 2 (A&S  7.3.3b)
%        S_2(x) = \sqrt(1/2/pi) \int_0^x sin(t) / \sqrt(t) dt
%
% arguments: (output)
%  FSint - array of the same size and shape as X, containing
%      the indicated Fresnel sine integral values.
%
% REFERENCES
% [1] Abramowitz, M. and Stegun, I. A. (Eds.). "Error Function and Fresnel
%     Integrals." Ch. 7 in Handbook of Mathematical Functions with
%     Formulas, Graphs, and Mathematical Tables, 9th printing. New York:
%     Dover, pp. 295-329, 1970.
%
% [2] Mielenz, K. D.; "Computation of Fresnel Integrals", Journal of
%     Research of the National Institute of Standards and Technology,
%     Vol 102, Number 3, May-June 1997
%        http://nvl.nist.gov/pub/nistpubs/jres/102/3/j23mie.pdf

persistent FSspl

if (nargin < 1) || (nargin > 2)
  error('FRESNELS:improperarguments','1 or 2 arguemtns are required.')
end

% default for fresnelType
if (nargin < 2) || isempty(fresnelType)
  fresnelType = 0;
else
  if ~isnumeric(fresnelType) || ~ismember(fresnelType,[0 1 2]) || (numel(fresnelType) ~=
1)
    error('FRESNELS:fresnelType', ...
```

```
      'fresnelType must be scalar, one of {0,1,2} if supplied.')
  end
end

% X must be real, but of any shape.
if any(imag(X) ~= 0)
  warning('FRESNELS:complexarguments','X should be real. Imaginary part will ignored.')
  X = real(X);
end

% preallocate FSint to the proper size
FSint = zeros(size(X));

% flag any negative X, make it positive.
S = X < 0;
X(S) = -X(S);

% transform the type 1 and 2 problems into type 0
switch fresnelType
  case 1
    X = sqrt(2/pi)*X;
  case 2
    X = sqrt(2*X/pi);
end

% The upper limit of the tables is 7.5.
Xlim = 7.5;
% klim is a boolean variable that indicates values that exceed Xlim.
klim = (X > Xlim);
if any(klim(:))
  % we found some values that exceed the limit. Use
  % the rational approximations provided in Mielenz [2]
  % for the associated functions f(z) (see (4a)) and
  % g(z) (see (4b)). The approximations are carried to
  % additional terms beyond that displayed in Mielenz.
  %
  % For abs(X) >= 7.5, these yield results with
  % roughly 15 significant digits.
  xk = X(klim);

  FSint(klim) = 0.5 - (1 - 3/pi^2 ./xk.^4 + 105/pi^4 ./xk.^8 - ...
    10395/pi^6 ./xk.^12 + 2027025/pi^8 ./xk.^16).*cos(pi/2*xk.^2)./(pi*xk) - ...
    (1 - 15/pi^2 ./xk.^4 + 945/pi^4 ./xk.^8 - 135135/pi^6 ./xk.^12 + ...
    34459425/pi^8 ./xk.^16).*sin(pi/2*xk.^2)./(pi^2*xk.^3);

end
klim = ~klim;

% for abs(Xlim) <= Xlim, we will use a spline interpolant of the
% sine integral itself.
if any(klim(:))
  % have we loaded the appropriate spline?
  if isempty(FSspl)
    load _Fresnel_data_ FSspl
  end

  % do the interpolation itself using ppval. This will be
  % better than calling interp1 with the 'spline' option,
  % since it avoids overhead of calling an already created
  % and stored spline. It will be better than pchip or the
  % 'cubic' option for interp1 since the spline will be
  % considerably more accurate.
  FSint(klim) = ppval(FSspl,X(klim));
end

% The Fresnel sine and cosine integrals are odd functions of X,
% so swap signs for any negative X.
```

```
FSint(S) = - FSint(S);

end % mainline end
```

fresnelC.m

```
function FCint = fresnelC(X,fresnelType)
% fresnelC - Fresnel cosine integrals, C(X), C1(X), or C2(X)
% usage: FCint = fresnelC(X,fresnelType)
%
% Fresnel cosine integrals fall into three classes, simple
% transformations of each other. All three types described
% by Abramowitz & Stegun are supported.
%
% The maximum error of this code has been shown to be less
% than 1.5e-14 for any value of X.

persistent FCspl

if (nargin < 1) || (nargin > 2)
  error('FRESNELC:improperarguments','1 or 2 arguemtns are required.')
end

% default for fresnelType
if (nargin < 2) || isempty(fresnelType)
  fresnelType = 0;
else
  if ~isnumeric(fresnelType) || ~ismember(fresnelType,[0 1 2]) || (numel(fresnelType) ~=
1)
    error('FRESNELC:fresnelType', ...
      'fresnelType must be scalar, one of {0,1,2} if supplied.')
  end
end

% X must be real, but of any shape.
if any(imag(X) ~= 0)
  warning('FRESNELC:complexarguments','X should be real. Imaginary part will ignored.')
  X = real(X);
end

% preallocate FCint to the proper size
FCint = zeros(size(X));

% flag any negative X, make it positive.
S = X < 0;
X(S) = -X(S);

% transform the type 1 and 2 problems into type 0
switch fresnelType
  case 1
    X = sqrt(2/pi)*X;
  case 2
    X = sqrt(2*X/pi);
end

% The upper limit of the tables is 7.5.
Xlim = 7.5;
% klim is a boolean variable that indicates values that exceed Xlim.
klim = (X >= Xlim);
if any(klim(:))
  % we found some values that exceed the limit. Use
  % the rational approximations provided in Mielenz [2]
  % for the associated functions f(z) (see (4a)) and
  % g(z) (see (4b)). The approximations are carried to
  % additional terms beyond that displayed in Mielenz.
  %
```

```
  % For abs(X) >= 7.5, these yield results with
  % roughly 15 significant digits.
  xk = X(klim);

  FCint(klim) = 0.5 + (1 - 3/pi^2 ./xk.^4 + 105/pi^4 ./xk.^8 - ...
    10395/pi^6 ./xk.^12 + 2027025/pi^8 ./xk.^16).*sin(pi/2*xk.^2)./(pi*xk) - ...
    (1 - 15/pi^2 ./xk.^4 + 945/pi^4 ./xk.^8 - 135135/pi^6 ./xk.^12 + ...
    34459425/pi^8 ./xk.^16).*cos(pi/2*xk.^2)./(pi^2*xk.^3);

end
klim = ~klim;

% for abs(Xlim) <= Xlim, we will use a spline interpolant of the
% cosine integral itself.
if any(klim(:))
  % have we loaded the appropriate spline?
  if isempty(FCspl)
    load _Fresnel_data_ FCspl
  end

  % do the interpolation itself using ppval. This will be
  % better than calling interp1 with the 'spline' option,
  % since it avoids overhead of calling an already created
  % and stored spline. It will be better than pchip or the
  % 'cubic' option for interp1 since the spline will be
  % considerably more accurate.
  FCint(klim) = ppval(FCspl,X(klim));
end

% The Fresnel sine and cosine integrals are odd functions of X,
% so swap signs for any negative X.
FCint(S) = - FCint(S);

end % mainline end
```

generateTables.m – To be called once before using fresnelC.m and fresnelS.m

```
% ==============================================================
%     Code used only to generate and save the integral tables
% ==============================================================
function generateTables

% Generate the integral tables, more accurate than Abramowitz &
% Stegun provide, since they give only 7 digits.
FresnelCObj = @(t) cos(pi*t.^2/2);
FresnelSObj = @(t) sin(pi*t.^2/2);

p = 1.75;
T0 = linspace(1,7.5.^p,501).' .^(1/p);
dt = T0(2) - T0(1);
T0 = [linspace(0,1 - dt,ceil(1./dt))';T0];
plot(diff(T0))

n = length(T0);
FC75 = zeros(n,1);
FS75 = zeros(n,1);

h = waitbar(0,'Computing Fresnel integrals');
for i = 2:n
  waitbar(i/n,h)
  FC75(i) = quadgk(FresnelCObj,0,T0(i),'abstol',1.e-16,'reltol',100*eps('double'));
  FS75(i) = quadgk(FresnelSObj,0,T0(i),'abstol',1.e-16,'reltol',100*eps('double'));
end
delete(h)

% Turn them into splines, then save the splines. These splines are
```

```
% first built in a Hermite form, since I can supply the 1st and second
% derivatives of the function. Then I turn them into a pp form, for use
% in fresnelC and fresnelS.
FCspl = hermite2slm([T0,FC75,FresnelCObj(T0), -pi*T0.*sin(pi*T0.^2/2), ...
  -pi*(sin(pi*T0.^2/2) + pi*T0.^2 .*cos(pi*T0.^2/2))]);
FCspl = slm2pp(FCspl);

FSspl = hermite2slm([T0,FS75,FresnelSObj(T0),pi*T0.*cos(pi*T0.^2/2), ...
  pi*(cos(pi*T0.^2/2) - pi*T0.^2 .*sin(pi*T0.^2/2))]);
FSspl = slm2pp(FSspl);

save _Fresnel_data_ FCspl FSspl


% test the result
clear functions

n = 1000;
T = sort(rand(n,1)*10);
FCquad = zeros(n,1);
FSquad = zeros(n,1);
for i = 1:n
  FCquad(i) = quadgk(FresnelCObj,0,T(i),'abstol',1.e-16);
  FSquad(i) = quadgk(FresnelSObj,0,T(i),'abstol',1.e-16);
end
FCpred = fresnelC(T,0);
FSpred = fresnelS(T,0);

subplot(1,2,1)
plot(T,FCquad - FCpred,'.')
grid on
subplot(1,2,2)
plot(T,FSquad - FSpred,'.')
grid on

end
```